Community Detection and Author Disambiguation in a High Energy Physics Citation Network

CS224W Project Final Report

Juan Pablo Alperin Nicole Rodia Benjamin Quimby

1. Introduction

Research is never carried out in isolation. Scholars build on each other's findings, work together on problems, work on problems in parallel, and often verify each other's results. The success of the academic enterprise is dependent on all its participants communicating their ideas, techniques, and findings with each other and with the public. Given the public importance of scholarly research, we believe that more attention must be given to understanding the dynamics of scholarly communications and to address problems facing the scholarly community.

In this project, we seek to explore and understand the types of communities that form within a specific discipline, high energy physics. Although work has been done to detect communities in 'science', communities and community detection algorithms have not been carefully explored for specific sub-disciplines such as that of high energy physics phenomenology. We explore the applicability of different network detection algorithms and try to evaluate their effectiveness at detecting 'meaningful' communities.

Our approach, however, is not simply aimed at detecting meaningful communities for the sake of identifying them. We propose a novel approach to use the detected communities as a tool for disambiguating author names. Most bibliographic databases contain author names, but they rarely record a unique identifier for each individual. Since multiple people can share the same name, even within a single discipline, there is a need to develop algorithms that can computationally determine which papers are authored by the same individual. By determining which duplicate names are different individuals, it will be possible to show readers work by the same author, provide a means for authors to track their publications, and enable universities to better determine their scholarly output.

Identifying individuals and correctly attributing bibliographic records to real people is an important problem for academics, academic institutions, and funding/granting agencies alike. From their different perspectives, each of these groups has an interest in identifying their scholarly output for evaluation and understanding of their academic impact. When looking for citation impact in existing databases, scholars want to identify papers they have written, universities want to identify papers written by their professors, and granting agencies want to identify the work of their grantees. Through computer-aided author disambiguation, it will be possible to provide services to study the scholarly output of an individual or an institution, as well as provide important services for the discovery of research, such as listing works by the same author.

2. Related Work

Author Disambiguation
Author disambiguation is a special case of *entity resolution*—discovering underlying entities and mapping database references to these entities (in our case, a specific person). Traditionally, entity resolution is done by looking at pair-wise comparisons of attributes for each record and resolving the entity based on attribute similarity. Famously, Bhattacharya & Getoor (2007) extend the traditional model by using relational data to aid in resolution.

There have been many other proposed methods of author disambiguation in recent literature. Some methods rely on building statistical models, such as those in Zhang et al. (2007). This particular approach requires empirically setting constraint weights for a Hidden Markov Random Field model and relies on each node having a complete set of relevant metadata (affiliation, email address, journal information, etc.) that may not always be present. Other methods look at relational attributes of the authors, such as co-authorship and co-citation. Kang et al. (2009) look solely at co-authorship of papers, but extend their information base with co-authorship data gleaned from web queries.

In this paper, we develop a novel technique for author disambiguation that extends the work carried out by others. We describe a method that, like Bhattacharya & Getoor (2007), uses the information contained in the relation between papers and like Kang et al. (2009) uses the citation information. However, our approach aims to take advantage of the information contained in the citation graph structure to identify individuals. In particular, we apply community detection algorithms to extract the underlying structures in the citation graph and develop an approach for using that information to disambiguate authors.

Community Detection and Information Flow
Various heuristic algorithms are used to determine communities in real-world graphs. The Girvan-Newman (2002) algorithm uses edge betweenness centrality, defined as the number of shortest paths between pairs of nodes along that edge, to find community boundaries. This technique gives a hierarchical decomposition of the network communities, with better results than traditional hierarchical clustering methods, but has complexity $O(n^3)$, making it too slow to analyze a large network.

Newman & Girvan (2004) define modularity as a measure of the quality of the division of a network into communities, where a good division is characterized by many edges within communities and only a few between them. A fast modularity maximization algorithm that uses hierarchical agglomeration is presented in Clauset et al. (2004) and Newman (2004). This algorithm employs a bottom-up approach, starting with every node in its own community and at each step merging communities that provide the best increase in modularity. For hierarchical networks that are most commonly found in real life, this algorithm achieves a run time of $O(n \log^2 n)$. Newman (2006) presents a community detection method based on optimal modularity known as the spectral modularity maximization algorithm. This technique is based on eigenvector decomposition of the modularity matrix, a modified graph adjacency matrix, and has a run time of $O(n^2 \log n)$.

Rosvall & Bergstrom's (2010) Infomap algorithm uses significance clustering on parametrically resampled networks to map change in a network over time and visualizes merging and splitting of clusters with alluvial diagrams. The paper uses a large scientific journal citation dataset as a canonical example.

Dynamic Sliding Window
To measure scientific consensus, Shwed & Bearman (2010) examine community structure in academic paper citation networks. To model changes in the network over time, Shwed & Bearman use a dynamic window approach, in which the median of the distribution of citation ages serves as the window width.

3. Academic Paper Citation Dataset

Dataset Description
We believe there is value in working with a group of papers and citations from a relatively homogeneous community, so we chose the ArXiv HEP-PH (high energy physics phenomenology) citation graph, extracted from the e-print arXiv, which covers many of the papers and citations from a period of 10 years beginning in 1993[1].

