A Relevance Feedback Approach for Social Network Clustering in the Context of Triangle Inequality Violations

Von der Fakultät 1 - MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik der Brandenburgischen Technischen Universität Cottbus–Senftenberg genehmigte Dissertation zur Erlangung des akademischen Grades eines Dr.-Ing.

vorgelegt von

Sanjit Kumar Saha

geboren am 30.12.1986 in Gopalganj

Vorsitzender: Prof. Dr.-Ing. Andriy Panchenko

Gutachter: Prof. Dr.-Ing. habil. Ingo Schmitt

Gutachterin: Prof. Dr.-Ing. habil. Meike Klettke

Tag der mündlichen Prüfung: 27.09.2023

DOI: https://doi.org/10.26127/BTUOpen-6519

Acknowledgements

I want to start by expressing my gratitude to Professor Ingo Schmitt for being my supervisor. He gave me the opportunity to complete this dissertation. He gave my work a lot of feedback and ideas. He frequently emphasized new ideas and suggestions for further research. It really helped me advance my work. He always had confidence in my abilities. He always made time for me, despite his busy schedule, when I needed direction or inspiration. He allowed me the freedom and support to pursue my ideas. His insights and recommendations always added something worthwhile to my work and helped me with a variety of issues. In addition, he taught me how to write short papers, which has turned out to be one of my most useful skills.

I am immensely grateful to Professor Meike Klettke for her valuable and diverse feedback on my work. To complete the research, the Graduate Research School is the key stakeholder. I convey my sincere thanks for the financial and administrative support of Mr. Robert Rode from the Graduate Research School. He has assisted me in my educational expenses.

Undoubtedly, a supportive environment was necessary for the success of this thesis. Therefore, I would like to thank my pleasant office mates, Alexander Stahl and Rrezart Murtezaj. I am also much obliged to the staff, particularly Ekkehard Schwaar, for his continuous support. I would like to thank Sandy Schneider for her assistance in sorting through various official documents regarding scholarships and conference registration.

Last but certainly not least, I want to express how grateful I am to my family, especially my mother, Laxmi Rani Saha and mother-in-law, Bakul Rani Banik, for their endless love, care, and support throughout my life. My mother has always believed in me and has helped me in any way she can. I also want to thank my lovely wife, Tapashi Gosswami, for being so patient with me and for always making me smile, especially when I was having bad moods toward the end of this thesis. I am grateful to my son, Soptok Gosswami Saha. He was born during my PhD studies, and I did not give him enough time as a father because of my engagement in thesis writing. I am grateful to my elder brother, Ranajit Shaha, and my brother-in-law, Bishwajit Gosswami, for always inspiring me. Finally, I am eternally grateful to Almighty God for the blessings he has bestowed upon me with all his grace.

Abstract

People in a social network are connected, and their homogeneity is reflected by the similarity of their attributes. For effective clustering in such a network, the similarities among people within a cluster must be much higher than the similarities between different clusters. The higher the similarity within a cluster and the greater the difference between clusters, the more effective the clustering results will be. Traditional clustering algorithms like hierarchical (agglomerative) clustering or k-medoids take distances between objects as input and find clusters of objects. The distance functions used should comply with the triangle inequality (TI) property, but sometimes this property may be violated, thus negatively impacting the quality of the generated clusters.

However, in certain scenarios, such as social networks, it is still possible to achieve meaningful clustering even if the TI does not hold. One possibility to find meaningful clusters are quantum-logic-inspired query languages such as the commuting quantum query language (CQQL). It is the base for the clique-guided non-TI clustering approach, which is used in this thesis. The CQQL is particularly noteworthy as it allows the formulation of logic-based queries that incorporate both Boolean and similarity conditions. Therefore, it can be used to derive the similarity value between two objects.

Furthermore, attributes in the network may not have equal impact on similarity and affect the resulting clusters, impacting user satisfaction. CQQL incorporates weights to express the varying importance of sub-conditions in a query while preserving consistency with Boolean algebra. This enables personalization of results through relevance feedback (RF), which enhances user interaction with the system. The approach incorporates user feedback to enhance the quality of clusters, finds clusters relevant to user needs, and also provides alternative possible feedback to the user.

The main challenge of comparing clusterings is that there is no ground truth data associated with this approach to compare to. In such situations, the relevance of any clustering can be measured based on human judgment. Human-generated gold standard clustering, which is considered to be "correct" in some sense, can be used in this scenario. However, the question is how to compare the performance of common clustering approaches to that of human-generated gold standard clustering. A noteworthy technique for comparing clusterings

involves counting the pairs of objects that are grouped identically in both clusterings. By doing so, a clustering distance is calculated that measures the dissimilarity between the two clusterings.

To validate the utility of the non-TI clustering approach, experiments are conducted on social networks of different sizes. Three central questions are addressed by the experiments mentioned: first, is it possible to detect meaningful clusters even though TI violates; second, how does a user interact with the system to provide feedback based on their needs; and third, how fast do the detected clusters based on the proposed approach converge to the ideal solution during the relevance feedback process?

To sum up, the experiments' objective is to demonstrate the validity of a theoretical approach. The research findings presented here provide sufficient evidence for detecting meaningful clusters based on user interaction. Furthermore, the experiments clearly demonstrate that the non-TI clustering approach can be used as an RF technique in clustering.

Zusammenfassung

Menschen in einem sozialen Netzwerk sind miteinander verbunden. Ihre Homogenität spiegelt sich in der Ähnlichkeit ihrer Eigenschaften wider. Für eine effektive Clusterbildung in einem solchen Netzwerk müssen die Ähnlichkeiten zwischen Personen innerhalb eines Clusters größer sein als die Ähnlichkeiten zwischen Personen verschiedener Cluster. Je größer die Ähnlichkeit innerhalb eines Clusters und je größer der Unterschied zwischen den Clustern ist, desto effektiver sind die Clustering-Ergebnisse. Herkömmliche Clustering-Algorithmen wie etwa hierarchisches (agglomeratives) Clustering oder k-Medoids nutzen Abstände zwischen Objekten zum Finden von Clustern von Objekten. Die verwendete Abstandsfunktion sollte die Eigenschaft der Dreiecksungleichheit (TI) aufweisen, aber manchmal kann diese Eigenschaft verletzt sein, was sich negativ auf die Qualität der erzeugten Cluster auswirken kann.

In bestimmten Szenarien, z.B. in sozialen Netzwerken, ist es jedoch möglich, eine sinnvolle Clusterbildung zu erzielen, selbst wenn die TI nicht erfüllt ist. Eine Möglichkeit, sinnvolle Cluster zu finden, sind von der Quantenlogik inspirierte Anfragesprachen wie die Commuting Quantum Query Language (CQQL). Sie ist die Grundlage für den cliquengesteuerten Non-TI-Clustering-Ansatz, der in dieser Arbeit verwendet wird. Die Sprache CQQL ist besonders bemerkenswert, da sie die Formulierung von logikbasierten Abfragen erlaubt, die sowohl Boole'sche als auch Ähnlichkeitsbedingungen enthalten. Daher kann sie verwendet werden, um einen Ähnlichkeitswert zwischen zwei Objekten ermitteln.

Darüber hinaus haben die Attribute im Netzwerk möglicherweise nicht den gleichen Einfluss auf die Ähnlichkeit und beeinflussen die resultierenden Cluster, was sich auf die Benutzerzufriedenheit auswirkt. CQQL bietet ein Gewichtungsschema, um die unterschiedliche Wichtigkeit von Unterbedingungen in einer Abfrage auszudrücken, wobei jedoch die Konsistenz mit der Boole'schen Algebra gewahrt bleibt. Dies ermöglicht eine Personalisierung der Ergebnisse durch Relevanz-Feedback (RF), was die Interaktion der Benutzer mit dem System erweitert. Der Ansatz bezieht das Feedback der Benutzer ein, um die Qualität der Cluster zu verbessern, findet also Cluster, die für die Bedürfnisse der Benutzer relevant sind, und bietet dem Benutzer alternative Rückmeldungen.

Die größte Herausforderung beim Vergleich von Clustern besteht darin, dass es bei diesem Ansatz keine anzustrebende Cluster als ground truth gibt. In solchen Situationen kann die Relevanz eines jeden Clusters auf der Grundlage eines menschlichen Urteils gemessen werden. Ein vom Menschen erstelltes Goldstandard-Clustering, das in gewissem Sinne als "richtig" angesehen wird, kann in diesem Szenario verwendet werden. Es stellt sich jedoch die Frage, wie die Leistung gängiger Clustering-Ansätze mit der eines vom Menschen erstellten Goldstandard-Clustering verglichen werden kann. Eine Technik zum Vergleich von Cluster-Ergebnissen besteht darin, die Paare von Objekten zu zählen, die in beiden Cluster-Ergebnissen gleich gruppiert sind. Auf diese Weise wird ein Clustering-Abstand berechnet, der die Unähnlichkeit zwischen den zwei Cluster-Ergebnissen misst.

Um den Nutzen des Non-TI-Clustering-Ansatzes zu überprüfen, werden Experimente mit sozialen Netzwerken unterschiedlicher Größe durchgeführt. Drei zentrale Fragen werden in den Experimenten behandelt: Erstens, ist es möglich, sinnvolle Cluster zu erkennen, auch wenn die TI verletzt wird; zweitens, wie interagiert ein Benutzer mit dem System, um Feedback auf der Grundlage seiner Bedürfnisse zu geben; und drittens, wie schnell konvergieren die Cluster auf der Grundlage des vorgeschlagenen Ansatzes während des Relevanz-Feedback-Prozesses zur idealen Lösung?

Zusammenfassend lässt sich sagen, dass das Ziel der Experimente darin besteht, die Gültigkeit eines theoretischen Ansatzes zu demonstrieren. Die hier vorgestellten Forschungsergebnisse liefern ausreichenden Nachweis für die Erkennung von sinnvollen Clustern auf der Grundlage von Benutzerinteraktionen. Darüber hinaus zeigen die Experimente deutlich, dass der Non-TI-Clustering-Ansatz als RF-Technik für das Clustering verwendet werden kann.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

A group of individuals can be used to represent a person's, group's, or community's social context. Some of them communicate with one another, while others don't. A social network is a pattern made up of all these communication ties. A network like this shows who communicates with whom. These communication ties form a web of direct and indirect connections between persons when taken together. The dyadic link of interpersonal communication between two people is referred to as the communication tie. The structure of a social network can be visualized as a graph composed of individuals or organizations (known as social actors) and the connections between them, represented by nodes and edges respectively.

After services like Facebook, Twitter, and Instagram emerged and became a part of our daily lives, the concept of a social digital network became widespread. Entities and relationships among these entities participating in the network are the two main aspects of social networks. Entities may be "people," and binary relationships may be these people's "friendship," as on Facebook and most other social media platforms, but they are not restricted to "people" and "friendship". Entities can be anything, such as organizations or websites, and relationships can be anything, such as business, trade, or collaboration.

The structure of a network is a reflection of its distinct information flow patterns. In a social networking service like Facebook, users feed their account with personal information, making these networks a rich data source. While data is valuable in and of itself, its full potential can only be realized by learning from it. The ability of humans to analyze data and extract information is impressive, but they simply cannot match the enormous amount of data that is available.

The ordinary human cannot even comprehend datasets of several kilobytes, such as a table with multiple rows and columns, while research and industry begin to prepare for Exascale computing. To address this issue, research areas such as machine learning, data mining, and knowledge discovery have emerged, each of which provides methods for processing large

volumes of data that cannot be evaluated manually. The methods and algorithms created in this field are intended to automate, replicate, and supplement various forms of human reasoning so that usable knowledge can be extracted from data.

One of the fundamental methods of gaining understanding and learning is by grouping data into meaningful categories. When faced with a complicated and unknown situation involving multiple entities, people tend to examine it and search for similarities in their appearance or behavior, and then categorize the entities based on these similarities. This idea of similarity indicating a semantic or functional relationship allows for new knowledge to be derived, even for complex problems. To help with this, the process of *clustering* is utilized in data analysis, which involves dividing a set of entities into separate groups called *clusters*. The members within each cluster are similar to one another and distinct from those in other clusters.

Even though the problem appears to be well-defined, solving it is incredibly difficult for two reasons. The first is the subjective and changing nature of similarity. While there are numerous functions for determining this attribute, each is based on a different concept of resemblance. As a result, the degree of similarity between objects is determined by the measuring function used. Furthermore, obtaining a number for object similarity is insufficient to determine whether or not they are similar enough to be classified together. A threshold must be established for this decision. Assessing similarity is a complex and subjective task that can be affected by various factors like application domain, experience, and personal bias. Developing a standardized framework for making decisions in an algorithm is challenging due to this subjectivity.

Clustering also faces the challenge of working with unknown data. The objective of clustering is to extract new and meaningful insights from the data. Although the discovered cluster structure provides new information, there is no guarantee that it will be useful. Since clustering aims to uncover new knowledge, there is no established reference point to validate its results. As a result, clustering outcomes should be considered as tentative findings that need to be further tested within the context of the application domain from which the data was collected.

