A Visual Approach for Name Disambiguation in Coauthorship Networks

Category: Research

ABSTRACT

Academic publication archives often draw from numerous, heterogeneous sources, whose records can follow differing naming conventions. As such, ambiguity concerning authorship of scientific papers such as authors sharing similar names, the use of first names versus initials, or alternate name spellings, has plagued research into scientific collaboration and influence. Detecting and correcting these errors is important for maintaining the archive, as well as for ensuring correctness and reliability in any desired subsequent analysis. We address two aspects of this problem: homonymy, when different authors have the same name; and synonymy, when the same author's name is represented in slightly different ways. We employ known methods of addressing homonymy and synonymy in the development of new visual tools for exploring and solving name ambiguity problems in a large compendium of scientific papers. Specifically, we rely on measures of author name similarity, paper abstract similarity, collaboration overlap, and betweenness centrality in the coauthorship network to identify cases of homonymy and/or synonymy, and present them visually for users to evaluate. Our system also provides merge, split, and undo operations for users to explore and better approximate the underlying real coauthorship network based on different settings of linear or nonlinear thresholds for each attribute.

1 INTRODUCTION

Ambiguity concerning authorship of scientific papers represents both a theoretical and practical problem. Analysis of scientific processes–collaboration networks, career trajectories, citation patterns, and the like–is plagued by errors of synonymy (where the same author's name occurs in slightly different ways) and homonymy (where two different authors share the same name). These issues also create practical problems for users of scientific archives, whose searches will either miss research of interest or turn up irrelevant papers. Many general measures of graph-level properties may be robust to uncorrected authorship information [15], although individual level statistics are likely to be more sensitive to such errors [12].

There are many existing approaches to the problem. They are generally either supervised (involving some manual curation of author names) or unsupervised (adopting sometimes highly complex techniques to cluster papers by author). Supervised name disambiguation is traditionally costly and time-consuming, so most researchers adopt unsupervised techniques, ranging from simple blocking, where only author names and initials are used to cluster papers, to complex algorithms involving techniques such as machine learning. Some research suggests that errors can be reduced dramatically using simple rules concerning names and initials [14]. Other techniques incorporate information about the relative frequency of names, coauthorship ties, citation patterns, source similarity, and abstract or manuscript similarity. But here we present a new, flexible hybrid approach which can be unsupervised but also permits visual supervision and manipulation of flexible, nonlinear thresholds for splitting or merging clusters

Simulation studies show that simple first initials-based algorithms accurately disambiguate author names in many cases [14], and while improved approaches [15] incorporated information about the relative frequency of a given last name into the disambiguation process, this approach did not significantly improve already high accuracy rates. Other approaches incorporate information from citation patterns, where higher rates of common citations increase the likelihood of assigning a paper to the same author [16]. While our data source does not include citation information, we take advantage of mutual coauthorship data, which alone is sufficient to achieve extremely high rates of accuracy [6], and also develop a measure of abstract similarity to make better informed decisions about when two papers are likely to be written by the same author and when the same author name is likely to represent more than one person.

In this paper, we present our hybrid visualization system which utilizes splitting and merging techniques to interactively modify the collaboration network in order to exploratively solve synonymy and homonymy problems. This system consists of a combination of several computation optimization, visualization, and graph techniques to help analysts to find a better methodology to solve name disambiguation problem interactively and answer important questions. The contributions of our work are:

- Interactive visual analytics approach with splitting, merging, and undo operations for name disambiguation.
- Nonlinear threshold control based on scatter plot selection and threshold lines approximation.
- An optimized algorithm for name and abstract similarities based on matrix subdivision by matching author last names.
- Statistics and DAG of split/merge history for better decision making.

2 RELATED WORKS

The challenge of name disambiguation has been investigated with data mining techniques in a number of ways (for a review and taxonomy, see [7]). Many of these rely on collaboration network measures [1, 16] or name similarity metrics [14]. While some name disambiguation approaches are supervised [9], most of these are unsupervised, using thresholds or clustering [10, 19]. However, unsupervised approaches offer little control or means of verification. Visual approaches are ideal for refining results and exerting more precise control over the analytic approach.

We employ techniques similar to those outlined by Torvik and colleagues [18] where we begin by blocking groups of papers based on author last name and first initial, such that papers are put into distinct author blocks if the last names differ or the first initials are inconsistent. We then compute similarity profiles for pairs of papers based on the similarity of their abstracts [13] and shared coauthors [11]. We also calculate betweenness centrality to help identify potential cases of homonymy, as is common among Asian scientists.

Many components of our work borrow from common visualization and visual analytic techniques and libraries. Most of the front-end visualization is built upon the d3.js framework [3], including its graph visualization and layout techniques [5]. We also utilize several graph analytic techniques that are common to many visualizations, such as betweenness centrality [8] and modularity clustering [4]. However, other than using visualization to simply visualize the resulting network [17], to our knowledge there are no visual analytic approaches to solving the name disambiguation problems.