The data consists of 34,546 nodes (papers) and 421,578 edges (citations), which will be used as an unweighted, directed, acyclic graph. The resulting citation graph is not complete, as it only includes citations for papers that can be found in arXiv.org within the 1993-2003 time period. The dataset also comes with no guarantees that citations to papers in the dataset are included.

Data Collection
We complement the data with titles, abstracts, publication dates, and author names for each of the papers. The additional metadata for each of the paper was collected over a 2 day period in October 2011, using the arXiv.org API[2]. The citation graph, made available through the 2003 AMC KDD Cup 2003, includes the arXiv identifier which was subsequently used to query the arXiv API. The API identifies the available metadata, including author names (as entered into arXiv by the uploader). In total, the papers in the dataset were written by 77,507 authors and co-authors[3] (an average of 2.24 co-authors per paper). However, out of the 77,507 authors, there are only 14,897 distinct names. In fact, there are 7 names that are listed as authors of over 100 papers, 98 names that are listed as authors of over 50 papers, and over 6,000 names that appear in at least 2 papers. ArXiv.org does not track which instances of these names are the same individual. The large amount of name duplication and the lack of mechanism for disambiguating indicate the magnitude of the challenge that our project aims to solve.

Author Name Consolidation
When working with this same dataset, McGovern et al. (2003) use only a heuristic approach to consolidating author names and do nothing to attempt disambiguation. Their resulting dataset of

[1] Details of the dataset can be found at http://snap.stanford.edu/data/cit-HepPh.html

[2] http://arxiv.org/help/api/index

[3] This does not imply 77,507 individuals, simply that this many names can be found as author or co-authors of papers.

9,200 author names has subsequently been mistakenly used as the benchmark for the oft-cited entity-resolution work of Bhattacharya and Getoor (2007). Lacking access to McGovern et al.'s consolidated dataset (and to some additional attributes that would help to re-create it), we follow a similar approach, but choose a simpler consolidation strategy. Using python's difflib library, we match names on the full string with different cutoff points to group together authors with similar names. We used a fairly strict cutoff of 0.95. This cutoff can be thought of as roughly a percentage-string similarity, where names are grouped together if they are very similar to each other according to this threshold. The resulting set has 14363 unique author names (534 names were consolidated). This reduced list provides a useful starting point for attempting disambiguation.

Without disambiguating, but after consolidation, we observe that most author names have published few papers (3,432 names have a single paper), while a few names have published many papers (19 consolidated names have over 100 papers attributed to them).

4. Method and Algorithm Descriptions

Our basic approach is to first apply the dynamic moving window technique to the full citation dataset to extract time-based subgraphs. Second, we find communities in each time window using a community detection algorithm, with the goal of determining topic-based communities. We use the communities to construct a meta-graph that describes community evolution over time. Our approach is based at using the information embedded in both the communities at each time point and the constructed community evolution graph to identify individuals from the existing author names.

Dynamic Moving Window

As scholars move through their careers, they may enter and leave different communities within their field. As such, we track the changes in the communities over time by using a dynamic sliding window. With this approach, algorithms are run once for each year and, for every given year, we will analyze a sub-graph composed of the union of the following three sub-graphs: (1) G1 = nodes corresponding to all the papers published in the given year; (2) G2 = nodes corresponding to all the papers published within x years of the papers in G1, where x is the median age of the citations found in the papers of G1; and (3) G3 = all the nodes (papers) and edges (citations) directly connected to G1 and G2, regardless of publication date. This dynamic window approach is similar to that proposed by Shwed & Bearman (2010).

By tracking the community detection algorithms over the evolution of the network, we have a point of comparison to the results of the same community detection algorithms over the entire corpus. It is possible that some communities remain stable through time, thereby appearing in the whole dataset. Other communities may appear or disappear as specific groups are created or dissolved throughout the ten year window spanning the dataset.

The dynamic moving window approach provides a finer level of granularity for detecting communities and attempting author disambiguation. While a specific author may belong to multiple communities within high energy physics, it is less likely (though not impossible) that he or she will belong to very different ones in a given time window. By dividing our dataset into

subgraphs using these windows, we observe an strong edge effect caused by the data from before the creation, and popularization, of arXiv.org.

The grouping of the data into time windows is illustrated in Figure 1. Figure 1(a) shows the increasing length of the median citation age, with all years before 1997 having a median age of one year, those between 1997 and 2001 having a median age of two years, and the last two years having a median age of three years. It is likely that the true median is at least three years, and that the first few years of data are lowered by missing data. Figure 1(b) corroborates this story, as we see an increasing number of papers per year in each moving window, increasing linearly over time, with a slight drop-off for 2003, likely caused by the fact that the dataset ends with papers in April 2003.