A plethora of clustering methods have been proposed over the years. The first, and certainly most widely used, method for locating clusters in networks is to look for strongly coupled nodes in comparison to other nodes. The way this idea is converted into operational implementations varies due to the generality of the notion.

Several formulations rely on detection of actors or edges with high centrality, as for instance, the very popular method of Newman and Girvan [2004], a divisive algorithm for undirected and unweighted graphs based on edge-betweenness, afterwards generalized by

Chen and Yuan [2006]. Further methods relying on a similar ground build on the optimisation of the cluster modularity (Danon et al. 2005), so that each community will include a larger number of inner edges than expected by chance. The Louvain method is unarguably one of the most popular representative of this category (Blondel et al. 2008). The aforementioned methods result in cohesive communities where transitivity is high and each actor is highly connected to each other inside the group. Notwithstanding, the idea of high density within a group may be also intended as the one arising in star-shaped clusters, where density is concentrated in the figure of some hubs attracting less prominent actors. Evidence of such a theoretical mechanism of aggregation has been explained by Goyal et al. [2006] as a combination of small-world behavior guided by the presence of interlinked stars. In fact, this principle has been largely neglected by SNA, with the works of Kloster and Gleich [2014], based on the local optimization of the so-called conductance and, to some extent, Falkowski et al. [2007] representing an exception. A further facet of the clustering problem in networks, known as cut-based perspective, aims at partitioning networks in a fixed number of balanced groups with a small number of edges between them, and no guarantees about a possible denser structure of inner connection. In this context, networks are often of a mesh- or grid-like form. Methods in this class refer back to the seminal work of Kernighan and Lin [1970] and often build on the spectrum of of the data.

All of these methods have one thing in common: they divide a network into distinct clusters, with each node belonging to just one of them. While this paradigm is well-understood, it is incapable of simulating real-world settings in which nodes may belong to many clusters. For example in a social network, a person might be part of a group of friends at work, a group of friends from school, a group of friends at a sport club and also a close family circle [Ahn et al. 2010; Leskovec and Mcauley 2012]. As a result, many overlapping community models and algorithms have been presented. Because it is more difficult to come up with excellent global quality indicators that can be optimized efficiently for overlapping communities, there is a much larger focus on the quality of individual communities [Xie et al. 2013; Amelio and Pizzuti 2014; Fortunato and Hric 2016]. To avoid confusion, we regard a community as a cluster.

The idea behind clustering is to group objects together based on their similarities, such that the similarities within each cluster are greater than the similarities between different clusters. To measure the similarity between objects, most clustering techniques use distance functions. These functions must satisfy certain properties, including the triangle inequality (TI), the $d(x_1,x_3) \leq d(x_1,x_2) + d(x_2,x_3)$ property where x_1, x_2, and x_3 are objects, and the $d(x_i,x_j) \in [0,\infty]$ distance function that returns the distance value of an object pair $\{x_i,x_j\}$. However, sometimes the TI property is not satisfied, and these distance functions are called

non-TI functions. The violation of TI can negatively impact the quality of the clustering results.

Some researchers proposed *clique* based method that can solve this issue.

Definition 1.1 (Clique). For an undirected graph $G = (V,E)$ of a set of N nodes $V = \{v_1, v_2, v_3, ..., v_N\}$ and edges $E = (v_i, v_j)$, a *clique* C is a subset of nodes $C \subseteq V(G)$ such that any two distinct nodes $v_i, v_j \in C$ are connected by an edge. A *maximal clique* is a clique that cannot be enlarged by adding an additional neighboring node.

A clique a closely connected group of individuals which follow a unique pattern of communication. One of the important *maximal cliques* extraction methods, is the Bron and Kerbosch [1973]. But it is ineffective when talking about a graph containing dozens of non-maximal cliques, since for each maximal clique, it makes a recursive call. For this reason, Bron and Kerbosch also described another variant of this algorithm involving a new heuristic called "Pivot" [Himmel et al. 2017]. The purpose of this heuristic is to limit the number of recursive calls made to save computation time [Tomita et al. 2006]. However, this version did not get any advantage compared to the basic version since its computation time was extended because of its degeneration, which requires additional time. For this reason, Eppstein et al. [2013] have proposed an alternative method, which extracts all the maximal cliques. This algorithm performs the external level of recursion by Bron–Kerbosch Algorithm, using a degeneracy order to order the sequence of recursive calls. Then, it performs the internal levels of recursion based on the pivot rule of Tomita et al. [2006]. Hao et al. [2016] used the basic component analysis method for finding maximum k-clique on social networks. Mirghorbani and Krokhmal [2013] proposed a method for finding a k-sized clique in k-segmented graphs. This method is based on the bit pattern used to find cliques. Though these approaches produce some satisfying results, they introduce a new problem: "not all attributes of an object have the same influence on the network, which may affect the network and compromise the quality of the resulting clusters" [Saha and Schmitt 2020]. Therefore, sometimes it does not meet user satisfaction. Sometimes a user wants to interact by providing feedback to reform clusters.

An important aspect of clustering algorithm implementation is determining how effective it is. This is a challenging challenge in and of itself due to the lack of a widely accepted definition of what defines a good cluster. We either don't know what clustering a user is searching for or if it can even be recognized in most graphs formed from real-world networks [Bader et al. 2014; Fortunato and Hric 2016]. Certain internal criteria, rooted in the inherent characteristics of the data and clusters, such as compactness, separation, and cohesion, serve as benchmarks for effective clustering. These criteria evaluate the extent to which objects within a cluster exhibit similarity to one another and the degree of differentiation between

clusters. It's important to note, however, that these criteria do not assess the alignment of clustering results with external information or the extent to which they diverge from random or expected clustering. Therefore, they are not enough in the context of social network clustering. A human-generated clustering, as an ideal solution, can be used as a remedy. This type of clustering uses objects that are considered as "ground truth" and can be expected to be recognizable by a user due to the method used to cluster a network.

Concerning the selection and comparison of clustering approaches, Rand [1971] examined the characteristics of clustering functions. He posed four questions:

1. "How well does a method retrieve "natural" clusters?
2. How sensitive is a method to perturbations of the data?
3. How sensitive is a method to incomplete dataset?
4. Given two methods, do they produce different results on the same data?"

In summary, modern clustering provides a diverse set of specialized tools geared precisely at professionals. This is an unpleasant condition, since enormous amounts of data collections in an increasing number of application fields, necessitating the use of clustering as a technique for data analysis. Clustering has gained additional users as it has progressed from a specialist use in research to a widely used analysis approach. As a result, previously overlooked concerns such as usability and applicability become vital issues. Users that want to use clustering currently have two options: employ professionals and set up a multi-year project to design a tailored clustering solution, or go on a trial-and-error journey attempting to build a suitable solution from existing methodologies. The goal of this thesis is to improve the current state of clustering despite TI violations by transforming clustering into a feedback-driven process with user interaction.

1.1 Motivation

A database object type usually consists of multiple attributes. Clustering objects involves the descriptive task of grouping these objects into clusters based on how similar their attribute values are when compared pairwise. Similarity values typically fall within the range of $[0, 1]$. When the similarity value is higher, it indicates that the objects are more alike, and conversely, a lower similarity value implies less similarity. The degree of similarity is assessed using a measure that quantifies how closely objects resemble each other. Consequently, when considering all possible pairs of objects, a similarity measure leads to the creation of a quadratic similarity matrix.

Clustering requires that objects within the same cluster should exhibit significantly greater similarities compared to objects in different clusters. The quality of clustering improves as the similarity within a cluster increases and the dissimilarity between clusters becomes more pronounced. From a technical standpoint, most clustering methods rely on distance measures between pairs of objects as input. These distance functions often include the Euclidean and Manhattan distance functions. Distance functions must satisfy amongst other the triangle inequality (TI) property.

The triangle inequality can be understood as saying that "if object x is relatively close to object y, and object y is also relatively close to object z, then objects x and z cannot be very far apart from each other". If we convert distance values to similarity values within the range $[0,1]$ using the formula $s(x,y) = e^{(} - d(x,y))$, where s represents the similarity measure and d is the distance measure, we obtain for property TI:

$$d(x,z) \leq d(x,y) + d(y,z) \overset{\cdot(-1)}{\Longleftrightarrow} -d(x,z) \geq -d(x,y) - d(y,z) \overset{(exp)}{\Longleftrightarrow}$$
$$e^{-d(x,z)} \geq e^{-d(x,y)-d(y,z)} \Leftrightarrow e^{-d(x,z)} \geq e^{-d(x,y)} \cdot e^{-d(y,z)}$$

which gives, $s(x,z) \geq s(x,y) \cdot s(y,z)$. And thus,

$$d(x,z) \leq d(x,y) + d(y,z) \Leftrightarrow s(x,z) \geq s(x,y) \cdot s(y,z) \text{ holds.}$$

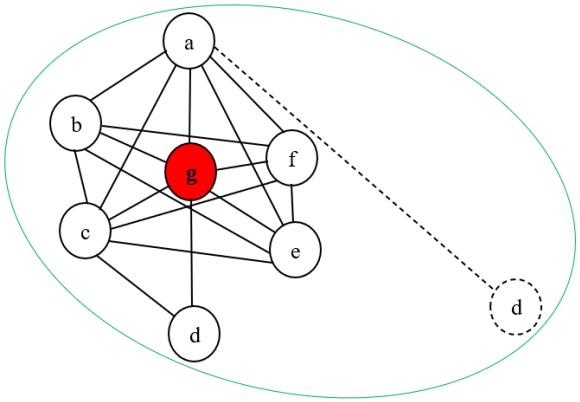


Fig. 1.1 TI violation in k-medoids clustering: due to the problem of visualizing non-TI in a plane object d appears twice.

Sometimes there are functions that fulfil all distance properties except TI. We will call them non-TI functions. The violation of TI may compromise the quality of the resulting clusters. In traditional k-medoids clustering technique, see Figure 1.1 for example, a single cluster can be found having all objects where g is the medoid. The triple $\{a,g,d\}$ within the cluster has two edges among them. There are edges between $\{a,g\}$ and between $\{g,d\}$

but no edge between $\{a,d\}$. That means the distance from a to d exceeds the sum of the distances from a to g to d. Therefore, the cohesion of d may be seriously impacted because a is close to g but very far away from d. Thus, it may violate TI property.

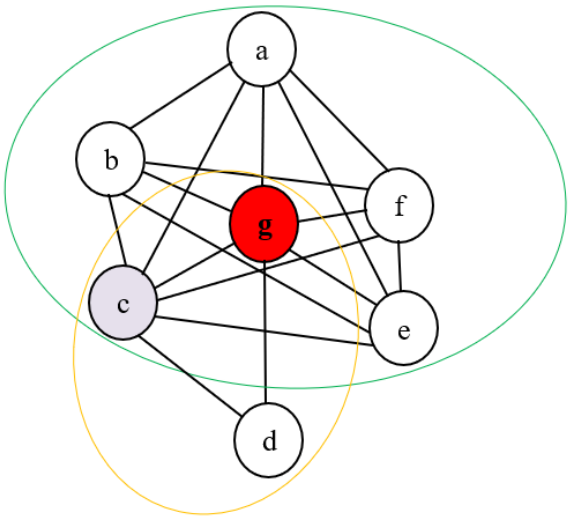


Fig. 1.2 Non-TI clustering.

But it is still possible to find meaningful clusters in such a way that a and d reside in different clusters and g resides in more than one cluster at the same time.

Consider a small social network G with eight objects namely $\{a,b,c,d,e,f,g,h\}$. Suppose a similarity matrix $S = \{s_{ij}\}$ of G is a symmetric 8×8 matrix as follows:

$$\mathrm{S} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{bmatrix} 1 & 0.86 & 0.58 & 0.71 & 0.43 & 0.2 & 0.37 & 0.48 \\ 0.86 & 1 & 0.65 & 0.98 & 0.15 & 0.29 & 0.47 & 0.4 \\ 0.58 & 0.65 & 1 & 0.55 & 0.3 & 0.49 & 0.18 & 0.25 \\ 0.71 & 0.98 & 0.55 & 1 & 0.82 & 0.77 & 0.64 & 0.9 \\ 0.43 & 0.15 & 0.3 & 0.82 & 1 & 0.58 & 0.7 & 0.66 \\ 0.2 & 0.29 & 0.49 & 0.77 & 0.58 & 1 & 0.82 & 0.57 \\ 0.37 & 0.47 & 0.18 & 0.64 & 0.7 & 0.82 & 1 & 0.65 \\ 0.48 & 0.4 & 0.25 & 0.9 & 0.66 & 0.57 & 0.65 & 1 \end{bmatrix} \end{array}$$

It is obvious in S that the triple $\{c,d,e\}$ violates the triangle inequality:

$$s(c,e) \geq s(c,d) \cdot s(d,e) \Rightarrow 0.3 \ngeq 0.55 \cdot 0.82 \Rightarrow 0.3 \ngeq 0.451$$

Despite of this violation, it still makes sense to find meaningful clusters. For example, an adjacency matrix $Adj = (a_{ij})$ can be derived from the similarity matrix S by using a mathematical function with threshold value $\theta = 0.5$:

$$a_{ij} = \begin{cases} 1 & \text{if } s_{ij} \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

$$Adj = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Therefore, Figure 1.3 shows the graphical representation of a small social network G based on adjacency matrix Adj.