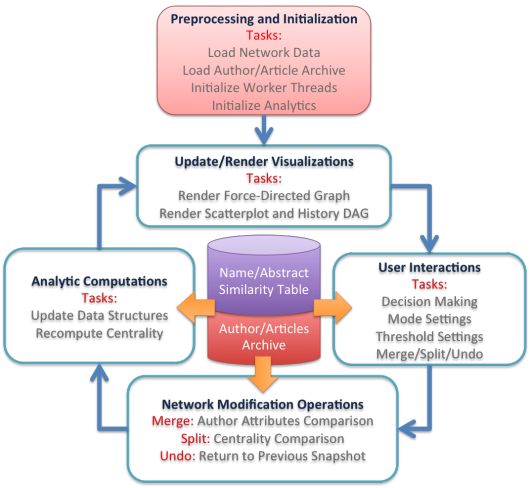


Figure 1: System's overall analytic process. Like most visual analytics, the general usage is cyclical.

3 Approach

The overall system process is divided to five steps. Except for the initial step, the latter four steps are run repeatedly during the whole name disambiguation analysis process. The five stages are:

Preprocessing and Initialization: The first step is to process and load the data. The data is collected and compiled, including generating an author list, an article list, a publication list, and an initial coauthorship network. Initial metrics are then pre-computed, such as the name similarity and abstract similarity tables. At run time, these are loaded and used to initialize the different views and worker threads in the system.

Update/Render Visualizations: The visual component of our system consists of three visualizations: an overview graph of the coauthorship network, a scatterplot of the analytic metrics, and a historical timeline of user operations. Like any graph visualization, a good layout is imperative for the legibility of the main graph. As such, the layout of the graph is continually being refined, in order to maintain a good visualization after splits or merges alter the network topology. The other views require less computation to render, and only change on user interaction.

User Interactions: This step accepts and responds to user interaction, allowing the user to set up and perform the different operations, as well as allowing the user to inspect the results with details on demand. This includes setting colors, threshold modes and values, and subsequent instantiations of the split, merge, or undo operations.

Network Modification Operations: After the user instantiates a split, merge, or undo operation, this step applies the user defined thresholds to the analytics to determine which nodes to change, then performs the nescessary modifications to the graph data structure and the visualization data structures.

Analytic Computations: As the network topology changes, several of the metrics that were initially precomputed need to be updated, such as centrality, name similarity, abstract similarity, and collaboration overlap. This step handles performing these updates, after which the visualizations can be updated.

4 Data Extraction and Analytics

Our approach uses several underlying analytics. The merge operation combines name similarity, abstract similarity, and collaboration overlap in the coauthorship network as measures to evaluate merge candidacy. The user can control the application of these metrics either with a simple weighted accumulation or by setting specific thresholds for each dimension. In the latter mode, a scatterplot interface is used to define linear or arbitrary nonlinear thresholds for two of the three attributes. The splitting operation primarily uses the betweenness centrality of the coauthorship network, as there is a strong correlation between high centrality and likelihood of the homonymy problem.

We first extracted authors and article abstracts from the arXiv repository [2] using the XML-like Open Archives Initiative (OAI) format, and built up an initial list of unique author names. We then compute lookup tables of name and abstract similarities between pairs of authors. For run-time computation optimization, we limit the size of these tables by only considering pairs with matching authors' last names, which subsequently reduces computational complexity dramatically when updating the network and its subsequent metrics upon merging or splitting.

4.1 Name Similarity

The name similarity of pairs of authors are computed using the Python string-matching package Fuzzywuzzy, which provides both full and partial string comparison functions. A full string similarity compares two strings with a measurement of standard edit distance. For the partial string comparison of two strings, it picks the shorter string and searches for the "best partial" through the longer string. That is to say, If the shorter string has the length m and the longer string has the length n, the best score would be computed by matching a substring of length m. As the author names often include initials, the partial string comparison algorithm suits our needs better, since it handles initials naturally.

We also handle first names that consist of only initials specially. As there is only one letter to check, there are few options to evaluate. To improve upon this, we also introduces a weighting based on the commonness of the names, weighing infrequent initials and names more strongly.

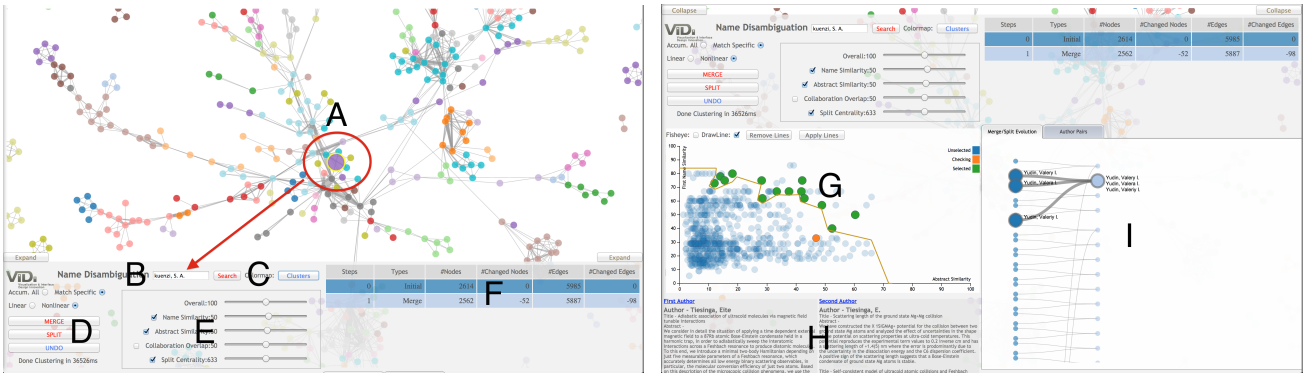
We also consider transpositions of names or initials. In some cases, the first and middle names or initials can be swapped, so when more than one name or initial is being considered, we compare them individually and use the maximum match.