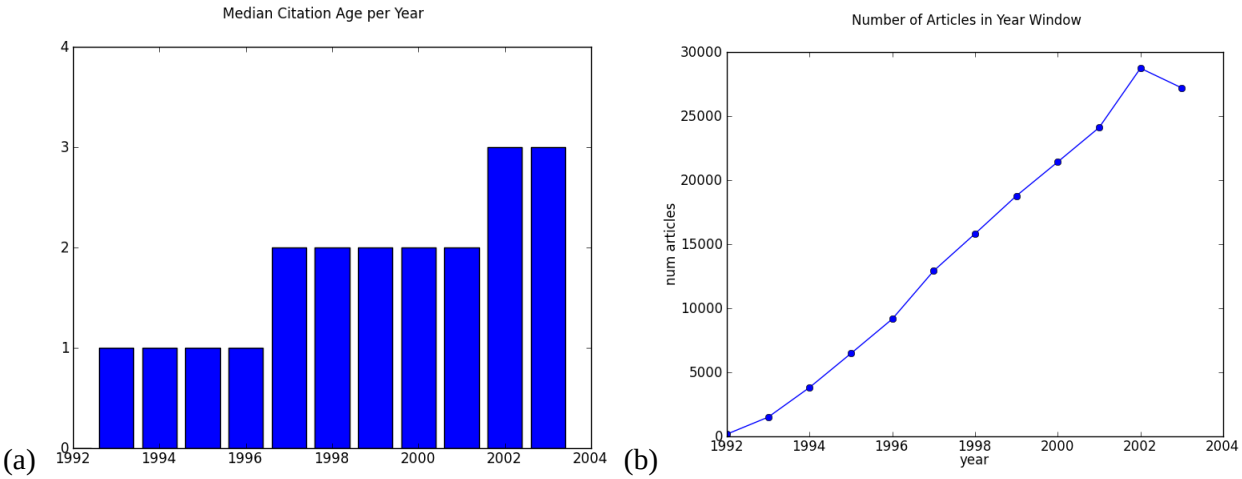

Figure 1. (a) median citation age per year. (b) Number of articles in each dynamic window.

Community Detection Analysis

Network communities are sets of nodes with many connections inside the group of nodes and few connections to the rest of the network. After consideration of several community detection approaches, we tested the applicability of three well-known community detection algorithms: Rosvall & Bergstrom's (2010) Infomap, Clauset et al.'s (2004) agglomerative modularity maximization algorithm, and Newman's (2006) spectral modularity maximization algorithm.

Infomap

For the Infomap algorithm, we downloaded the version on the paper's companion Website mapequation.org, written in C. To run this algorithm, we read our graph's edge list using NetworkX and re-wrote the graph in Pajek format, used to input data to Rosvall & Bergstrom's implementation. All year subgraphs were processed in less than half an hour on a 2.0 GHz laptop with 2 GB of RAM. The resulting output files, written in Pajek form, could then be parsed in python to create a representation of the communities.

The Infomap algorithm generates significance clusters and alluvial diagrams to map change in networks. It works as follows: (1) cluster the original networks for each time point, partitioning the network into non-overlapping modules; (2) generate and cluster the bootstrap networks for each time point, using a parametric bootstrap technique; (3) for each time point, identify journals

that are significant in their own clusters and clusters that are significantly distinct from other clusters; (4) generate an alluvial diagram to illustrate changes between the time points.

The clustering method, used in steps (1) and (2), uses the map equation as an objective function and a new algorithm to partition the network by minimizing the expected description length of a random walk across the network. To generate a bootstrap replicate network of the original network in step (2), each link weight is resampled from a Poisson distribution with the mean equal to the link weight in the original network. Approximately 1,000 of these bootstrap networks are created. Step (3), the significance clustering step, uses simulated to annealing to search for the largest subsets of journals in the bootstrap networks to identify significant journals, and to search for clusters whose significant subset is separated from other significant subsets in the bootstrap networks to identify significantly distinct clusters.

Agglomerative Modularity Maximization Algorithm

For the Clauset et al. algorithm, we first attempted to write our own version from scratch in Python, since there was no NetworkX module available. Unfortunately, due either to programmer error or a bad heap/queue implementation, we could not run the algorithm on large data without getting a segmentation fault. We then discovered an implementation of the algorithm distributed with SNAP. SNAP is capable of reading in the dataset's original edge-list and produces an output that could be parsed in a similar way to Infomap's output communities.

Both the agglomerative modularity maximization algorithm and the spectral modularity maximization algorithm use the modularity score as a fundamental metric to measure the quality of of the community partitioning of a graph. Newman & Girvan (2004) define modularity as a measure of the quality of the division of a network into communities, where a good division is characterized by many edges within communities and only a few between them. Mathematically, modularity can be expressed in terms of a matrix computation.

First, we define a $k \times k$ symmetric matrix **e**. Each element e_{ij} of matrix **e** is the fraction of all edges in the network that connect vertices in community i with vertices in community j. The trace of matrix **e**, **Tr e** = $\sum_i e_{ii}$, is the fraction of network edges that connect vertices within the same community. Next we define row or column sums $a_i = \sum_j e_{ij}$. The a_i are the fraction of edges in the network that connect to vertices in community i. Modularity is defined as $Q = \sum_i (e_{ii} - a_i^2)$ = **Tr e** - $\|\mathbf{e}\|^2$, where $\|\mathbf{x}\|$ is the sum of the elements in the matrix **x**. Intuitively, this modularity definition represents the fraction of within-community edges minus the expected value of the within-community edges in a network with the same community divisions, but random connections between vertices. Q can take on values from 0 to 1, where $Q = 0$ means that the number of within-community edges is equivalent to the number in a random network, and values close to $Q = 1$ mean that there is strong community structure.