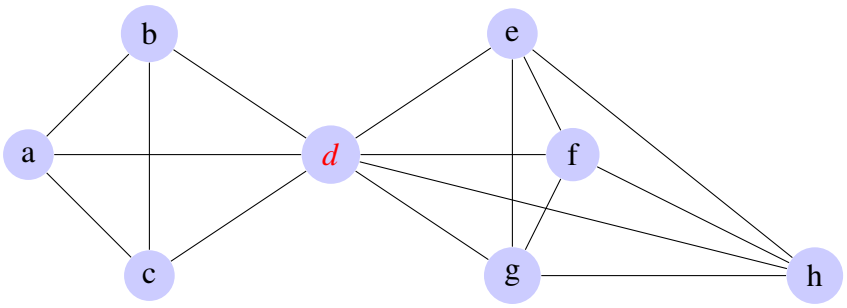


Fig. 1.3 Example of a small social network G.

Technically, G contains two maximal cliques, $\{a,b,c,d\}$ and $\{d,e,f,g,h\}$. And the node d is common for both clusters. We regard cliques as clusters. They are meaningful in that scenario, although TI does not hold.

The Non-TI clustering approach introduced by Saha and Schmitt [2020] offers promising results, however it creates a new problem of not always satisfying the user's needs.

Example 1.1. In a social network of alumni of a school, let $P = \{p_1, p_2, ..., p_n\}$ be a set of n people and the similarity among people can be sufficiently expressed by relying on their admission year and graduation year. The similarity value between two people p_1 and p_2

can be expressed as a Commuting Quantum Query Language (CQQL) [read Chapter 4 for details] condition q proposed by Schmitt [2008].

The set $P = \{p_1, p_2, ..., p_n\}$ represents n people in a social network of alumni from a school. The similarity between individuals can be determined using their admission year and graduation year. The similarity between two people, p_1 and p_2, can be expressed using the Commuting Quantum Query Language (CQQL) [read Chapter 4 for details] condition q, as proposed by Schmitt [2008]:

$$q(p_1, p_2) = \underbrace{school(p_1, p_2)}_{Boolean\ condition} \wedge \underbrace{graduation(p_1, p_2)}_{similarity\ condition} \wedge \underbrace{admission(p_1, p_2)}_{similarity\ condition} \tag{1.2}$$

where

$$school(p_1, p_2) = (p_1.school = p_2.school) \in \{0, 1\}$$
$$graduation(p_1, p_2) = (p_1.graduation \approx p_2.graduation) \in [0, 1]$$
$$admission(p_1, p_2) = (p_1.admission \approx p_2.admission) \in [0, 1]$$

The similarity between persons can be accurately expressed by taking into account their year of admission and graduation. However, it is possible for friends to have different graduation years but the same admission year. In this case, their graduation year should have less impact on the similarity calculation than their admission year. A solution to this is the *"quantitative weighting approach"*, which allows for the assigning of weights to atomic conditions based on their relevance. By re-weighting, new clusters are formed. This approach is referred to as *"adaptive clustering"*. Systems that use this method are not limited to pre-defined user requirements and can better meet their needs compared to systems with a fixed set of aspects.

In this dissertation, CQQL [Schmitt 2008] is used to implement the Relevance Feedback (RF) based on user needs. This thesis combines two approaches for social network clustering: CQQL, a quantum logic-based query language that incorporates weights that reflect the varying significance of sub-conditions and personalises resulting clusters using an RF approach, and user feedback, which addresses the dynamic nature of the search process and information need. The formulation of weighted CQQL queries and the idea behind relevance feedback make up the theoretical background of the dissertation, offering a holistic and consistent theory for social network clustering.

1.2 Contributions

Clustering is a technique that has many available tools but lacks clear guidance on how to effectively use them. To achieve successful clustering, users must select an appropriate algorithm and run it with the right parameters. Both activities have a significant impact on the outcome and, if not done appropriately, might result in pointless clusterings. The user must analyze the produced result and determine whether or not the clusters are meaningful. If the result isn't satisfactory, the parameters or algorithm must be tweaked to produce a better result. To carry out these tasks efficiently, a certain level of understanding is required. Current clustering practice focuses primarily on clustering generation and treats all of the aforementioned activities as separate tasks. Clustering is a difficult technique for novice users to adopt due to the loose coupling of activities and the arbitrariness with which they are carried out.

The main objective of this thesis is to address the challenges faced in conventional clustering practices by introducing a new approach that involves user interaction in the design of clustering structures. This approach aims to simplify the process for users by integrating them into the design and eliminating complex technical aspects. The clustering actions are no longer seen as separate stages but as integral parts of a comprehensive process, which gives users a standardized procedure to follow. The goal is not to create a universal clustering algorithm that can handle all datasets, but a flexible tool that can be easily applied to current methodologies. The thesis utilizes simplicity and abstraction to create a template for the process, which is versatile and offers essential qualities that can be achieved in different ways. The novel interaction introduced in the thesis allows users to understand and improve clustering results through natural feedback, which requires new ways of clustering construction that can handle a wide range of existing techniques and provide an interface for incorporating user feedback into the results. In summary, this thesis presents a new and innovative way of designing clustering structures by involving users and improving the process through natural feedback.

- We provide an analysis and assessment of the existing clustering algorithms in use today. We focus on the basic types of clustering algorithms and introduce some of their representatives to give an idea of the range of options available. We also go over many methods for evaluating findings that are currently available. All of the techniques presented are evaluated for their applicability and usability. Versatility, robustness, configuration, complexity, and user support are all factors to consider. We deduce the primary repercussions users encounter in current clustering from our analysis.

- A template for a flexible clustering procedure is defined. This provides a summary of the essential tasks and requirements that must be met to achieve a comprehensive and user-friendly clustering application. The procedure has two primary purposes. The first is to take care of clustering creation in its entirety, including method definition and specification, options for integrating and managing existing clustering techniques, and the establishment of a control interface that receives and applies user feedback. The provision of an interactive interface, which communicates the properties of the clustering result to the user, is the second key role.

- We suggest novel approaches for incorporating existing clustering techniques into our workflow. The concept of ensemble clustering is utilized as a starting point for this since it allows various solutions to be combined into a single final solution, which enhances overall quality. This concept is expanded upon, and approaches for the controlling integration of various outcomes are proposed. This integration approach also serves as an abstraction layer for the management of numerous alternative clustering algorithms, due to the control choices we've presented.

- We define a compact set of universally valid feedback options for clustering adjustment. We do this by taking the users' perspective, abstracting the most basic clustering modifications, and converting them into feedback. Unlike typical adjustments, which change the reason for clustering, i.e. the parameters used to create it, our feedback directly expresses the impacts a user wishes to establish in the result. Usability is improved by keeping feedback stable and independent of the underlying algorithms, in addition to this direct character.

- We use a human-generated clustering as a benchmark for evaluating the effectiveness of our approach. This benchmark clustering is considered to be the ideal solution and serves as a reference point to calculate the *clustering distance*, which is a measure of the dissimilarity between two clusterings.

1.3 Structure of the Dissertation

There are five main sections to the thesis. The implementation and evaluation presented in part four have a theoretical foundation in the first three parts. The final section of the thesis concludes by recapping its main points and identifying possible avenues for further research. The dissertation is organized in the following manner:

Part I. Foundations and Background To make reading easier, a list of central definitions crucial for understanding is provided in Chapter 2.

Chapter 3 outlines the core concepts and techniques of Social Network and Social Network Clustering by defining key terms and laying out fundamental principles. Additionally, it provides a comprehensive look at conventional Social Network Clustering methods.

Part II. Learning User-specific Weights in Logic-based Queries In Chapter 4, the implementation of a Non-TI Clustering based on Quantum Logic is described, including an explanation of the theoretic foundation of the CQQL query language which is based on quantum logic.

The Relevance Feedback based learning approach, which is fundamentally based on CQQL, is introduced in Chapter 5. The concept of feedback is also covered in this chapter, with a particular emphasis on the so-called Explicit and Implicit feedback approaches used in databases.

The chapters 4 and Chapter 5 provide the theoretical foundation for the conceptual design and implementation discussed in Chapter 7 of the dissertation.

Part III. Adaptation of Clustering by Feedback Adjustment Chapter 6 defines the adaptation of clustering through feedback adjustment, and it evaluates the dissimilarity between two clusterings through the calculation of a clustering distance.

Part IV. Implementation and Evaluation In Chapter 7, experiments are performed to evaluate the non-TI clustering approach. It is divided into three separate evaluations of the non-TI clustering approach:

- Clustering with unweighted and weighted CQQL.
- Clustering based on user feedback.
- Effectiveness of the clustering based on clustering distance.

Part V. Conclusions Finally, Chapter 8 summarizes the findings and contributions of this work. The limitations of the proposed methods along with possible future works are briefly addressed at the end of the chapter.

Chapter 2

Fundamentals

2.1 Logic

According to the definition provided by the Oxford English Dictionary, logic can be defined as

> "the systematic use of symbolic and mathematical techniques to determine the forms of valid deductive argument."

The focus of this thesis is on formal logical systems used for reasoning such as classical Boolean logic. Schöning1989, Siefkes [1990], and Dominich [2008] all have similar definitions. A formal logic is a language that operates on symbols to determine the truth value of a statement. A statement is composed of propositions represented by capital letters and logical connectors such as conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). In Boolean logic, propositions (or elementary propositions [Wittgenstein Wittgenstein, see. Sec. 5 and 5.01]) can be replaced with their truth value, either $true$ or $false$, which are represented by 1 and 0, respectively. The rules for determining the truth value of a sentence in a formal logical system, such as classical Boolean logic, are then put into practice, as outlined in Table 2.1. These rules are defined by axioms. For example, the law of the excluded middle, represented as $A \vee \neg A$, and the law of identity, expressed as $A = A$, are two axioms. It is true that both of these statements are true.

2.2 Boolean Algebra

A Boolean algebra is a formal logical systems with at least two elements, 0 and 1, endowed with binary operations $\wedge$ and $\vee$ and a unary operation $\neg$. The Boolean operations satisfy the following axioms:

Table 2.1 Semantics of Boolean logical connectors and propositions A and B

A	B	$A \wedge B$	$A \vee B$	$\neg A$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Definition 2.1 (Axiom of Associativity).

$$(A \wedge B) \wedge C = A \wedge (B \wedge C) \quad \textit{and} \quad (A \vee B) \vee C = A \vee (B \vee C) \tag{2.1}$$

Definition 2.2 (Axiom of Commutativity).

$$A \wedge B = B \wedge A \quad \textit{and} \quad A \vee B = B \vee A \tag{2.2}$$

Definition 2.3 (Axiom of Distributivity).

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \quad \textit{and} \quad A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C) \tag{2.3}$$

Definition 2.4 (Axiom of Identity).

$$A \wedge 1 = A \quad \textit{and} \quad A \vee 0 = A \tag{2.4}$$

Definition 2.5 (Axiom of Complements).

$$A \wedge \neg A = 0 \quad \textit{and} \quad A \vee \neg A = 1 \tag{2.5}$$

Definition 2.6 (Axiom of Idempotence).

$$A \wedge A = A \quad \textit{and} \quad A \vee A = A \tag{2.6}$$

Definition 2.7 (Axiom of Double Negation).

$$\neg(\neg A) = A \tag{2.7}$$

Definition 2.8 (De Morgan's Law).

$$\neg(A \wedge B) = \neg A \vee \neg B \quad \textit{and} \quad \neg(A \vee B) = \neg A \wedge \neg B \tag{2.8}$$

2.3 Hilbert Space

A Hilbert space H is a mathematical vector space with either finite or infinite dimensions. It must have an inner product to measure the distance and angle between vectors. The Hilbert space must also be complete, meaning it has limits and is suitable for the use of calculus techniques (see Dieudonné [1960, p. 115]).

2.4 Dirac notation

"Dirac [1958] introduced the Dirac notation, which can be considered the lingua franca of quantum mechanics. The definitions that follow are similar to the overviews of notation provided by Schmitt [2008, cf. Sec. 2] and van Rijsbergen [2004, cf. App. I]. More examples of the notation can be found in these contributions.

Definition 2.9 (Ket vector). A column vector x in a Hilbert space H is represented by a ket: $|x\rangle$.

Definition 2.10 (Bra vector). The transpose of $|x\rangle$ yields a row vector bra: $\langle x|$.

Definition 2.11 (Inner product in bra-ket form). The inner (or scalar) product of two ket vectors $|x\rangle$ and $|y\rangle$ is stated by a bra(c)ket: $\langle x|y\rangle$.

Definition 2.12 (Norm of a ket vector). The norm $||\,|x\rangle\,||$ of a ket $|x\rangle$ is defined as: $\sqrt{\langle x|x\rangle}$.

Definition 2.13 (Outer product in bra-ket form). The outer product of two kets $|x\rangle$ and $|y\rangle$, which generates a linear operator in form of a matrix, is denoted by: $|x\rangle\langle y|$.