4.2 Abstract Similarity

The abstract similarity of pairs of authors are computed by comparing keyword occurances in their abstracts via cosine similarity. The first step of this is to determine which words in the text are important keywords. We do this by evaluating the frequency of word occurances across the entire corpus of abstracts, as the more common a word is, the less useful it is as a keyword. We then construct vectors for each author of the more important words they use, ignoring the very common words. Finally, computing the cosine similarity is computed as a weighted dot product between the vectors of pairs of authors.

4.3 Collaboration Overlap

If two authors share the exact same neighborhood of collaborators in the coauthorship network, there is a much higher chance that they could be the same author than if they had no collaborators in common. So for a pair of nodes in the graph, we compare the sets of neighbors and use their overlap as a similarity metric. While the Jaccard index is commonly used for this task, we want to match pairs that could be of vastly different size neighbor sets. For instance, there could be many papers published under one name and only one published under another name, as could be the case with a spelling error in the data entry. So rather than using the union of the two sets, we set the denominator to be the minimum of the two



(a) Panels collapsed (b) Panels extended

Figure 2: The overall system interface consists of a graph view with two collabsable detail panels.

sets' sizes. That is, we evaluate the collaboration overlap C for two names x and y as:

$$C(N_x, N_y) = \frac{|N_x \cap N_y|}{min(|N_x|, |N_y|)}$$

where N_x and N_y are the associated neighborhood sets.

4.4 Betweenness Centrality

Betweenness centrality [8] is defined as the number of shortest paths between any pair of nodes that go through a particular node. We compute this for each node in the graph for use by the splitting operation. It is also recomputed as needed after each merging or splitting operations, as these operations affect the topology of the graph. The betweenness centrality of one node in a network is equal to the number of shortest paths from all nodes to all other nodes passing through it. A node with high betweenness centrality is usually considered to have higher influence in the network, since it is often a bridge node among clusters. In regards to the name disambiguation problem, such a bridging node may actually represent multiple authors with the same name (or at least same first initial and last name) belonging to different research groups. As a result, we employ this metric as a threshold for potential node splitting.

4.5 Modularity Clustering

Modularity [4] measures the quality of a partitioning of a network into clusters. A network with high modularity has dense connections between nodes within clusters and sparse inter-cluster connections. In our approach, we utilize modularity as part of the split operation; we replicate a splitting node into cluster of its local neighborhood, as defined by a greedy modularity clustering.

There are two main parameters for modularity score computation:

- e_{ii} - fraction of edges in cluster i

$$e_{ii} = \frac{|\{(u,v) : u \in V_i, v \in V_i, (u,v) \in E\}|}{|E|} \quad (1)$$

- a_i - fraction of edges in cluster i

$$a_i = \frac{|\{(u,v) : u \in V_i, (u,v) \in E\}|}{|E|} \quad (2)$$

- Modularity score - Q

$$Q = \sum_{i=1}^{n} (e_{ii} - a_i^2) \quad (3)$$

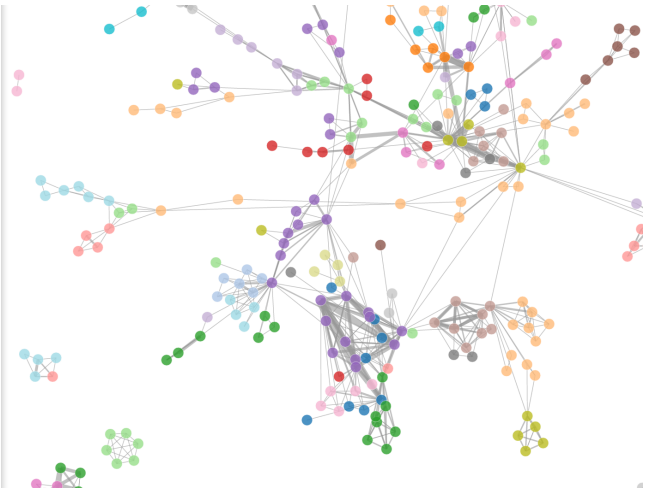
Where $|E|$ is the total number of edges and n is the number of clusters. Initially we start with each node having its own cluster. Then, we iteratively merge the pairs of clusters that would maximize the increase in modularity score, until we get to a point where merging would no longer help increase the modularity. There are some optimizations that we apply, such as only considering merges between clusters that have at least one edge between them.