The agglomerative modularity maximization algorithm employs a bottom-up approach, starting with every node in its own community and at each step merging communities that provide the best increase in modularity. The algorithm runs until the entire graph is one community, keeping track of the merges along the way in the form of a dendrogram. Then, one can backtrack to find the optimal cut in the dendrogram, which will produce a set of communities that maximize the modularity of the graph. The algorithm also makes clever use of balanced binary trees and max

heaps to ensure that calculations are only being done on non-zero cells of the adjacency matrix instead of the entire sparse matrix.

Spectral Modularity Maximization Algorithm

We implemented our own version of Newman's spectral modularity maximization algorithm, based on the implementation currently available in Ben Edwards's NetworkX community branch, with the Leicht and Newman extensions for directed graphs. The base algorithm divides graphs into two communities, however, our interest in producing smaller communities required an implementation to recursively divide each community until no improvement in modularity could be found. The resulting code, currently only available in this project's repository, will be contributed to the NetworkX open source project and represents a meaningful code contribution. Details of the implementation can be found in the code repository.[4]

The spectral modularity maximization algorithm is based on spectral graph partitioning and utilizes eigenvector decomposition of the modularity matrix, a modified graph adjacency matrix. The algorithm functions as follows: (1) a pre-processing step constructs the matrix representation, (2) a decomposition step computes the eigenvalues and eigenvectors, and maps each node to a lower-dimensional representation based on the eigenvector grouping, and (3) assign nodes to one of two clusters based on the new lower-dimensional representation.

To divide a network into more than two communities, the approach described in Newman (2006) is to continue dividing into groups of two, until a proposed division makes a zero or negative contribution to the modularity of the network, in which case that subgraph is not further divided. When dividing any of the subgraphs in the network no longer improves the overall modularity, the algorithm terminates. In this paper, Newman proposes a definition of a community as an indivisible subgraph, based on the modularity score.

Leicht and Newman (2008) extend the modularity maximization approach to incorporate information contained in edge directions. Since our citation network is directed, we implemented this extension in the spectral modularity maximization algorithm. The intuition behind incorporating edge directions is that the "surprise" of a seeing a particular edge, which modularity captures by comparing the actual number of edges between communities to the expected number, should take direction into account. If node A has high in-degree and low out-degree, and node B has high out-degree and low in-degree, then it should be more surprising to see an edge from A to B and less surprising to see the opposite.

Evaluating Communities

In an attempt to understand the content of the communities, and to see if the detected communities are topically related (as we expect intuitively), we apply a technique known as latent semantic indexing (LSI). We have used LSI both to create clusters and to evaluate the clusters produced by the algorithms described above.

We applied LSI on the metadata available for our dataset: titles and abstracts. In LSI, the text is first tokenized and filtered to remove words that belong to a manually created stop word list

[4] http://code.google.com/p/cs224w-citation/source/browse/trunk/networkx-community/networkx/algorithms/community/partition_recursive.py

(consisting of articles, conjunctions, common prepositions, and other words that provide no indication of content). After this we reduce tokens to their stemmed versions by using the Porter stemmer. This leaves many smaller words unchanged, but converts forms of words to a common root - e.g. symmetric and symmetrically would both be stemmed to symmetri. Finally, we throw out all tokens which only appear once in the corpus (or, collection of all the tokens from all the text). Now each paper can be represented by a feature vector, in which the terms from the corpus are the positions and the number of occurrences of that term in the paper title/abstract are the values. In practice, we want to use a reweighted value instead of a raw term count. We tried two common reweighting methods to convert our raw counts. The first is called Term Frequency - Inverse Document Frequency (TF-IDF), and encodes the idea that a term that appears in nearly every document ("physics", perhaps for this collection) is not as useful in distinguishing between documents as a rarer term. TF is in this case just the raw term counts, and is multiplied by IDF. IDF is calculated by taking the log of the total number of documents over the number of documents that contain the term in question. The second method is log-entropy. Here, instead of raw counts we take the log of the term count + 1, the log portion of the equation, and multiply by the entropy portion of the equation. The entropy portion is calculated as

$$1 + \sum_j \frac{p_{ij} \log p_{ij}}{\log n}$$

Here p_{ij} is the probability of word *i* appearing in document *j*, which is empirically given by the raw count of term *i* in document *j*, over the number of times term i appeared in the whole corpus.

From these feature vectors, we were able to use LSI to find the most common topics across the collection of abstracts, along with the words that contribute to those topics. We then assign each abstract to the category for which it scores most highly based on its constituent terms. We then tried to overlay this category metadata onto the nodes in our network, and look at the topical homogeneity of the communities that are detected by each algorithm. The results of this are further discussed in the Results section below.

We also take advantage of the temporal nature of our dataset by comparing clusters over time. We use the dynamic moving windows as a unit of study and look at how papers move between communities over time. Since each time window contains papers from, at the very least, all papers from the year before, it is possible to see paper x's cluster at time t_i and again at t_{i+1}. With random clusters, we expect a community of papers at time t_i to be randomly distributed among the communities at time t_{i+1}. However, we observe that the clustering algorithms consistently place the same papers into the same communities over time.