Definition 2.14 (Tensor product). The tensor product between two kets $|x\rangle$ and $|y\rangle$ is denoted by $|x\rangle \otimes |y\rangle$ or short by $|xy\rangle$. If $|x\rangle$ is m-dimensional and $|y\rangle$ n-dimensional then $|xy\rangle$ is an $m \cdot n$-dimensional ket vector. The tensor product of two-dimensional kets $|x\rangle$ and $|y\rangle$ is defined by:

$$|xy\rangle \equiv |x\rangle \otimes |y\rangle \equiv \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \equiv \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ x_2 y_1 \\ x_2 y_2 \end{pmatrix}$$

The tensor product of matrices is defined analogously.

Definition 2.15 (Projector). A projector $p = \sum_i |i\rangle\langle i|$ is a symmetric ($p^t = p$) and idempotent ($pp = p$) linear operator defined over a set of orthonormal vectors $|i\rangle$. Multiplying a projector with a state vector means to project the vector onto the respective vector subspace."

[Schmitt et al. 2008]

2.5 Notation

Notation and symbols used throughout this thesis are summarized in Tables 2.2, 2.3, and 2.4.

Table 2.2 General notation and symbols.

SYMBOL	MEANING
$\mathbb{N}, \mathbb{N}^+$	Natural numbers with and without zero
$\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}^+$	All, non-negative, positive real numbers
$v_i = [v]_i$	Element at position i of vector v
$\|v\|$	Norm
$m_{ij} = [M]_{ij}$	Element at (i, j) of matrix M
$f(x)$	Function of x
$\approx$	Similarity
$\sum$	Summation: sum of all values in range of series
$\otimes$	Tensor Product
$\forall$	For All
$\exists$	There Exists

Table 2.3 Notation and symbols for Sets.

SYMBOL	MEANING
$\{\}$	Set: a collection of elements
$A \cup B$	Union: in A or B (or both)
$A \cap B$	Intersection: in both A and B
$A \subseteq B$	Subset: every element of A is in B
$A \subset B$	Proper Subset: every element of A is in B, but B has more elements
$A \not\subset B$	Not a Subset: A is not a subset of B
A^C	Complement: elements not in A
$A - B$	Difference: in A but not in B
$a \in A$	Element of: a is in A
$b \notin A$	Not element of: b is not in A
$\emptyset$	Empty set
$\mathbb{U}$	Universal Set: set of all possible values (in the area of interest)
$\|A\|$	Cardinality: the number of elements of set A
$\mid$	Such that

Table 2.4 Notation and symbols for graphs.

SYMBOL	MEANING
G	A graph
V	Set of nodes in the graph
E	Set of edges in the graph
$\|G\|$	Order of G, i.e., $\|V(G)\|$
$\|\|G\|\|$	Size of G, i.e., $\|E(G)\|$
$deg(v)$	Degree of vertex v
$N(v)$	Neighbors of v
$\overline{G}$	Complement graph of G
$G[V]$	Subgraph induced by V
Adj	Adjacency matrix, $\|V\| \times \|V\|$
S	Similarity matrix, $\|V\| \times \|V\|$
$\subset, \sqsubseteq$	Subgraph, induced subgraph relation
$G \setminus S$	Graph obtained by deleting S

Chapter 3

Social Network Analysis

3.1 Social Network

The relationships between social entities can be depicted in a social network. This network is represented as an undirected graph with a set of actors, such as individuals or organizations, as nodes and ties between them as edges. An example of this concept can be seen in a friendship network on Facebook where students from the same department of a college may form a group due to their shared location and academic background. In contrast, students with different backgrounds, education, or geography have a lower likelihood of forming networks. The homogeneity of the network can reflect the similarity between nodes in terms of parameters like location or education.

A *graph* is represented by $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of n nodes and E is the set of edges. The nodes represent the network entities and edges represent the relationships between the entities. An edge $e_{ij} \in E$ between two nodes v_i and v_j is represented by a pair of the nodes (v_i, v_j). A graph G is called a *weighted graph* if a weight is assigned to each edge.

Example 3.1. An illustration of a small social network is shown in Figure 3.1. The nodes, labeled as a to j, are the entities, and the edges signify the relationships, such as friendship, between them. For instance, node b has a friendship relationship with nodes a, c, d, and e.

Clusters in a network are groups of nodes or vertices within the network that exhibit a higher degree of connectivity to each other compared to nodes outside of the group. Edges and vertices are not distributed uniformly, but rather in locally dense groups. Common interests or goals, friendship, or other similarities between actors are common reasons for the implicit or explicit formation of groups.

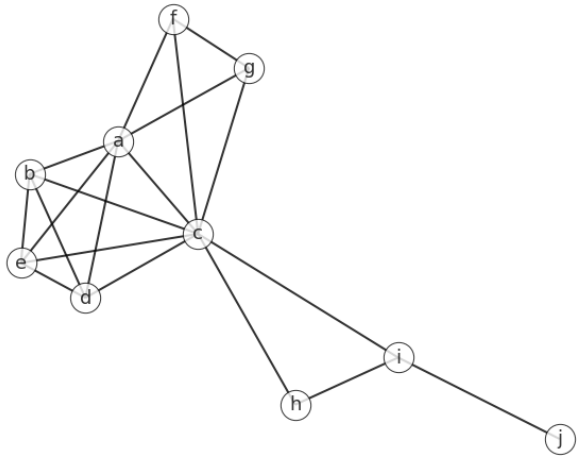


Fig. 3.1 A small social network.

3.2 Varieties of Social Networks

There are numerous instances of social networks beyond just networks of friends. Let's list some other examples of networks that display a local quality of relationships.

Telephone Networks

The example in Figure 3.2 illustrates a small telephone network. The nodes in the network represent individual phone numbers and if a call was made between two nodes during a specific time period, such as last month or "ever," an edge is formed between them. The edges can be weighted based on the number of calls made between the phones throughout the specified time period.

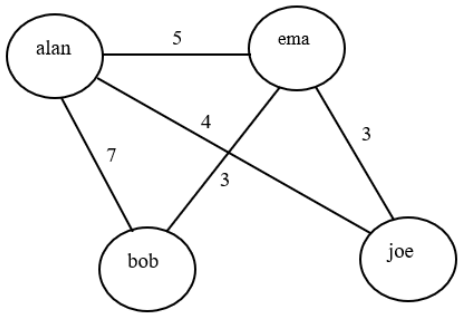


Fig. 3.2 A small telephone network.

In a telephone network, communities will arise from groups of individuals who communicate frequently, such as friends, members of a club, or people who work for the same company.

Email Networks

Individuals are represented by the nodes, which are email addresses. An edge indicates that at least one email was sent between the two addresses in at least one direction. Alternatively, if there were emails in both directions, we could only place an edge. We avoid seeing spammers as "friends" with all of their victims in this way. Another option is to categorize edges as strong or weak. Communication in both directions is represented by strong edges, and communication in one direction is represented by weak edges. The communities identified in email networks are formed by the same kind of groupings that we see in telephone networks. Individuals who text other people on their cell phones form a similar network.

Collaboration Networks

Individuals who have published research articles are depicted as nodes. Edges exist between two individuals who have co-authored one or more papers. The strength of these edges can be determined by the number of joint publications. Communities can be formed in this network based on the authors who focus on a particular subject matter.

Another way to look at the same data is as a graph with nodes that are publications. If two papers share at least one author, they are related by an edge. We now create communities, which are collections of writings on a single subject.

There are various additional types of data that can be used to create two networks in the same way. We may look at the people who edit Wikipedia pages as well as the articles that they edit, for example. If two editors have edited the same article, they are related. The communities are groups of editors who share a common interest in a particular topic. The creation of a network of articles that are connected if they were modified by the same person can be used to find communities of articles on similar topics.

This concept can be applied to the data used in collaborative filtering, where two networks are established, one for customers and one for items. The communities in these networks are determined by customers who buy similar items and by products that are purchased by the same customers.

Other Examples of Social Graphs

Many other processes, particularly those demonstrating locality, result in graphs that resemble social graphs. Information networks (documents, web graphs, patents), infrastructure networks (roads, airlines, water pipelines, power grids), biological networks (genes, proteins,

food-webs of animals eating each other), and other sorts, such as product co-purchasing networks, are only a few examples (e.g., Groupon).

3.3 Social-Network Graph Drawing

A drawing of a graph is a pictorial representation of the vertices and edges of a graph. It is important to note that this visual representation should not be confused with the graph itself, as different layouts can represent the same graph. In abstract, what truly matters is identifying which pairs of nodes are connected by edges.

There are many different graph layout strategies available that can be used to visualise a graph. We discuss the force-directed placement strategy proposed by Fruchterman and Reingold [1991] in this dissertation.

Normally, graphs are depicted with their vertices as points in a plane and their edges as line segments or curves connecting those points. Fruchterman-Reingold concentrates on the most general class of graphs: general, undirected graphs, drawn with straight edges. In their paper, they introduced an algorithm that attempts to produce aesthetically–pleasing, two–dimensional pictures of graphs by doing simplified simulations of physical systems.

They are concerned with drawing general undirected graphs according to some generally–accepted aesthetic criteria:

- Evenly distribute the vertices in the frame.
- Minimize edge crossings.
- Make edge lengths uniform.
- Reflect inherent symmetry.
- Conform to the frame.

They have only two principles for graph drawing:

- Vertices that are neighbors should be drawn near each other.
- Vertices should not be drawn too close to each other.

How close vertices should be placed depends on how many there are and how much space is available. For more details please read Fruchterman and Reingold [1991].

3.4 Social-Network Graphs Clustering

A *cluster* is a set of nodes, and *clustering* is the process of grouping a set of nodes into subsets, where each subset represents a cluster. Consider the nodes: $S = x_1, x_2, ..., x_n$. Clustering divides this set into k subsets $(C_1, C_2, ..., C_k)$, each representing a cluster of similar nodes. That is, $S = C_1 \cup C_2 \cup ... \cup C_k$ and clustering $C = \{C_1, C_2, ..., C_k\}$.

A measure of similarity or dissimilarity is required to organize similar nodes into groups. The degree to which two nodes are alike is measured numerically by the similarity between them. As a result, dissimilarity is a numerical measure of how distinct the two nodes are. The higher the similarity, the lower the dissimilarity, and the lower the similarity, the higher the dissimilarity. When discussing dissimilarity, the term "distance" is commonly employed. The discussion of the various clustering techniques and their applications is relevant because the objective of clustering aligns with our aim of identifying groups of actors who are similar or close to each other.

3.4.1 Measures for Social-Network Graphs

The first step in applying typical clustering techniques to a social-network graph is defining a distance measure. Depending on what the weights on the graph's edges indicated, as in a telephone network (see Figure 3.2), these weights may be used to label a distance measure. However, when the edges are unweighted, as in a "friends" graph (see Figure 3.1), we are limited in our ability to establish one suitable distance.

Distance Measure

In social network clustering, distance measures are used to quantify the similarity or dissimilarity between nodes (individuals) within the network. These measures help identify clusters or communities of nodes that exhibit similar patterns of connections or interactions.

A distance $d(x,y)$ between two nodes x and y fulfils the following properties:

- *Non-negativity:* $d(x,y) \geq 0$ for all x and y
- *Identity of indiscernibles:* $d(x,y) = 0$ only if $x = y$
- *Symmetry:* $d(x,y) = d(y,x)$ for all x and y
- *Triangle inequality:* $d(x,z) \leq dist(x,y) + d(y,z)$ for all x, y and z

Distance measures that adhere to all of the specified properties are referred to as metrics. These properties can be beneficial for certain applications, for instance, if the triangle

inequality property holds, clustering can be performed more efficiently. Here are some common distance measures used in social network clustering:

Euclidean distance

This measure calculates the straight-line distance between two nodes in a multi-dimensional space. In social networks, each dimension may represent a different attribute or feature of the nodes. Euclidean distance can be used to assess the dissimilarity between nodes based on their attribute values. The Euclidean distance between two nodes in n-dimensional space is defined by the following formula:

$d(P,Q) = \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^2}$
where $P = \{p_1, p_2, ..., p_n\}$ and $Q = \{q_1, q_2, ..., q_n\}$.

Manhattan distance

The Manhattan distance between two nodes $P = \{p_1, p_2, ..., p_n\}$ and $Q = \{q_1, q_2, ..., q_n\}$ in n-dimensional space is the sum of the distances in each dimension.

$d(P,Q) = \sum_{i=1}^{n} |P_i - Q_i|$

Distance functions must satisfy amongst other the triangle inequality property. But sometimes the triangle inequality property is violated. The reason is that when there are three nodes connected by two edges, if there is an edge between nodes A and B and another edge between nodes B and C but no edge between nodes A and C, then the distance between nodes A and C is greater than the sum of the distances between nodes A and B and nodes B and C.

Similarity Measure

The similarity measure is a way of measuring how nodes or objects are related or being close to each other. Typically, the similarity values are expressed within the range of $[0,1]$. A similarity value closer to 1 indicates a stronger similarity between the objects, whereas a value closer to 0 indicates less similarity. The degree to which objects are alike is determined by a measure that assesses their similarities. As a result, calculating the similarity between all pairs of objects results in a quadratic similarity matrix.

Similarities, also have some well known properties.