5 Visual Representations and Interface

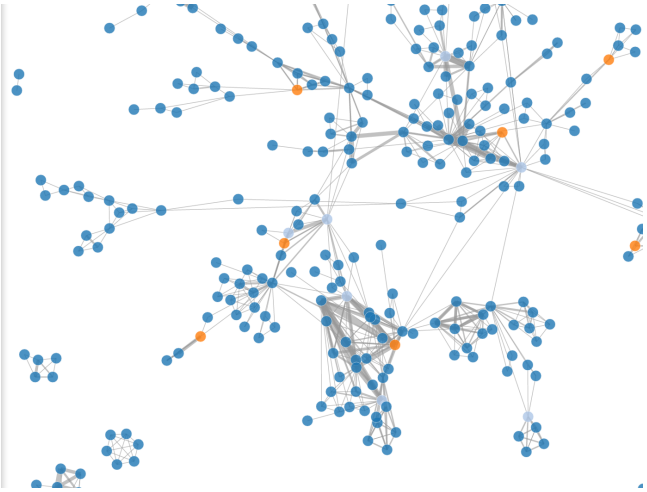
Our interface consists primarily of an overview of the network graph and two sliding windows in the bottom. Figure 2 shows the user interface with the left and right sliding windows both collapsed and extended. The user interface is divided to several parts. The primary display is the overview of the collaboration network (A). Specific authors can be searched for by name (B). The color mapping (C) can be set according to a clustering or according to which merging or splitting operation created the node. Widgets (D) control the mode settings, merging / splitting / undo operations, and execution messages. In linear mode, more widgets (E) control the threshold settings. A statistics table (F) shows the basic statistics data of each operation that has been run. A scatter plot (G) shows the merge/split similarity metrics directly and provides an interface for drawing the nonlinear threshold. Detailed author information (H) allows for side-by-side comparison of author pairs from the scatterplot. A merging and splitting evolution directed acyclic graph (DAG) (I) shows the overall history and effect of each operation.

Our system starts with an overview of the initial collaboration network graph, laid out with a force-directed algorithm. Each node is a unique author name, and the ties (edges) are weighted based on the number of collaborations between the associated pairs of authors. Our system provides a methodology to solve name disambiguation problem interactively and iteratively. As the disambiguation process proceeds to merge and split nodes, this graph updates dynamically, with color highlighting the affected nodes at each step.

The system also uses a directed acyclic graph (DAG) representation to show the history of merging and splitting operations. Nodes in the DAG correspond to nodes in collaboration network graph, and are colored according to the split or merge step. Nodes in the DAG can be selected and found in the overview graph. Paths of merging and splitting in the DAG can be highlighted to better help analysts aware of the accuracy of name disambiguation process.



(a) Clustering



(b) Merge/Split Step

Figure 3: The main graph view provides a summary of the current state of the overall collaboration network. It can be colored either by a modularity clustering or according to what stage in the disambiguation process the node was created.

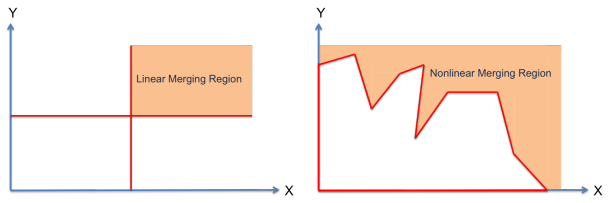


Figure 4: Traditional, linear thresholds would define a box (left). Our system allows drawing of an arbitrary threshold curve (right)

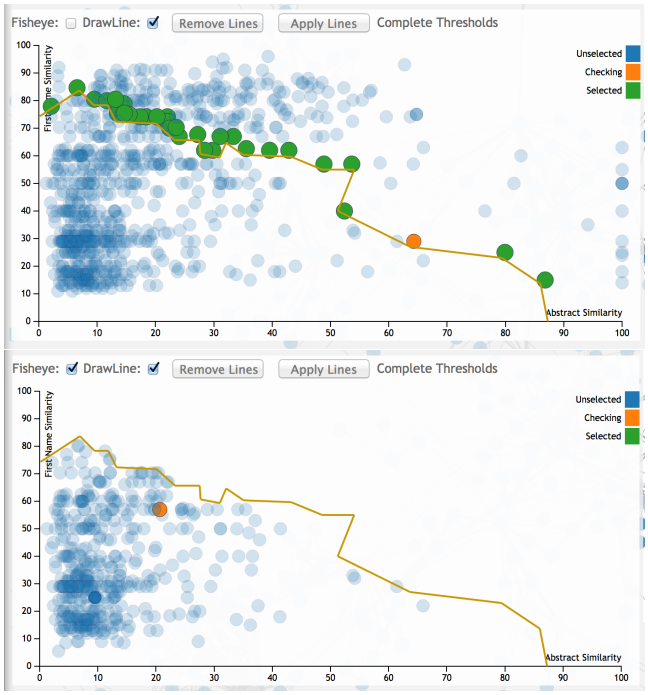


Figure 5: The scatterplot view plots pairs of authors according to the metrics of likelihood of them being the same person. The user can inspect them and mark some as mnemonics, then draw a threshold curve (top). Points above this curve correspond with pairs of authors that get merged, and hence get removed from the plot (bottom).

5.1 Graph Overview

For the overview graph, we apply a force-directed layout to position the nodes of a graph aesthetically and smoothly adapt to changes in the graph. We utilize the data-driven javascript library d3.js [3] for laying out and rendering the network efficiently.

When merging, splitting, or undo operations are executed, this view updates with animation to show the evolution of changes. The larger node in the middle of the figure is highlighted and centered in response to an author name search. Author nodes in the overview network graph can also be selected for detailed inspection from the scatterplot or history DAG.

5.2 Scatterplot Interface

The scatterplot provides an interface both for setting complex, non-linear thresholds on the similarity metrics and for inspecting borderline cases directly. The x-axis and y-axis correspond to two user selected similarities; we generally use the abstract similarity and first name similarity, respectively. The user can mark specific nodes for merging then draw a threshold line by hand to encapsulate them.

As nodes in this plot are pairs of nodes in other views, they do not follow the same color map. The orange point is the current user selection, which is listed in detail in the two columns below the scatterplot. Green nodes are those that are either over the threshold or specificly selected by the user to be merged. Blue points are the remainder that are neither currently selected for merging nor under inspection by the user.