We calculate the proportion of papers that stay together in a community by constructing a *cluster evolution* weighted-directed graph C. In this graph, each node is a community and we place an edge between communities that have papers in common between communities at time t_i and time t_{i+1}. The weight of each edge is defined by the number of papers shared by each community. More precisely, this graph is defined by the following algorithm:

```
#initialize graph with one node for each community and time year
for t in years:
    for c in communities[t]:
        add node(t,c)
```

```
# now add edges
for t in years:
    for clusters_at_ti in communities[t]:
        for clusters_at_ti_plus_1 in communities[t+1]:
            intersection=set intersection(cluster_at_ti,
cluster_at_ti_plus_1)
            if intersect > 0:
                add edge (cluster_at_ti, cluster_at_ti_plus_1, weight=size of
intersection)
```

We calculate the proportion of nodes that stay together by taking the maximum of the outgoing edge weights for each community node. For example, if community #1 at t=1992 (hereafter, c[1992, 1]) has 10 papers and c[1993,1] has 8 of those papers, c[1993,2] has 1, and c[1993,3] has 1, then we say that the proportion of nodes that stay together is 0.8.

Author Disambiguation

Finally, we use the communities detected through the previously described algorithms in order to disambiguate authors. The intuition is that instances of an author's name are more likely to refer to the same person within the context of a given community than in the entire corpus. We also assume that within the time period of a year it is unlikely for an author to publish papers in many different topics, and so it is likely for the nodes representing that author's papers to lie within the same community. We therefore take a given time window and say that if a (consolidated version) of an author's name appears multiple times in the same community, it belongs to a single *person*. If the same name appears in different communities within the *same* time window, it belongs to *different* persons who share the same name.

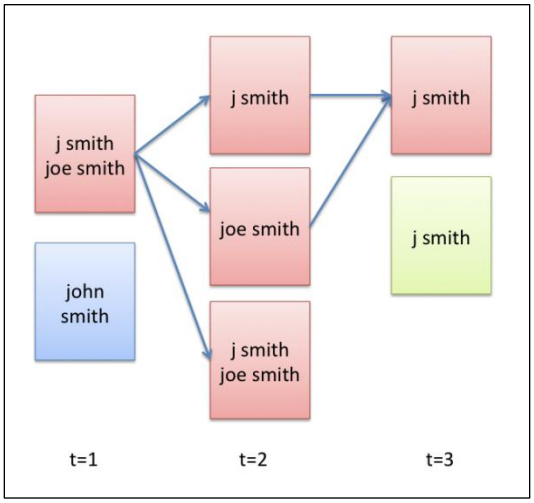

Figure 2. Author disambiguation on cluster evolution sub-graph.

We subsequently use the cluster evolution graph C, described above, to track people's movement between communities over time. At time t_i we identify all the different *persons* with similar names. We subsequently consider all people with the similar names who are connected in the cluster evolution graph to be the same individual. At each time step, any names that have not been labeled as an already-found person are considered a new person. Figure 2 shows a simplified version of the cluster evolution graph, with communities that contain three similar names (j smith, joe smith, and john smith). Each square represents a community at a given time

window, the edges represent communities that share members between time periods, and the colors represent individuals. In this example, the algorithm would detect three individuals that share similar names.

5. Results and Discussion

Evaluating Communities

Results of running LSI on the title/abstract of the papers looked initially unpromising. The vast majority (20,000-25,000) of the papers were being clustered into one topic, while the others were grouped into clusters ranging from a few papers to a few hundred. When we discovered that the results of the graph-based community detection algorithms were giving us a similar distribution of a few disproportionately large communities with many small communities, we began to think that perhaps the LSI results could be showing us a compatible view of our community structure. So, for each community detected by the graph-based algorithm, we determined which LSI-discovered topic had the most associated nodes present. What we would have liked to see is many of the small communities matching up with small topics, but unfortunately the majority of the communities had the one large topic as their main topic (determined by plurality). This does not necessarily mean that our detected communities are poor, but it does cast some doubt on our assumption that the communities we detected would have some underlying topical homogeneity.

Of the three community detection algorithms that we used, the Infomap and agglomerative modularity maximization algorithms gave somewhat reasonable results, dividing the dynamic window subgraphs into a number of communities of various sizes. The spectral modularity maximization algorithm with edge directions, however, gave extremely poor results. For most of the subgraphs, the algorithm left the network undivided, or only divided into two communities. The modularity scores of these community division were very low, generally less than 0.1, indicating that community structure was essentially absent from these networks. Further, the Python implementation was extremely memory intensive, making it difficult to run the algorithm on the larger subgraphs. Consequently, we have not included the results of the spectral modularity maximization algorithm in this section.