1. $s(x,y) = 1$ (or maximum similarity) only if $x = y$
2. $s(x,y) = s(y,x)$ for all x and y (Symmetry)

where $s(x,y)$ is the similarity between nodes, x and y.

If we transform distance values into similarity values within the range of $[0,1]$ using the function $s(x,y) = e^{-d(x,y)}$, we obtain for property TI:

$$d(x,z) \leq d(x,y) + d(y,z) \overset{\cdot(-1)}{\Longleftrightarrow} -d(x,z) \geq -d(x,y) - d(y,z) \overset{(exp)}{\Longleftrightarrow}$$
$$e^{-d(x,z)} \geq e^{-d(x,y)-d(y,z)} \Leftrightarrow e^{-d(x,z)} \geq e^{-d(x,y)} \cdot e^{-d(y,z)}$$

which gives, $s(x,z) \geq s(x,y) \cdot s(y,z)$. And thus, $d(x,z) \leq d(x,y) + d(y,z) \Leftrightarrow s(x,z) \geq s(x,y) \cdot s(y,z)$ holds.

Cosine similarity

Cosine similarity is a technique used to determine the similarity between two documents or to rank documents in relation to a set of query words. Given vectors P and Q, the cosine similarity between these vectors is calculated as:

$sim(P,Q) = \frac{P \cdot Q}{|P||Q|}$

where $P \cdot Q = \sum_{i=1}^{n} P_i Q_i$ is the dot product of the vectors P and Q, and $|P|$ is the Euclidean norm of vector $P = \{p_1, p_2, ..., p_n\}$, which is defined as $\sqrt{p_1^2 + p_2^2 + ... + p_n^2}$, or the length of the vector. Similarly, $|Q|$ is the Euclidean norm of vector Q. The cosine similarity measure is calculated as the cosine of the angle between the two vectors. If the cosine value is equal to 0, it indicates that the vectors are orthogonal and have no match. The closer the cosine value is to 1, the lower the angle between the vectors and the better the match. Note that cosine similarity is referred to as a non-metric measure as it does not satisfy all the properties of a metric measure.

Jaccard similarity index

The Jaccard similarity index (also known as the Jaccard similarity coefficient) analyzes members from two sets to determine which are common and which are unique. It's a proportional measure of similarity between two sets of data, ranging from 0% to 100%. The larger the proportion, the closer the two groups are. Although it is simple to use, it is particularly sensitive to tiny sample sizes and can produce incorrect findings, especially when dealing with very small samples or data sets with missing observations.

The Jaccard similarity measures is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets.

$J(P,Q) = \frac{|P \cap Q|}{|P \cup Q|}$

CQQL

CQQL, which stands for Commuting Quantum Query Language, is a language used for querying the similarity between two objects. It provides the ability to incorporate weighting into queries and formulate logic-based queries that include both Boolean and similarity conditions. The evaluation of a CQQL condition returns a similarity value within the interval of $[0,1]$ for the similarity condition, indicating the closeness of attribute values between two objects. A CQQL expression can be seen as a similarity measure when all atomic conditions are similarity measures between two objects and CQQL combines them by logic junctors. The boolean values $true$ for perfect match and $false$ otherwise are mapped to the score values 1 or 0, respectively, for the boolean condition. A detailed discussion can be found in Schmitt [2008] and Schmitt [2019].

3.4.2 Matrices that describe Graphs

To utilize the concept of matrix algebra in finding good partitions in a graph, it's necessary to have an understanding of three different matrices that represent aspects of the graph. Consider a set of n individuals, denoted as $P = \{p_1, p_2, ..., p_n\}$. The first should be familiar: the ***adjacency matrix*** $Adj = \{a_{ij}\}$ is the $n \times n$ matrix defined as

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge } (p_i, p_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The adjacency matrix for the social network of Figure 3.1 appears below:

$$Adj = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cccccccccc} a & b & c & d & e & f & g & h & i & j \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

The second matrix we need is the ***degree matrix*** for a graph. This example matrix has nonzero entries only on the diagonal. The entry for row and column i is the degree of the ith node.

The degree matrix for the social network of Figure 3.1 is shown below:

$$Deg = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cccccccccc} a & b & c & d & e & f & g & h & i & j \end{array} \\ \left[\begin{array}{cccccccccc} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

The Laplacian matrix, represented as $L = Deg - Adj$, is a matrix that can be derived from a graph's adjacency matrix Adj and degree matrix Deg. The Laplacian matrix is the difference between the degree matrix and the adjacency matrix. The diagonal entries of the Laplacian matrix are the same as those in the degree matrix Deg. For entries off the diagonal, at the i^{th} row and j^{th} column, a value of -1 is present if there is an edge between nodes i and j, and a value of 0 if not. The Laplacian matrix has the property that each row and each column adds up to zero, which is typical of any Laplacian matrix.

The Laplacian matrix for the social network 3.1 is shown below:

$$L = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cccccccccc} a & b & c & d & e & f & g & h & i & j \end{array} \\ \left[\begin{array}{cccccccccc} 6 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 8 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 4 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 3 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & -1 & 3 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right] \end{array}$$

3.4.3 Clustering Methods

The various clustering techniques discussed in the following can be adapted to graphs as they are based on a general distance or similarity measure. We will briefly touch upon their suitability and direct readers to later chapters for more in-depth information on the methods.

Partitioning Methods

The main aim of dividing data objects into k distinct clusters is to identify groups where the members exhibit high similarity within their respective clusters but differ significantly from members in other clusters. The Partitioning Around Medoids (PAM) method, introduced by Kaufman and Rousseeuw [1990], aims to identify a representative object, known as a medoid, for each cluster. These medoids are the objects that are most centrally located within their respective clusters.

Initially, a set of k objects is chosen as the initial medoids. Then, during each step of the algorithm, all objects in the input dataset that are not currently designated as medoids are individually assessed to determine if they should replace one of the existing medoids. In other words, the algorithm checks if there is an object that would be a better fit as a medoid than the current ones. The decision to swap medoids with other non-selected objects is based on minimizing the total cost.

PAM's approach involves representing a cluster by its medoid, and as a result, it is also commonly referred to as the k-medoids algorithm. Algorithm 1 summarizes the PAM technique.

When evaluating the cost associated with swapping a non-medoid object (let's call it p_{rand}) with a medoid object (let's call it p_i), there are four scenarios to consider for each non-medoid object p. These scenarios are as follows:

Case 1: If p originally belongs to the medoid object p_i, and after the swap, p becomes closer to another medoid object p_j than to p_i, then p is reassigned to p_j.

Case 2: If p originally belongs to the medoid object p_i, and after the swap, p becomes closer to p_{rand} than to p_i, then p is reassigned to p_{rand}.

Case 3: If p originally belongs to one of the other medoid object p_j (where j is not equal to i), and after the swap, p remains closest to p_j, then there is no need to reassign p.

Case 4: If p originally belongs to one of the other medoid object p_j, and after the swap, p becomes closer to p_{rand} than to p_j, then p is reassigned to p_{rand}.

Algorithm 1: PAM algorithm for partitioning.

Input: k: the number of clusters, D: a data set containing n objects
Output: A set of k clusters

```
Arbitarily choose k objects in D as initial medoids
C = −∞
while C < 0 do
    for each non-medoid object p in D do
        find the nearest medoid and assign p to the corresponding cluster
    end
    randomly select a non-medoid p_rand
    compute the overall cost C of swapping a medoid p_i with p_rand
    if C < 0 then
        swap p_j with p_rand to form a new set of k medoids
    end
end
return k clusters
```

The cost function in this context is defined as the change in the value of the distortion function that occurs when a medoid object is replaced by a non-medoid object. The total cost C of making such replacements is calculated as the sum of the costs incurred by all non-medoid objects. If the total cost C is negative, it indicates that the replacement is permissible because it reduces the value of the distortion function.

Example 3.2. Consider a social network of objects located in a $2-$dimensional space, as shown in Figure 3.1. The user wants to divide the objects into 3 clusters, with $k = 3$.

The starting point of the k-medoids method is the dissimilarity matrix which is obtained by using distance or similarity measure. Clustering by k-medoids partitioning involves selecting three random objects as initial cluster centers (medoids) and assigning each object to a cluster based on its proximity to the cluster center. The cluster centers will then be updated. Figure 3.3 shows the resulting clusters $\{a,c,g,f\},\{b,d,e\}$, and $\{h,i,j\}$ by applying k-medoids clustering method. The Fruchterman-Reingold algorithm [Fruchterman and Reingold 1991] is used to visualize social networks by creating 2D representations based on the adjacency matrix, which is derived from the dissimilarity matrix.

By observing the triple $\{h,i,j\}$ closely, it is visible that the triple violates the TI property. Because there are edges between $\{h,i\}$ and between $\{i,j\}$, but there is no edge between $\{h,j\}$. Therefore, h and j must belong to different clusters to achieve meaningful clusters.

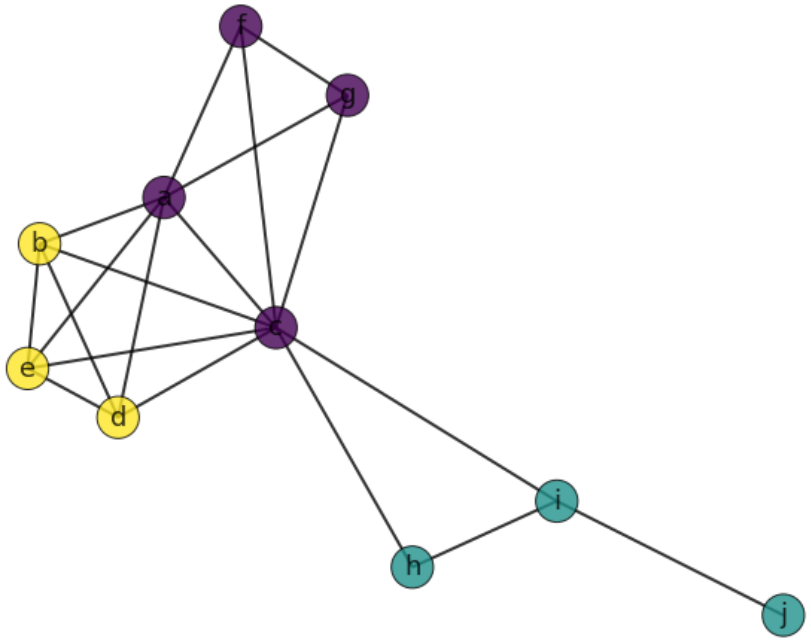


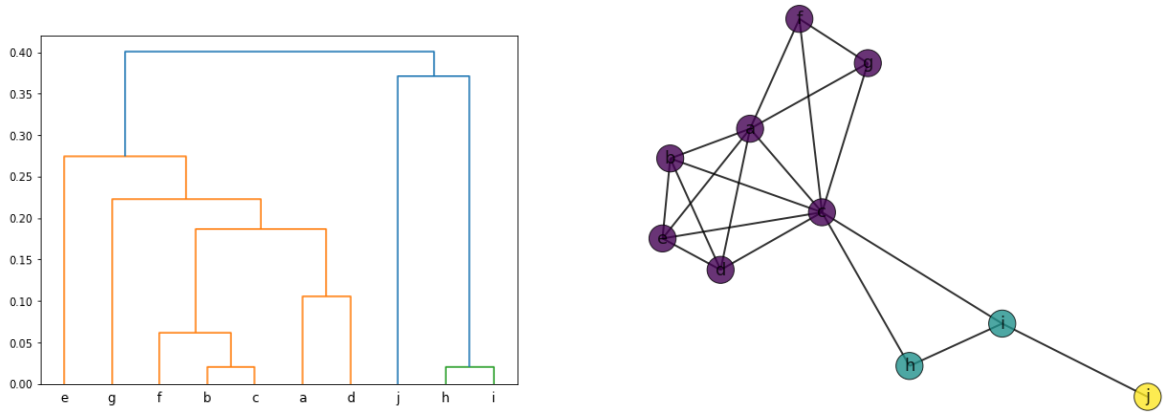
Fig. 3.3 Clustering of data objects $\{a,b,c,d,e,f,g,h,i,j\}$ using the k-medoids method.

Hierarchical Methods

Hierarchical clustering creates a cluster hierarchy, often known as a dendrogram, or a tree of clusters. Child clusters exist in every cluster node, while sibling clusters divide the points covered by their shared parent. This method enables data exploration at many levels of granularity. Hierarchical clustering methods are categorized into *agglomerative* (bottom-up) and *divisive* (top-down) [Jain and Dubes 1988]. An *agglomerative clustering* starts with one-point (singleton) clusters and merges two or more of the most appropriate clusters in a recursive manner. A *divisive clustering* starts with a single cluster of all objects and splits the most appropriate cluster recursively. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved.

Example 3.3 (Agglomerative versus divisive hierarchical clustering.). Consider the social network of a data set of ten objects $\{a,b,c,d,e,f,g,h,i,j\}$, as depicted in Figure 3.1. The single linkage method is a well-known agglomerative hierarchical clustering technique. In this approach, the distance between two clusters is determined by measuring the distance between the closest pair of objects, with each object belonging to a different cluster. During the merging step, the algorithm identifies the nearest pair of clusters and combines them into a new, single cluster. Subsequently, it updates the distances between this newly formed cluster and the other clusters that remain unchanged. This merging process continues iteratively until the number of clusters ultimately reaches one.