The non-linear name/abstract similarity threshold is set by drawing a series of line segments in the scatterplot between the two axes. Points outside of this line (above and to the right) meet the threshold to merge, while those closer to the origin do not. Each node is checked by a point-in-polygon algorithm (Algorithm 1) to check if it is inside a polygon enclosed by the drawn line, x-axis, and y-axis. This algorithm checks the number of intersections of a random line passing through the node and the polygon. If the number is odd, the node is inside the polygon, and vice versa.

Figure 5 shows the result of merging nodes according to the threshold line. Several nodes are merged and the new nodes are inserted into the plot based on the name/abstract similarity of the merged author lists. Note that new created nodes may still lie above the threshold line, and as such it can make sense to re-apply

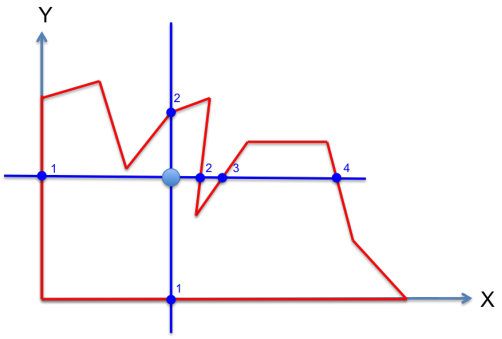


Figure 6: Point-in-Polygon algorithm. Points on an 'odd' segment of a random line through the polygon are inside (and vice-versa).

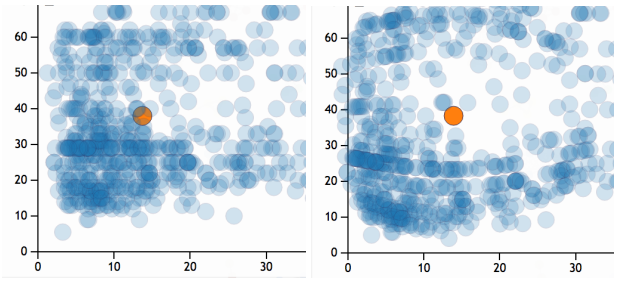


Figure 7: Without fisheye effect (left) and with fisheye effect (right). The orange node is the node analyst has currently selected. The fisheye effect makes it easier to locate the position of the node and explore its surroundings, which helps in drawing the non-linear threshold line.

merge operations with the same settings a number of times before the names are disambiguated.

A fisheye technique is optionally applied to zoom in on dense regions of points in the scatterplot. Nodes are repelled by the strength inversely proportional to the distance to the cursor. We found this technique to be particularly helpful for exploring the points in dense clusters. An example is shown in Figure 7.

input : Array of lines and position of one node (x, y)
output: Boolean b if point-in-polygon

foreach *line l of the line array* **do**
  if $(y < l(s).y) \neq (y < l(t).y)$ **then**
    if $x < ((l(t).x - l(s).x) \times (y - l(s).y)/(l(t).y - l(s).y) + l(s).x)$ **then**
      $xInRegion \leftarrow \neg xInRegion$
    end
  end
  if $(x < l(s).x) \neq (x < l(t).x)$ **then**
    if $y < ((l(t).y - l(s).y) \times (x - l(s).x)/(l(t).x - l(s).x) + l(s).y)$ **then**
      $yInRegion \leftarrow \neg yInRegion$
    end
  end
  return $xInRegion \cap yInRegion$
end

Algorithm 1: Point-in-Polygon algorithm

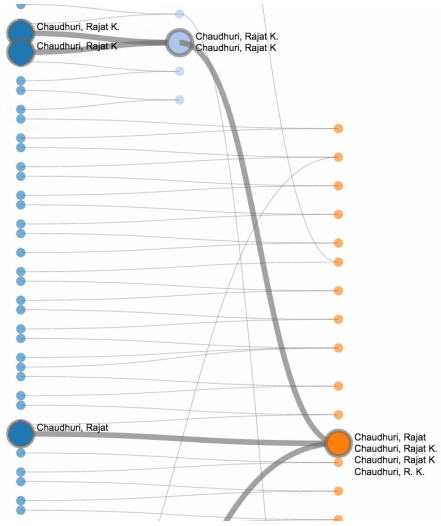


Figure 8: Merging/splitting history is represented with a directed acyclic graph (DAG). Each operation is a column of the graph, and the user select nodes to track their evolution backwards or forwards through the history, and highlight their current position in the network.

Steps	Types	#Nodes	#Changed Nodes	#Edges	#Changed Edges
0	Initial	2614	0	5985	0
1	Merge	2586	-28	5933	-52
2	Merge	2530	-56	5852	-81

Figure 9: The statistics table indicates how many nodes and edges were added (positive number, due to splits) or removed (negative number, due to merges) by each operation, and also serves as a color legend.

6 Network Modification Operations

Our system is based on allowing the user to inspect and control three main operations: Splitting author nodes, merging author nodes, and undoing operations.

6.1 Splitting

The split operation duplicates nodes based on the betweenness centrality of nodes in the dynamic network, since abnormally high centrality implies a high probability that a specific node is actually several authors. In our current implementation, we only consider nodes presenting only one author that never merged, as merged nodes that would subsequently need to split is more indicative of an overly aggressive merge, and would be more easily handled with an undo operation. As such, it is generally better for the user to perform split operations before merges. Once we find nodes that have higher centrality than a user defined threshold we have to determine how many times to replicate the node and how to connect the edges to the new nodes based on the number of clusters that remain in its neighborhood after removing the node.