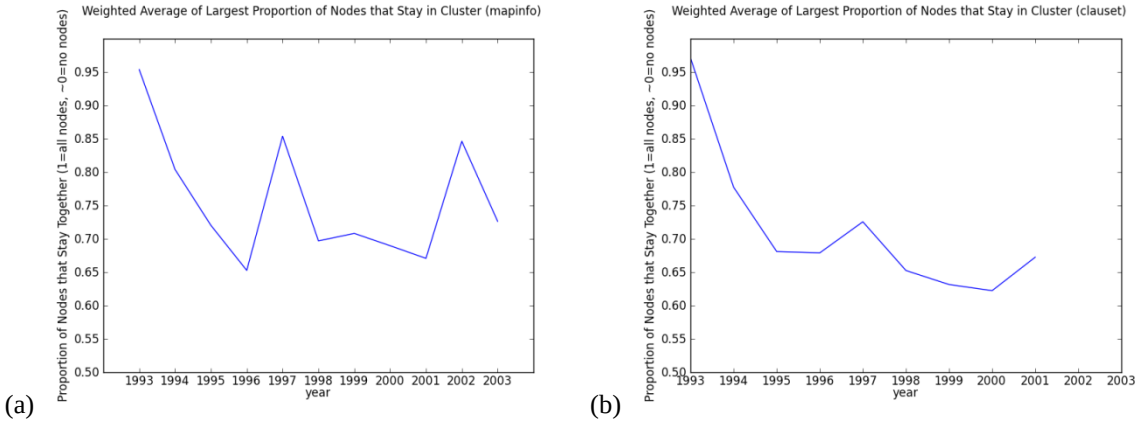

Figure 3. Proportion of papers that stay together in clusters between years for (a) Infomap and (b) Clauset.

Figure 3 shows the proportion of articles that stay together in a cluster between years for both the Infomap and the Clauset algorithms, as measured simply by the largest sub-community from one year's sub-graph that stayed together in the next year's subgraph . While there is a good amount of variance, especially in the Infomap graphs, both algorithms arguably do a good job of keeping the same nodes grouped together from year to year. The Infomap algorithm appears to have a slight edge based on our data, keeping between 65% and 85% of the nodes consistently in the same community, whereas the Clauset algorithm keeps nodes in the same community between 60% and 80% of the time.

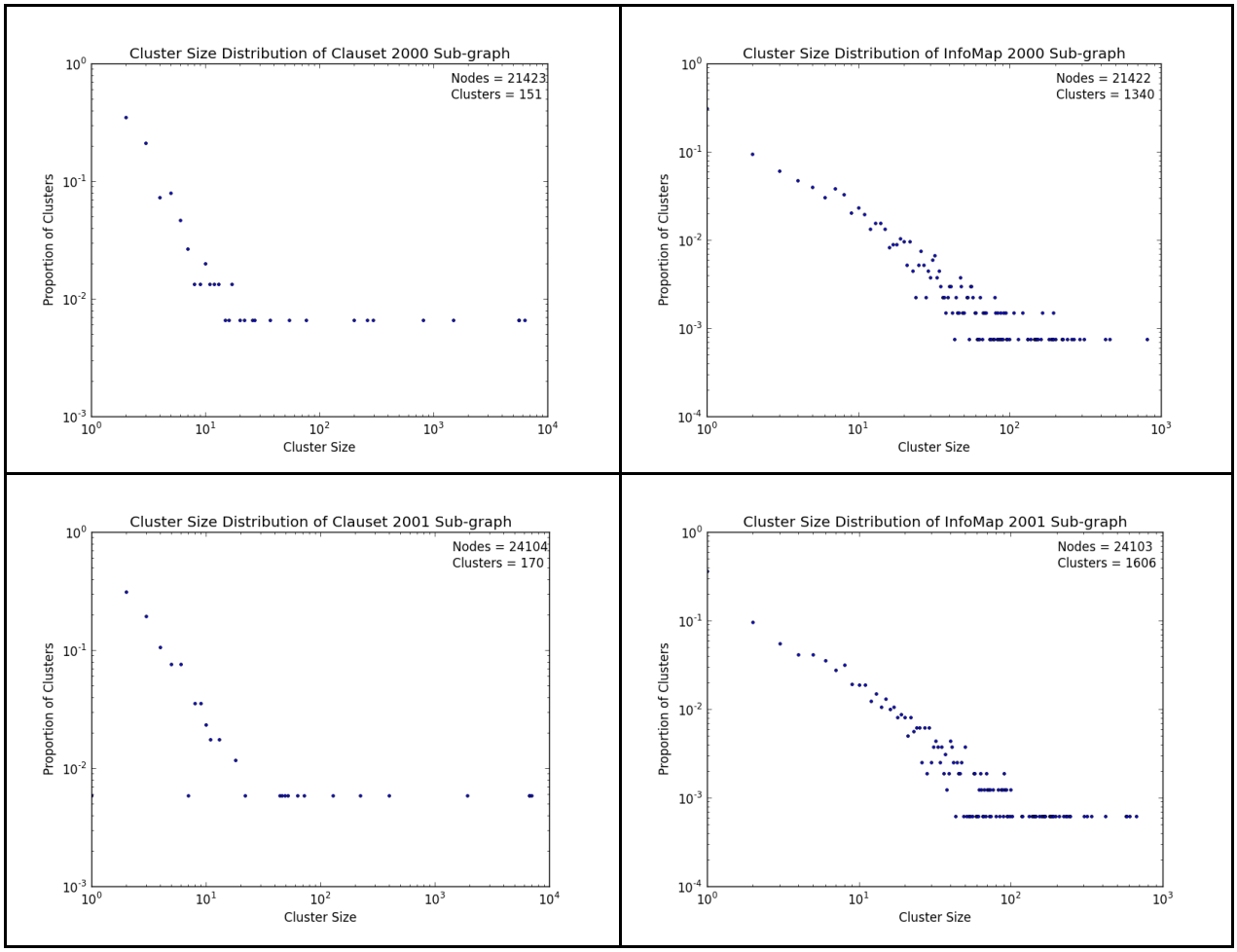

Figure 4. Comparison of Community Size Distribution between Clauset and Infomap Algorithms