The outcome of this algorithm is the creation of clusters in such a way that every member of a cluster shares a closer relationship with at least one other member of the same cluster than with any object outside of it. Additionally, this method has the capability to group together a chain of objects into a single cluster and can identify clusters with arbitrary shapes.



(a) Dendogram (single-linkage) (b) Clusters ($k = 3$)

Fig. 3.4 Hierarchical clustering of data objects $\{a,b,c,d,e,f,g,h,i,j\}$.

In contrast, the divisive method works in a different way. To begin, all of the objects are combined into one cluster. The cluster is divided based on some criterion, such as the maximum Euclidean distance between the cluster's nearest neighbors. The process of cluster splitting continues until each new cluster contains only one object.

A dendrogram showing the clustering of objects $\{a,b,c,d,e,f,g,h,i,j\}$ is depicted in Figure 3.4a. The vertical axis represents the distance scale between the clusters. The algorithm merges two groups of objects into a single cluster when their distance is roughly 0.38, for example, $\{a,b,c,d,e,f,g\}$ and $\{h,i,j\}$

Figure 3.4b shows three clusters $\{a,b,c,d,e,f,g\}$, $\{h,i\}$, and $\{j\}$ by applying single-linkage hierarchical clustering method. It is clearly visible that the triple $\{b,a,f\}$ violates the TI property. Because there are edges between $\{b,a\}$ and between $\{a,f\}$ but there is no edge between $\{b,f\}$. But b and f must belong to different clusters.

Density-based Methods

A cluster in density-based clustering is a region with a high density of points surrounded by a low density region. DBSCAN, a density-based algorithm introduced by Ester et al. [1996], produces a partitional clustering and defines a cluster as a continuous area of any shape with greater density than its surroundings. The algorithm scans the data points in a

dataset, computes neighborhoods with a defined radius and minimum number of points, and connects these dense neighborhoods to form clusters. A neighborhood with a defined radius and minimum number of points is referred to as a core point, while a data point without such a neighborhood is either considered a noise point or a border point if it is in the same neighborhood as a core point. The radius ε and the minimum number of points $MinPts$ serve as thresholds for determining the density of a neighborhood.

DENGRAPH is a graph clustering algorithm based on density, developed by Falkowski et al. [2007] to identify groups of similar nodes in graphs that contain numerous noise objects. Clusters in the graph are areas where nodes are densely packed and separated by low node density regions. DENGRAPH calculates neighborhoods by utilizing a specific radius (ε) and a minimum number of nodes ($MinPts$) to ensure that they are dense. A node with a neighborhood of this type is referred to as a core node. Nodes lacking such a neighborhood are classified as either border nodes if they are within a core node's neighborhood or noise nodes.

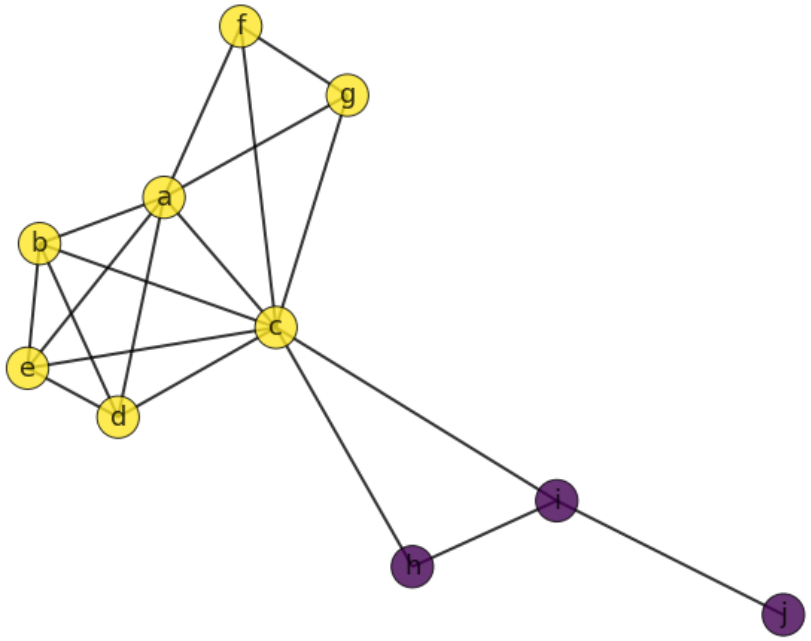


Fig. 3.5 DBSCAN clustering of data objects ($\varepsilon = 0.3, minPts = 3$).

To group similar data vertices, a measure of similarity or dissimilarity is needed. The similarity between two vertices is quantified by a numerical measure of how similar they are, and dissimilarity is quantified by the numerical difference between two objects. The higher the similarity, the more alike the objects are and the lower the dissimilarity. When referring to dissimilarity, the term "distance" is often used.

Figure 3.5 shows two clusters $\{a,b,c,d,e,f,g\}$ and $\{h,i,j\}$ by applying DENGRAPH. Unfortunately, the triples for example, $\{b,a,f\}$ and $\{h,i,j\}$ violates TI property. Thus may compromises the quality of resulting clusters.

Clique-based Methods

A clique is defined as a complete network on a set of nodes, with each pair of nodes connected by an edge in graph theory. If a clique can't be extended to a larger clique by including any nearby node, it's called maximal. Each community in a network should be a maximal clique in the ideal clustering outcome. In practice, this is difficult to achieve. It's considerably more difficult to guarantee that each community forms a clique in many real-world social networks. As a result, an acceptable measure for assessing a community's degree of connectivity is required.

Bron and Kerbosch [1973] presented a depth-first search algorithm for generating all the maximal cliques of an undirected graph as shown in Algorithm 2. That is, it lists all subsets of vertices with the two properties that each pair of vertices in one of the listed subsets is connected by an edge, and no listed subset can have any additional vertices added to it while preserving its complete connectivity.

Algorithm 2: Algorithm for searching all maximal cliques.

```
clique(P,R,X)
1 if P ∪ X = ∅ then
2 |   report R as a maximal clique
3 end
4 choose a pivot u ∈ P ∪ X to maximize |P ∩ N(u)|
5 for each vertex v ∈ P \ N(u) do
6 |   clique(P ∩ N(v), R ∪ {v}, X ∩ N(v))
7 |   P ← P \ {v}
8 |   X ← X ∪ {v}
9 end
```

The Bron–Kerbosch algorithm is a simple recursive algorithm that maintains three sets of vertices: a partial clique R, a set of candidates for clique expansion P, and a set of forbidden vertices X. In each recursive call, a vertex v from P is added to the partial clique R, and the sets of candidates for expansion and forbidden vertices are restricted to include only neighbors of v. If $P \cup X$ becomes empty, the algorithm reports R as a maximal clique, otherwise the algorithm chooses a vertex u in $P \cup X$ called a *pivot*. All maximal cliques must contain a non-neighbor of u (counting u itself as a non-neighbor), and therefore, the recursive calls can be restricted to the intersection of P with the non-neighbors.

Example 3.4. Consider the social network of ten objects shown in Figure 3.1. Algorithm 2 generates all maximal cliques $\{a,b,c,d,e\}$, $\{a,c,f,g\}$, $\{c,h,i\}$ and $\{i,j\}$ shown in Figure 3.6 where intra-cluster similarities of objects are higher and each cluster capture the natural structure of objects that reflects their relationship.

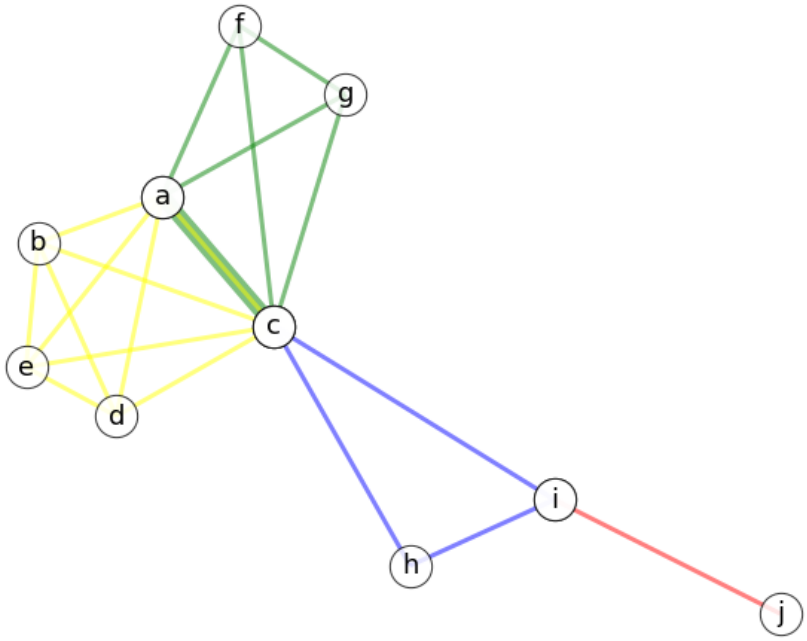


Fig. 3.6 Clique based clustering of data objects $\{a,b,c,d,e,f,g,h,i,j\}$.

Computing both the pivot and the vertex sets for the recursive calls can be done in time $O(|P| \cdot (|P| + |X|))$ within each call to the algorithm, using an adjacency matrix to quickly test the adjacency of pairs of vertices. This pivoting strategy, together with this adjacency-matrix-based method for computing the pivots, leads to a worst case time bound of $O(3^{n/3})$ for listing all maximal cliques.

3.5 Summary

To analyze experimental data in a variety of scientific disciplines, a large collection of clustering algorithms is available. In the scientific literature, new clustering programs are constantly appearing. The majority of these algorithms, however, are based on two popular clustering techniques: partitional clustering and agglomerative hierarchical clustering.

This chapter focused on the theoretical concepts of different clustering techniques and the measures used in these techniques: distance measure and similarity measure. It explained

the violation of TI for detecting clusters in a social network using traditional clustering techniques and finally described a clique-guided approach to detect meaningful clusters.

Chapter 4

A Quantum Logic-based Model for Non-TI Clustering

The origins of quantum mechanics (QM) can be traced back to the early twentieth century. Physicists such as Einstein, Planck, Bohr, Schrödinger, and Heisenberg had a significant impact on the theory. It is concerned with elementary particle phenomena such as measurement uncertainty in closed microscopic physical systems and entangled states. In recent years, computer scientists have become interested in quantum mechanics as they attempt to use its power to solve computationally difficult problems.

Quantum logic, first developed by Birkhoff and Neumann [1936], is an appealing part of the mathematical formalism of quantum mechanics. Quantum logic is an unconventional logic that is based on projectors of a complex separable Hilbert space. Many mathematicians have investigated quantum logic's properties. A concise summary of the most important findings is given in Beltrametti and Fraassen [1981], Lock and Hardegree [1985a], Lock and Hardegree [1985b], and Ziegler [2005].

Schmitt [2008] developed the Commuting Quantum Query Language (CQQL), a query language that is rooted in the mathematical formalism of quantum mechanics. This language is used in this thesis as the query language for non-TI clustering.

Section 4.1 in this thesis explains how Schmitt [2008] defines and employs the fundamental principles of quantum mechanics to develop a retrieval model resulting in CQQL. Section 4.2 then presents CQQL and its formal evaluation procedure. The use of weights to customize CQQL queries is discussed in Section 4.3. The capabilities of CQQL as a multimodal logical query language are demonstrated through an example in Section 4.4. Section 4.5 illustrates how CQQL can be viewed as a non-TI clustering implementation, with the help of a flow diagram. Finally, the chapter concludes in Section 4.6 with a discussion of clustering properties.

4.1 Theoretical Concept of Commuting Quantum Query Language

Traditional database systems based on Boolean logic typically evaluate each database object to determine if it is part of the result set, returning a Boolean value of *true* for a perfect match and *false* otherwise (ignoring the *unknown* case). However, this approach fails to meet user requirements in text retrieval searches, where a complete match is often impossible or when vague aspects are part of the query. To address these issues, various query languages have been proposed that incorporate vague or imprecise predicates through the use of fuzzy logic. Fuzzy set theory and its derivatives have been applied in various areas of information retrieval (IR).

A Fuzzy set consists of a set of ordered pairs, the first element of which denotes an element of a predefined universe and the second the degree of membership of that element as value from interval $[0,1]$. This membership values μ can be interpreted as similarity values in the IR domain. But fuzzy set theory suffers from weaknesses in the context of IR.

First, Fuzzy set theory only deals with membership values, not considering the origin or meaning of these values (such as the evaluation of a database or information retrieval condition). Hence, the entire theory is built solely on the basis of membership values.