Specifically, the we split a target node by the following steps:

- Determine how many clusters its neighbors belong to
- Replicate the node for each of these cluster
- Rewire the edges to these newly created nodes.
- Add new nodes into history directed acyclic graph.

6.2 Merging

To evaluate whether or not to merge two author nodes, we consider their name similarity, abstract similarity, and collaboration overlap. For efficiency, we first use the similarity of the last names as an initial filter. The first name similarity and abstract similarity matrices are precomputed during the preprocessing step. But, since the coauthorship network is dynamically changing, any single node may represent multiple authors. In such cases, we take the mean of scores computed based on the name similarity and abstract similarity of all pairs between nodes' lists of authors. We combine these three metrics with user defined thresholds. However, there are different modes of threshold settings in our system:

- Accumulate all three metrics weighted by user defined values, and merge if this sum exceeds a user defined threshold.
- Independently threshold each similarity dimension, as shown in 4.
- Nonlinearly threshold two similaritys by a user defined curve in a 2D scatter plot 4.

Once these thresholds have been calculated, we have a list of pairs to merge, which we distill into a list of to-be-removed nodes and a list of new corresponding replacement nodes. We then merge these nodes according to the following process:

- Take a 'snapshot' of current network data and history DAG data for possible undo operation.
- Rewire the edges connecting to the to-be-removed nodes to corresponding replacing nodes.
- Merge duplicate edges between pairs of authors. The number of collaborations (weight) of the old edges are added together in the new edge.
- Set the author list of the new nodes by concatenating the author lists of all old nodes.
- Remove old nodes and update node indices of the node list.
- Add modified nodes into history directed acyclic graph.

6.3 Undo

The visualization tool also provides an Undo feature to return to previous step. A snapshot of current network structure is saved right before each operation is run. If Undo executed, the system reverts to the saved copy, including all modified or derived metrics.

7 Case Studies

To demonstrate the utility of our approach, we created case studies by applying it to real-world data. Cornell University Library's arXiv.org [2] provides a readily accessible repository of nearly a million articles, with little enforcement of consistency of authorship. We have collected and parsed their entire collection of physics articles. However, for clarity of presentation, the results shown here are focused on the articles in the atomic physics subdomain, published in the years 2001 to 2005.

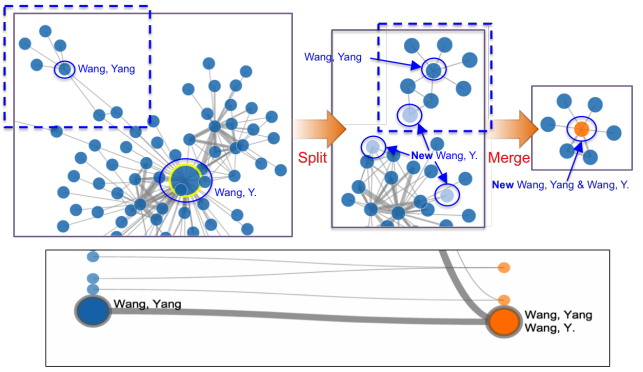


Figure 10: Case Study: "Y. Wang" is quite ambiguous. A split operation divides it into several communities, then a merge operation combines them with more complete author names, such as "Yang Wang".

7.1 Correcting an Ambiguous Author Name

The author name "Y. Wang" is fairly ambiguous: its first name is an initial, and it has a quite common last name. The initial graph view in the upper left of Figure 10 shows that it belongs to more than one community. Applying a split operation divides the node up into its different communities, and we see that one community in particular is fairly isolated, and also has a "Yang Wang" in it. So, we subsequently apply a merge operation and combine the two into one node.

7.2 Interactive Non-linear Threshold Drawing

Most traditional approaches apply some sort of linear threshold. However, it becomes apparent just by looking at a scatterplot of the metrics (as in Figure 11), that there is no clear seperation in which to draw a straight line. While the top-right points are pairs of author nodes that are almost surely correct to merge, and the bottom-left points are pairs with almost no match, the boundary between the two groups is not clear, as it is a fairly dense point cloud. In order to find the boundary, the user explores the points near the middle of the plot, interactively evaluating which points are correct or not and marking the correct merges green. By doing this, the user can more precisely locate this boundary between correct and incorrect merges and draw the appropriate threshold curve. An example of this process is depicted in Figure 11. A number of nodes have been explored along the middle of the plot; some of them can be confirmed as correct pairs to merge and marked green (at top) while others are confirmed as being two seperate people (at bottom). It can be clearly seen that no simple linear threshold would correctly separate these two groups. So the user can interactively draw a complex threshold curve to include just the green nodes, resulting in a more correct solution.

7.3 Verification and Correction

During a split or merge operation, it is possible for erroneous changes to be incurred. Inspection of the results can find such cases, and the interactivity of our system allows the user to go back and correct the mistake. Figure 12 shows an example of this. At left are some nodes that are the result of an overly aggressive merge, where names that should have remained separate got merged together. After undoing the merge operation and searching for the names, we find the associated points in the scatterplot and note that the threshold is currently defined to merge them. Next, we correct the threshold curve to exclude them, and reapply the merge. Lastly, the results can be verified as being fixed in the history view.