Besides consistency of communities, we observed some interesting differences in the characteristics of the communities detected by each of our community detection algorithms. Figure 4 shows the community size distribution of our communities for the years 2000 and 2001 (all the other years have similar distribution profile). The cluster size distribution generated by the InfoMap algorithm follows the same familiar power law distribution that is seen in other properties of real-world graphs, such as degree distribution, strongly connected component (SCC) size distribution, etc. The cluster size distribution generated by the Clauset algorithm is characterized by a small handful of disproportionately large clusters, with the majority of clusters being smaller than 10. Given that edges join papers that have cited each other, we had expected

communities to be topically similar. The existence of a small number of very large communities is consistent with the intuition that, within a given discipline, the large majority of papers are related to one another. However, the very large number small communities seems to break from our assumptions of how communities would be formed, since it seems unlikely that there are many papers not related to the rest of the field.

Even though the distribution of community sizes in Infomap looked "better behaved" than the Clauset distribution, we believe that this is a function of the constraints of our network on the Infomap algorithm, and not a measure of the goodness of the algorithm. Infomap uses random walks through the graph and determines communities essentially by where it spends most of its time (inter-community travel is less probable than intra-community travel). Since our graph is a directed, acyclic graph (DAG), Infomap finds communities that are anchored by nodes with high in-degree and comparatively low out-degree. These high in-degree nodes can be thought of as sinks, where many random paths through the network consistently end up. These communities are in contrast to the communities the algorithm would find in an undirected graph (or directed graph with cycles), in which a random walker would revisit many of the same nodes in a given walk. So, we propose that for a DAG, Infomap produces communities that are highly correlated with the degree distribution of the network, hence the power law distribution of community size. On the other hand, the communities detected by Clauset are based on maximizing modularity, and hence tend to cluster together nodes with a more balanced in-out degree ratio. As an example, for the 1995 year graph, we calculated the average in-degree to out-degree ratio in communities. For Infomap this number was 1.398, while for Clauset this number was 1.021.

Author Disambiguation

The results for author disambiguation have not proven to be reliable, but show promise in several regards. There is no known way to systematically evaluate the quality of the author disambiguation through automated means, but a visual inspection of some of the data allows us to understand the successes and failures of our approach. We tried the author disambiguation on several different sets of names: those with the most number of variations, those with the most number of papers, and a randomly selected set. The results of the top ten names on each of these lists can be found in Appendix A, and are described in summary here.

We test our resulting data to see the average number of people detected for each name. We expect the majority of names to be associated with a single person, since the most names are unique. However, we observe that both clustering algorithms are overly aggressive in the division of names into multiple people. In this measure, Figure 5 shows that Clauset consistently produces fewer persons, regardless of the number of papers associated with that name. On average, the Clauset-based clusters produce 3.0 persons per name, while the Infomap-based clusters produce 7.5 persons per name. We subsequently turn to specific examples to gain a better understanding of the structure of the cluster evolution graph and of the results of our algorithm.

Due to the smaller number of papers per name, the randomly selected set of authors allows for a more careful manual inspection of the author disambiguation approach. Upon observation, we see several trends that emerge in both the Clauset and the Infomap results. After manually verifying all of the papers for the 10 randomly selected authors under both sets of clusters, we

found no instances where the algorithm incorrectly identified two different people as the same person (based on additional information we could glean from looking at the paper itself, namely the author's affiliation). We note, however, that in one example in the sample, author Mihir Worah, a subset of the papers contain two different affiliations (University of Chicago and University of California at Berkeley) and yet our algorithm identifies the papers with both affiliations as belonging to one person. Hence, our approach succeeds in a case that might prove difficult for a system that puts an emphasis on attribute data.

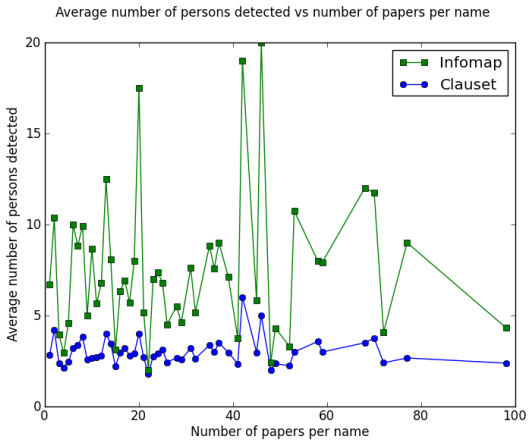

Figure 5. Number of persons detected vs. number of papers published by that name

However, when looking at all the papers attributed to the name Worah, our algorithm run on the Clauset communities, attributes this set to two people, despite the fact that all the papers in the dataset seem to belong to the same real person. Looking at the output from all of the datasets, it becomes apparent that the majority of new persons detected are created in the early years where the name appears in the dataset. That is, a single name appears in multiple clusters within a single time window. This is contrary to our assumptions about individuals only appearing in a single cluster due to only publishing in one community at a time. After our analysis, we can say that the community detection algorithms analyzed do not, for a single time window, correctly identify individuals. There appears to be a trend towards dividing the papers of a single individual into multiple communities.