Second, the dominance problem affects some fuzzy set operations such as union ($\cup$), intersection ($\cap$) and complement ($\neg$). For instance, if X is a set of objects represented by x and $\mu(x)$ is its membership function, Zadeh [1965] suggests using the *min* function for intersection, *max* function for union, and $1-\mu(x)$ for complement. However, the dominance problem arises with the use of the *min* function, which only returns the minimum of two values and ignores the larger value, as well as their absolute difference. This can lead to unexpected results, as demonstrated in [Lee et al. 1994]. For example, $min(0.1,0.9)$ and $min(0.1,0.2)$ both return 0.1, neglecting the significant difference in similarity between the two cases. As a result, one value dominates the other, losing potentially valuable information.

Third, not all fuzzy set operations obey the laws of Boolean algebra. Although min/max are idempotent, they violate the complement axiom, as follows:

For example, for $\mu(x) = 0.5$,

$$\mu_{x\wedge\neg x}(x) = min(\mu(x), 1-\mu(x)) = min(0.5,0.5) = 0.5 \neq 0$$

Substituting min/max by other `t-norms/t-conorms` cause other violations of the Boolean algebra described by Fagin and Wimmers [2000]. One alternative to min/max is to express the intersection by the algebraic product $(x \cdot y)$ and the union by the algebraic

sum $(x+y-x\cdot y)$. By involving both values in the evaluation, this function set overcomes the dominance problem. It equals the CQQL formula presented by Schmitt [2008] but is not idempotent in the case of fuzzy logic because the fuzzy model does not account for the properties of the projector lattice:

$$x \wedge x \rightsquigarrow x \cdot x = x^2 \neq x$$

The CQQL query language offers the ability to calculate the similarity between two objects by incorporating both Boolean and similarity conditions into logical-based queries. It also enables the addition of weighting to the query. The similarity values used in the queries typically range from 0 to 1 and are used to determine the proximity of attribute values between the two objects being compared.

Schmitt's [2008] works are based on the mathematical formalism of quantum mechanics, but this dissertation does not intend to provide a comprehensive introduction or discussion of QM. It only provides a general overview of the core concepts necessary to understand CQQL. For a more in-depth definition and understanding of the theoretical foundations of CQQL, it is recommended to refer to Schmitt [2008].

The Dirac (or bra-ket) notation, which is explained in Section 2.4, is used in all definitions and throughout the rest of the thesis. The formalism of quantum mechanics deals with vectors of a complex separable Hilbert space H. Schmitt's [2008] presented four postulates sketched below:

"Postulate 1: Every closed physical microscopic system corresponds to a separable complex Hilbert space and every state of the system is completely described by a ket vector $|\varphi\rangle$ of length one.

Postulate 2: Every evolution of a state $|\varphi\rangle$ can be represented by the product of $|\varphi\rangle$ and an orthonormal operator.

Postulate 3: This postulate describes how to measure a state, which entails calculating the probability of various outcomes. When a specific outcome is measured, the system switches to that mode automatically. Here, we focus on a simplified measurement provided by projectors (each one represents one possible outcome and is bijectively associated with one vector subspace). The probability of an outcome associated with a projector p and a state $|\varphi\rangle$ is defined by

$\langle\varphi|p|\varphi\rangle = \langle\varphi| \left(\sum_i |i\rangle \langle i| \right) |\varphi\rangle = \sum_i \langle\varphi|i\rangle \langle i|\varphi\rangle$

Thus, the probability value equals the squared length of the state vector $|\varphi\rangle$ after its projection onto the subspace spanned by the vectors $|i\rangle$. Due to normalization, the probability

value, furthermore, equals geometrically the squared cosine of the minimal angle between $|\varphi\rangle$ and the subspace represented by p.

Postulate 4: This postulate defines how to assemble various quantum systems into one system. The base vectors of the composed system are constructed by applying the tensor product $'\otimes'$ on the subsystems base vectors."

[Schmitt 2008]

The core concept of using quantum mechanics for database querying involves representing objects as state vectors and queries as projectors within a carefully crafted vector space. In this setup, a state vector, serving as a database object, holds all possible measurement outcomes, while queries, represented as projectors, specify subspaces. Table 4.1 shows how retrieval concepts are related to quantum mechanics concepts.

Table 4.1 Concepts that are associated with database querying and quantum mechanics

Database querying	**Quantum mechanics**
Database tuple	State vector
Query	Projector
Query processing	Quantum measurement
Truth values	Probability values
Boolean logic	Quantum logic

CQQL, being a quantum logic-based query language, requires an understanding of quantum logic (QL). QL was first introduced by Neumann [1932] and Birkhoff and Neumann [1936] and is a logic used to analyze formal structures within the Hilbert space.

Let P be the set of all projectors of a vector space H of dimensions greater than two. Each projector $p \in P$ is bijectively related to a closed subspace via $p(H) = \{p|\varphi\rangle \,||\varphi\rangle \in H\}$. The subset relation $p_1(H) \subseteq p_2(H)$ on P which is equivalent to $p_2p_1 = p_1p_2 = p_1$ forms a complete poset. Furthermore, we obtain a lattice with the binary operations meet $(\wedge)$ and join $(\vee)$ being defined as intersection of the corresponding subspaces and linear hull of vectors of their union

$$p_{p_1}(H) \wedge p_{p_2}(H) \equiv p_{p_1(H)\cap p_2(H)}$$
$$p_{p_1}(H) \vee p_{p_2}(H) \equiv p_{closure(p_1(H)\cup p_2(H))}$$

where $closure()$ yields the set of all possible vector linear combinations.

The negation (orthocomplement) for our quantum logic is defined as $\neg p \equiv I - p$ encompassing all projectors being orthogonal to p.

Quantum logic violates the distributive law and therefore cannot form a Boolean algebra, which is required for CQQL to be relationally complete. This is proven in Schmitt [2008]. To address this issue, it is necessary to identify a sublattice of quantum logic that complies with Boolean algebra laws, which can be achieved by considering commuting projectors.

Definition 4.1 (commuting projectors). "Two projectors p_1 and p_2 of a vector space H are called commuting projectors if and only if $p_1 p_2 = p_2 p_1$ holds.

From linear algebra we know that two projectors $p_1 = \sum_i |i\rangle \langle i|$ and $p_2 = \sum_j |j\rangle \langle j|$ commute if and only if their ket vectors $|i\rangle$ and $|j\rangle$ are basis vectors of the same orthonormal basis of the underlying vector space. In that case, we can write $p_1 = \sum_{i_1} |k_{i_1}\rangle \langle k_{i_1}|$ and $p_2 = \sum_{i_2} |k_{i_2}\rangle \langle k_{i_2}|$ where the ket vectors $|k_i\rangle$ form an orthonormal basis. If two projectors commute then their join corresponds to the union of the respective one-dimensional operators $\{|k_{i_j}\rangle \langle k_{i_j}|\}$ and their meet to their intersection. Thus, all projectors over a given orthonormal basis form a Boolean algebra."

[Schmitt 2008]

To summarize, QL is a generalization of Boolean algebra that violates the distributivity law. As a result, QL's expressive power must be limited to only allow commuting projectors - hence Schmitt's proposed query language's name: the commuting quantum query language (CQQL).

This dissertation focuses on the evaluation rules of CQQL and how the query language can be used, rather than delving into the details of how CQQL is derived from QL or the mapping of database or retrieval conditions to state vectors. It is recommended to refer to Schmitt [2008] (the central work on CQQL and its origin in QL) for a complete understanding of the theoretical aspects. To continue with the thesis, it is only necessary to have an understanding of the summarized comparisons presented in Table 4.1.

4.2 Construction and Arithmetic Evaluation of CQQL

This section of the dissertation sketches the construction and evaluation of CQQL based on the discussion presented in Schmitt [2008].

Definition 4.2 (Basic CQQL elements). "Let $A = \{a_j\}$ be a finite set of attributes and $AC = \{ac_i(a_j)\}$ be a finite set of atomic attribute conditions of the form:

- $'a_j = value'$ (Boolean or database condition), or
- $'a_j \approx value'$ (retrieval or proximity condition).

Let furthermore be given a function $vs(ac_i(a_j)) = \{ac_1^i, ..., ac_k^i\}$ which assigns to every condition a set of orthonormal vectors forming a vector subspace. The condition set AC is called commutative if $\forall ac_{i_1}(a_{j_1}), ac_{i_2}(a_{j_2}) \in AC : (type(ac_{i_1}(a_{j_1})) \neq d \wedge type(ac_{i_2}(a_{j_2})) \neq d \wedge j_1 = j_2) \implies i_1 = i_2$ holds where type returns d for a database condition and p or r for a proximity or retrieval condition, respectively.

Commutativity means that no two proximity or retrieval conditions $'a_j \approx value_1'$ and $'a_j \approx value_2'$' with $value_1 \neq value_2$ on the same attribute a_j are allowed.

Lemma 4.1. Let AC be a commutative set of atomic conditions over $A = \{a_j\}$. The set $CVS(AC) = \bigcup_{ac_i(a_j) \in AC} vs(ac_i(a_j))$ is a set of mutually orthonormal projectors.

The lemma is a direct consequence of how the mapping function *vs* is realized (for more details see Schmitt [2008]). It is very essential because it means that every subset of $CVS(AC)$ spans a vector subspace and corresponds bijectively to a CQQL condition."

[Schmitt et al. 2008]

Definition 4.3 (CQQL conditions). Let AC be a set of atomic conditions on database tuples (objects). In this set, there is a constraint that no two distinct similarity conditions are expressed concerning the same attribute. Each $ac(o_1, o_2) \in AC$ is a binary condition on two objects that measures the closeness of two given objects based on the values of an attribute. A CQQL condition $\varphi(o_1, o_2)$ on an object pair (o_1, o_2) is recursively defined by

$$\begin{aligned}
\varphi(o_1, o_2) &\stackrel{def}{=} ac(o_1, o_2) \in AC, \\
\varphi(o_1, o_2) &\stackrel{def}{=} (\varphi_1(o_1, o_2) \wedge \varphi_2(o_1, o_2)), \\
\varphi(o_1, o_2) &\stackrel{def}{=} (\varphi_1(o_1, o_2) \vee \varphi_2(o_1, o_2)), \\
\varphi(o_1, o_2) &\stackrel{def}{=} (\neg\varphi_1(o_1, o_2))
\end{aligned}$$

where φ_1, φ_2 are CQQL conditions with

$$\begin{aligned}
vs(\varphi_1 \wedge \varphi_2) &= vs(\varphi_1) \cap vs(\varphi_2) \subseteq CVS(AC) \\
vs(\varphi_1 \vee \varphi_2) &= vs(\varphi_1) \cup vs(\varphi_2) \subseteq CVS(AC) \\
vs(\neg\varphi_1) &= CVS(AC) \setminus vs(\varphi_1) \subseteq CVS(AC)
\end{aligned}$$

Theorem 4.1. "All CQQL conditions over a commutative set of atomic conditions together with conjunction, disjunction, and negation form a boolean algebra.

Proof. The function *vs* maps bijectively every condition to a subset of $CVS(AC)$. Conjunction, disjunction, and negation are mapped to corresponding set operations. A set together with these standard set operations is a boolean algebra." □

[Schmitt 2008]

Arithmetic Evaluation of CQQL

"A CQQL condition can be evaluated using simple and straightforward arithmetic" [Schmitt 2008] (see Figure 4.1). The necessary syntax normalization algorithm has been omitted here for the sake of simplicity but described with an example in Appendix B. Schmitt [2008] provide a comprehensive description of the algorithm. Because CQQL follows the laws of Boolean algebra, the operations carried out by the algorithm can be easily understood. Therefore, all CQQL conditions can be transformed into the necessary syntactical form, making it possible to apply Boolean transformation rules.

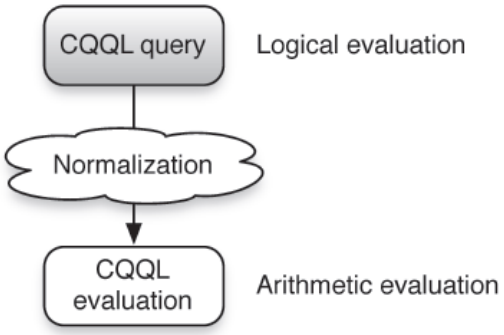


Fig. 4.1 Transformation of CQQL queries.

Definition 4.4 (CQQL evaluation). Let $eval(\varphi(o_1,o_2))$ be the evaluation of a condition φ on two objects o_1 and o_2 in a special syntactical normal form. If φ is an atomic condition, $eval(\varphi(o_1,o_2))$ forms the base case for the evaluation of a condition and results either a Boolean or similarity value for the pair of objects, o_1 and o_2, i.e.,

- for a Boolean or database condition, where 0 means $false$ and 1 refers to $true$:

$$eval(\varphi(o_1,o_2)) \rightarrow \{0,1\}$$

- for a retrieval or similarity condition, where 0 indicates maximum dissimilarity and 1 maximum similarity:

$$eval(\varphi(o_1,o_2)) \rightarrow [0,1]$$

which fulfils the following properties:

- *Reflexivity:* $\varphi(o_1, o_1) = 1$
- *Symmetric:* $\varphi(o_1, o_2) = \varphi(o_2, o_1)$.