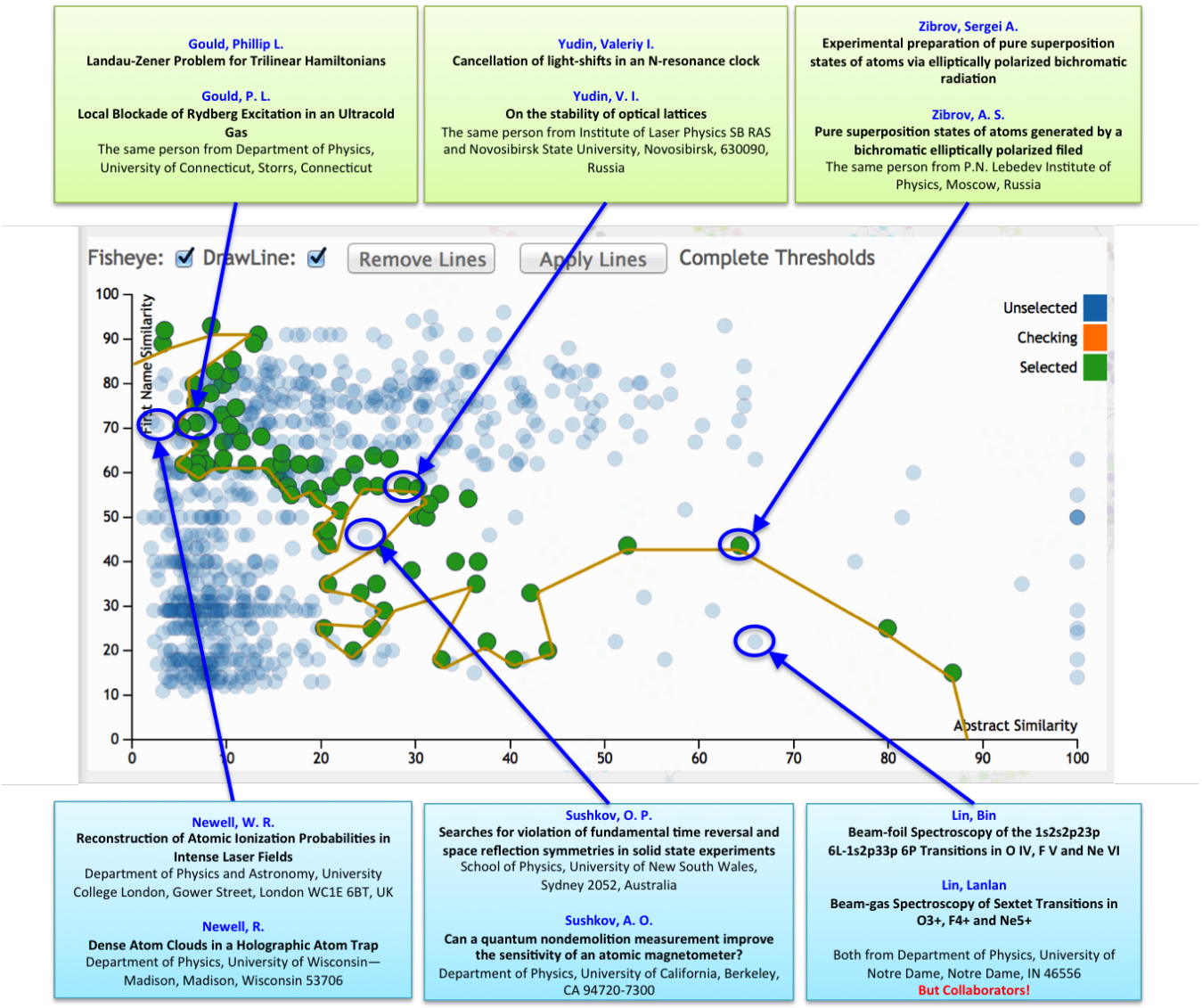


Figure 11: Case study: Exploring the points near the middle of the plot lets the user interactively craft an appropriate threshold curve between correct and incorrect potential merges. Many explored points near the boundary were marked green, as they indicate author pairs that should be merged. Details of several of these are shown in the row above the plot. However, the explored points near the curve that were left blue correspond to pairs of authors with similar metric values but which should not be merged. Details of some of these are shown below. The curve is then drawn accordingly.