Despite this limitation, the cluster evolution graph approach shows promise for correctly identifying individuals in subsequent years, especially in the case of the Clauset-based clusters. We observe that the same name remains connected throughout the evolution of the clusters, and is thus not split into many people, even for names associated with many papers in many clusters. If names were randomly distributed among the papers, and our algorithm randomly associated clusters, we would expect more people to be detected by our algorithm.

We were unable to identify instances where the same name belonged to two different individuals, except in the cases where our consolidation strategy had been overly aggressive. In Appendix A, we include a sample that uses a less strict name consolidation strategy (cutoff = 0.8). This consolidation strategy groups names that are evidently not the same person, yet, our algorithm does not succeed at separating out the different individuals.

In short, the author disambiguation approach is successful at grouping together many of the papers produced by the same name, especially in later time windows. In its current form, our implementation only ever separates a name into multiple individuals and does not allow for individuals to be joined together into a single person. Considering the shortcomings of our approach, especially for the earlier years, we might explore the possibility of merging individuals beyond the first time window.

Our approach was hindered by the number and size of the detected communities in each time window. The existence of a few very large communities and many small ones at each time window groups together many instances of a given name as a single individual, and increases the probability of linking to the largest cluster in the following time window.

6. Future Work

The communities that were detected in our citation network were not suitable for doing author disambiguation the way we had originally envisioned. However, our study shows that there is value in detecting communities, constructing a cluster evolution network, and using the network structure for the purposes of disambiguating authors. One future avenue of work to explore would be using community detection to develop features for a classifier-based or hybrid system. One could use a Boolean feature of whether or not the pair of names in question are in the same community, or even come up with some notion of distance between communities. For example, if citation networks do tend to have one or only a few large central communities, then perhaps only the peripheral communities are actually meaningful. We could then calculate the shortest path through the central "community" between two peripheral communities as a proxy for measuring how closely related those two communities are.

Given the propensity for our algorithm to split names into multiple people (when in fact, the name belongs to a single person), it would be valuable to explore strategies for merging names back into individuals in subsequent time windows. Merging names in future windows would address the issue that the first time windows do not benefit from the information implicit in the cluster evolution graph.

We also believe that there is still a benefit to looking at the topical similarity of papers within time windows, and that once again this might serve best as a feature for a classifier or smaller piece of a larger hybrid system. One possible approach would be to look at the kind of clustering that occurs when we apply LSI to a citation network of a much more topically diverse nature. We had originally intended to use the ISI Web of Science citation network for the purposes of this project, but could not obtain permission to use the data. We think that we would see much better results, in terms of well-formed topical communities, if the pool of papers were not all from a highly specialized field such as high-energy physics phenomenology.

References

1. Bhattacharya, I. and Getoor, L. 2007. Collective Entity Resolution in Relational Data. *ACM Transactions on Knowledge Discovery from Data*, 1(1), Article 5
2. Clauset, A., Newman, M. E. J., and Moore, C. 2004. "Finding community structure in very large networks." Phys. Rev. E 70, 066111
3. Girvan, M. and Newman, M. E. J. 2002. "Community structure in social and biological networks." Proc. Natl. Acad. Sci. USA 99, 7821–7826
4. Kang, I.S., Na, S.H., Lee, S., Jung, H.,Kim, P., Sung, W.K. and Lee, J.H. 2009. "On co-authorship for author disambiguation." Information Processing & Management, 45(1): 84--97.
5. Leicht, E. A. and Newman, M. E. J. 2008. ''Community Structure in Directed Networks.'' Physical Review Letters 100:118703.
6. McGovern, A., Friedland, L., Hay, M., Gallagher, B., Fast, A., Neville, J., and Jensen, D. 2003. Exploiting relational structure to understand publication patterns in high-energy physics. *ACM SIGKDD Explor. Newsl*., 5(2):165–172.
7. Newman, M. E. J. 2004. "Fast algorithm for detecting community structure in networks." Phys. Rev. E 69, 066133
8. Newman, M. E. J. 2006. ''Modularity and Community Structure in Networks.'' Proceedings of the National Academy of Sciences of the United States of America 103:8577–82.
9. Newman, M. E. J. and Girvan, M. 2004. "Finding and evaluating community structure in networks." Phys. Rev. E 69, 026113.
10. Rosvall, M. and Bergstrom, C.T. 2008. "Maps of random walks on complex networks reveal community structure." PNAS 105: 1118–1123
11. Shwed, U. and Bearman, P. S. 2010. "The Temporal Structure of Scientific Consensus Formation." American Sociology Review, 75: 817
12. Zhang, D., Tang, J., Li, J., and Wang, K. 2007. "A constraint-based probabilistic framework for name disambiguation." In Proc. sixteenth ACM conference on Conference on information and knowledge management (CIKM '07). ACM, New York, NY, USA, 1019-1022.