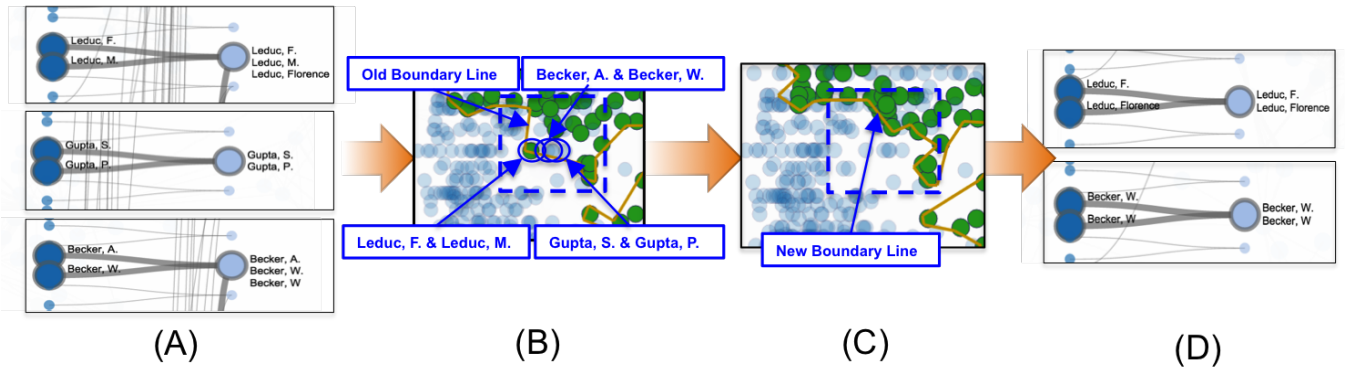


Figure 12: Case study: If an operation mistakenly merges distinct authors (a), the user can undo the operation (b), adjust the threshold curve (c), and reapply the merge (d) to generate the correct results.

8 Future Work

While our system has been found to be effective, there are a number of ways it can be extended. Other metrics for splitting besides or combined with betweeness could be considered, and controlled with the same scatterplot-based approach. Partial undos could be helpful in optimizing the error correction method. Specific special case controls could be helpful in cases where the user finds or has a priori knowledge of individual fringe cases that are clear outliers. And optimizations, such as incremental modularity calculations or GPU accelerations could be implemented to scale the system up to larger networks. One extension of our work would be to use it to define settings and thresholds on a smaller subset of the network data interactively, and then applying them offline to a larger academic network. Our approach is also not limited to co-authorship networks, but would be applicable to any other networks with potentially labels, such as key term coincidence in large document corpora.

9 Conclusion

We have designed and implemented a visual analytic system for solving name ambiguity problems by combining several distinct name disambiguation metrics with an interactive visualization framework. The exploration and inspection techniques that we employ enable the user to combine and control the metrics in new ways that greatly help in solving fringe cases, reducing the number of false positives and negatives that would occur in simple applications or combinations of these or possibly even other more complex metrics. By enabling the user to better disambiguate such a co-authorship network, our system will enable more accurate research into its graph-theoretical and sociological properties.

References

[1] D. R. Amancio, O. N. Oliveira, and L. d. F. Costa. On the use of topological features and hierarchical characterization for disambiguating names in collaborative networks. *EPL*, 99(arXiv:1302.4504):2. mult. p, Feb 2013.

[2] The arXiv archive. `http://arxiv.org/`.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[4] A. Clauset, M. E. J. Newman, , and C. Moore. Finding community structure in very large networks. *Physical Review E*, pages 1– 6, 2004.

[5] T. Dwyer. Scalable, versatile and simple constrained graph layout. In *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'09, pages 991–1006, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[6] X. Fan, J. Wang, X. Pu, L. Zhou, and B. Lv. On graph-based name disambiguation. *J. Data and Information Quality*, 2(2):10:1–10:23, Feb. 2011.

[7] A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. A brief survey of automatic methods for author name disambiguation. *SIGMOD Rec.*, 41(2):15–26, Aug. 2012.

[8] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, page 215, 1978.

[9] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In H. Chen, H. D. Wactlar, C. chih Chen, E.-P. Lim, and M. G. Christel, editors, *JCDL*, pages 296–305. ACM, 2004.

[10] H. Han, H. Zha, and C. L. Giles. A model-based k-means algorithm for name disambiguation. In *INTERNATIONAL SEMANTIC WEB CONFERENCE*, 2003.

[11] I.-S. Kang, S.-H. Na, S. Lee, H. Jung, P. Kim, W.-K. Sung, and J.-H. Lee. On co-authorship for author disambiguation. *Inf. Process. Manage.*, 45(1):84–97, Jan. 2009.

[12] D. F. Klosik, S. Bornholdt, and M.-T. Hütt. Motif-based success scores in coauthorship networks are highly sensitive to author name disambiguation. *Phys. Rev. E*, 90:032811, Sep 2014.

[13] G. S. Mahalakshmi, S. D. Sam, and S. Sendhilkumar. Mining research abstracts for exploration of research communities. In *Proceedings of the 5th ACM COMPUTE Conference: Intelligent & Scalable System Technologies*, COMPUTE '12, pages 6:1–6:10, New York, NY, USA, 2012. ACM.

[14] S. Milojevic. Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics*, 7(4):767–773, 2013.

[15] J. Moody. The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. *American Sociological Review*, 69(2):213–238, 2004.

[16] C. Schulz, A. Mazloumian, A. M. Petersen, O. Penner, and D. Helbing. Exploiting citation networks for large-scale author name disambiguation. arXiv preprint arXiv:1401.6157, 2014.

[17] A. Strotmann, D. Zhao, and T. Bubela. Author name disambiguation for collaboration network analysis and visualization. *Proceedings of the American Society for Information Science and Technology*, 46(1):1–20, 2009.

[18] V. I. Torvik, M. Weeber, D. R. Swanson, and N. R. Smalheiser. A probabilistic similarity metric for medline records: A model for author name disambiguation. *Journal of the American Society for Information Science and Technology*, 56:40–158, 2005.

[19] F. Wang, J. Li, J. Tang, J. Zhang, and K. Wang. Name disambiguation using atomic clusters. In *Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, WAIM '08, pages 357–364, Washington, DC, USA, 2008. IEEE Computer Society.