Subsequently, the evaluation of a CQQL condition in a special syntactical normal form is performed by recursively applying the succeeding formulas until the base case is reached:

$$eval(\varphi_1 \wedge \varphi_2) = eval(\varphi_1) \cdot eval(\varphi_2) \tag{4.1}$$
$$eval(\varphi_1 \vee \varphi_2) = eval(\varphi_1) + eval(\varphi_2) - eval(\varphi_1) \cdot eval(\varphi_2) \tag{4.2}$$
$$eval(\varphi_1 \dot{\vee} \varphi_2) = eval(\varphi_1) + eval(\varphi_2) \tag{4.3}$$
$$eval(\neg\varphi_1) = 1 - eval(\varphi_1) \tag{4.4}$$

For simplification, the objects o_1 and o_2 are dropped here. The dot (.) over disjunction operator ($\vee$) signals an exclusive disjunction ($\dot{\vee}$).

Equation 1 and 2 applies when φ_1 and φ_2 are not exclusive and equation 3 applies when φ_1 and φ_2 are exclusive (i.e. $eval(\varphi_1 \wedge \varphi_2) = 0$).

An example demonstrating the application of CQQL as a query language for social network clustering will be provided in Section 4.5 after discussing another aspect of the language, which is the integration of weighted logical connectors.

4.3 Weighting of CQQL Conditions

CQQL uses weights to reflect the varying significance of sub-conditions within a query, while maintaining consistency with Boolean algebra. The weights $\Theta = \theta_i$ are considered to be values within the range of $[0, 1]$. These weights allow for the sub-conditions in a conjunction or disjunction to have varying levels of influence on the outcome of the evaluation.

The idea behind weighted CQQL queries is straightforward: every logical connector is assigned a weight to regulate its operand's impact on the evaluation outcome. To convert this weighted query into a weightless logical formula, Schmitt [2019] proposes transforming the weighting variables $\theta_i \in [0, 1]$ attached to the logical connectors into constants. When $\theta_i = 0$, the operand has no effect on the result, and when $\theta_i = 1$, the operand behaves as in an unweighted scenario. The formalization of weighted CQQL queries q^{Θ} can be done by following the cases:

- Weights based on the Influence of Objects

- Complimented Form of a Weight

The details of these cases are discussed in the following subsections.

4.3.1 Weights based on the Influence of Objects

The weight θ_i regulates the impact of a condition φ_i for a weighted conjunction $\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2$ and an analogous disjunction $\varphi_1 \vee_{\theta_1,\theta_2} \varphi_2$. The core idea of this approach is the direct transformation of a weighted conjunction or disjunction into a logical expression in which weight values are converted into score values:

$$\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2 = (\varphi_1 \vee \neg\theta_1) \wedge (\varphi_2 \vee \neg\theta_2) \tag{4.5}$$

$$\varphi_1 \vee_{\theta_1,\theta_2} \varphi_2 = (\varphi_1 \wedge \theta_1) \vee (\varphi_2 \wedge \theta_2) \tag{4.6}$$

Therefore the arithmetic evaluation of the weighted conjunction and weighted disjunction is:

$$eval(\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2) = (\varphi_1 + \neg\theta_1 - \varphi_1 \cdot \neg\theta_1) \cdot (\varphi_2 + \neg\theta_2 - \varphi_2 \cdot \neg\theta_2) \tag{4.7}$$

$$eval(\varphi_1 \vee_{\theta_1,\theta_2} \varphi_2) = (\varphi_1 \cdot \theta_1) + (\varphi_2 \cdot \theta_2) - (\varphi_1 \cdot \varphi_2 \cdot \theta_1 \cdot \theta_2) \tag{4.8}$$

where $\neg\theta_x = 1 - \theta_x$.

Table 4.2 presents the "truth" table for the weighted conjunction and disjunction under various weight configurations.

Table 4.2 Impact of Weights in CQQL

φ_1	φ_2	θ_1	θ_2	$\varphi_1 \vee \neg\theta_1$	$\varphi_2 \vee \neg\theta_2$	$\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2$	$\varphi_1 \wedge \theta_1$	$\varphi_2 \wedge \theta_2$	$\varphi_1 \vee_{\theta_1,\theta_2} \varphi_2$
0.3	0.7	0.0	0.0	1.0	1.0	1.00	0.0	0.0	0.00
		0.0	1.0	1.0	0.7	0.70	0.0	0.7	0.70
		1.0	0.0	0.3	1.0	0.30	0.3	0.0	0.30
		1.0	1.0	0.3	0.7	0.21	0.3	0.7	0.79

For a logical connector of any arbitrary n-ary, the concept is generalized by associating the operands with the respective weight constants. Thus, a ternary conjunction $\wedge_{\theta_1,\theta_2,\theta_3}(\varphi_1,\varphi_2,\varphi_3)$ is transformed into:

$$\wedge_{\theta_1,\theta_2,\theta_3}(\varphi_1,\varphi_2,\varphi_3) = (\varphi_1 \vee \neg\theta_1) \wedge (\varphi_2 \vee \neg\theta_2) \wedge (\varphi_3 \vee \neg\theta_3) \tag{4.9}$$

$$\wedge_{\theta_1,\theta_2,\theta_3}(\varphi_1,\varphi_2,\varphi_3) = (\varphi_1 + \neg\theta_1 - \varphi_1 \cdot \neg\theta_1) \cdot (\varphi_2 + \neg\theta_2 - \varphi_2 \cdot \neg\theta_2) \cdot (\varphi_3 + \neg\theta_3 - \varphi_3 \cdot \neg\theta_3) \tag{4.10}$$

The disjunction is treated analogously:

$$\vee_{\theta_1,\theta_2,\theta_3}(\varphi_1,\varphi_2,\varphi_3) = (\varphi_1 \wedge \theta_1) \vee (\varphi_2 \wedge \theta_2) \vee (\varphi_3 \wedge \theta_3) \tag{4.11}$$

Equation 4.10 suggests the possible intricacy of the evaluation that results from the rules presented earlier. The formula includes a non-exclusive disjunction, which increases the computational cost of the evaluation due to the arithmetic sum $(a+b-a\cdot b)$ (as shown in Equation 4.2).

Fortunately, adding weights to CQQL does not violate its Boolean algebra property (proof can be found in Schmitt 2019). This means that all Boolean transformation rules hold and can be employed to simplify complex logical statements, as demonstrated in an example:

$$\begin{array}{c} (\varphi_1 \wedge \theta_1) \vee (\varphi_2 \wedge \theta_2) \vee (\varphi_3 \wedge \theta_3) \\ \Updownarrow \text{ Double negation} \\ \overline{\overline{(\varphi_1 \wedge \theta_1) \vee (\varphi_2 \wedge \theta_2) \vee (\varphi_3 \wedge \theta_3)}} \\ \Updownarrow \text{ De Morgan's law} \\ \overline{\overline{(\varphi_1 \wedge \theta_1)} \wedge \overline{(\varphi_2 \wedge \theta_2)} \wedge \overline{(\varphi_3 \wedge \theta_3)}} \\ \Updownarrow \text{ Arithmetic evaluation} \\ 1-((1-(\varphi_1\cdot\theta_1))\cdot(1-(\varphi_2\cdot\theta_2))\cdot(1-(\varphi_3\cdot\theta_3))) \end{array}$$

Therefore the arithmetic evaluation of the ternary disjunction of Equation 4.11 is:

$$eval(\vee_{\theta_1,\theta_2,\theta_3}(\varphi_1,\varphi_2,\varphi_3)) = 1-((1-(\varphi_1\cdot\theta_1))\cdot(1-(\varphi_2\cdot\theta_2))\cdot(1-(\varphi_3\cdot\theta_3))) \tag{4.12}$$

4.3.2 Complimented Form of a Weight

We present an interesting special case of a weighting that requires fewer weighting variables compared with the previous weighting concept. It is simply a complemented form of a weight one against another:

$$\varphi_1 \wedge_\theta \varphi_2 = (\varphi_1 \vee \neg\theta) \wedge (\varphi_2 \vee \theta) \tag{4.13}$$

$$\varphi_1 \vee_\theta \varphi_2 = (\varphi_1 \wedge \theta) \dot{\vee} (\varphi_2 \wedge \neg\theta) \tag{4.14}$$

After undergoing a logical normalization process to achieve a special syntactical normal form, the weighted conjunction or disjunction will be evaluated between two objects o_1 and o_2 using basic arithmetic operations like addition, subtraction, and multiplication. The

evaluation formula of the weighted disjunction is the weighted sum that results from this process.

$$eval(\varphi_1 \dot{\vee}_\theta \varphi_2) = \varphi_1 \cdot \theta + \varphi_2 \cdot (1 - \theta) \tag{4.15}$$

Very surprisingly, logical transformation based on Boolean algebra shows that connected weights in a conjunction equals exactly connected weights in a disjunction.

$$\begin{gathered}
(\varphi_1 \vee \neg\theta) \wedge (\varphi_2 \vee \theta) \\
\downarrow \\
(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \theta) \vee (\varphi_2 \wedge \neg\theta) \vee (\theta \wedge \neg\theta) \\
\downarrow \\
(\varphi_1 \wedge \varphi_2 \wedge \theta) \vee (\varphi_1 \wedge \varphi_2 \wedge \neg\theta) \vee (\varphi_1 \wedge \theta) \vee (\varphi_2 \wedge \neg\theta) \\
\downarrow \\
(((\varphi_1 \wedge \varphi_2) \vee \varphi_1) \wedge \theta) \dot{\vee} (((\varphi_1 \wedge \varphi_2) \vee \varphi_2) \wedge \neg\theta) \\
\downarrow \\
(\varphi_1 \wedge \theta) \dot{\vee} (\varphi_2 \wedge \neg\theta)
\end{gathered}$$

Therefore the arithmetic evaluation of the weighted conjunction is also the weighted sum:

$$eval(\varphi_1 \wedge_\theta \varphi_2) = \varphi_1 \cdot \theta + \varphi_2 \cdot (1 - \theta)" \tag{4.16}$$

Schmitt et al. [2008] summarizes the weighting approach, which shows the following properties:

1. An operand with zero weight has no impact on the result.
2. An operand with a weight of one behaves equivalently to an unweighted operand.
3. Weighting is achieved through conjunction, disjunction, and negation, ensuring that all the rules of Boolean algebra remain applicable. In contrast, many other weighting methods, like the one proposed by Fagin and Wimmers [2000], involve arithmetic operations outside the logical framework, thereby contradicting the laws of Boolean algebra.

4.4 CQQL: A Logical Query Language with Multiple Modes

As previously mentioned, CQQL offers a way to merge different retrieval approaches and weights to express the significance of various conditions (which is subjective). This section illustrates how CQQL can be employed in Non-TI clustering through the use of an example.

Example 4.1. In a social network comprising former students of a school, consider a set of n people denoted as $P = p_1, p_2, ..., p_n$. The degree of similarity among these people can effectively be expressed by considering their respective admission and graduation years.

The similarity value between two people p_1 and p_2 can be expressed as a CQQL condition q as:

$$q(p_1,p_2) = school(p_1,p_2) \wedge \underbrace{\Big(graduation(p_1,p_2) \wedge admission(p_1,p_2)\Big)}_{conjunction} \tag{4.17}$$

Alternatively

$$q(p_1,p_2) = school(p_1,p_2) \wedge \underbrace{\Big(graduation(p_1,p_2) \vee admission(p_1,p_2)\Big)}_{disjunction} \tag{4.18}$$

where

$$\begin{aligned} school(p_1,p_2) &= (p_1.school = p_2.school) \\ graduation(p_1,p_2) &= (p_1.graduation \approx p_2.graduation) \\ admission(p_1,p_2) &= (p_1.admission \approx p_2.admission) \end{aligned}$$

The similarity between two people can be effectively represented by considering both their admission year and graduation year. However, it's worth noting that friends can complete their degrees in different graduating classes while still sharing the same admission year. Due to this variation in graduation years, the impact of their respective similarity values on the final result should be relatively minor. Therefore, the CQQL query $q(p_1,p_2)$ can be written as a weighted CQQL query $q^{\Theta}(p_1,p_2)$ with weight variables $\Theta = \{\theta_i\}$. A weighting function w assigns every weight variable $\theta_i \in \Theta$ a value from $[0,1]$.

Weighted conjunction

1. **Complimented form of a weight**
   Following Schmitt [2019], we reformulate the query q in Equation 4.17 by assigning a weight to each atomic condition in a complimented form of one against other.

$$q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta} admission(p_1,p_2)\Big) \quad (4.19)$$

To perform the evaluation of this query, the weighted conjunction needs to undergo transformation following the rules outlined in Section 4.3:

$$\begin{gathered}(p_1.school = p_2.school) \wedge \\ ((p_1.graduation \approx p_2.graduation \vee \neg\theta) \wedge (p_1.admission \approx p_2.admission \vee \theta)) \\ \Updownarrow \\ (p_1.school = p_2.school) \wedge \\ ((p_1.graduation \approx p_2.graduation \wedge \theta) \dot{\vee} (p_1.admission \approx p_2.admission \wedge \neg\theta))\end{gathered}$$

To facilitate reading, the query will be subdivided into parts that are evaluated separately.

$$\underbrace{\underbrace{(p_1.school = p_2.school)}_{school} \wedge}_{db\_part}$$
$$\underbrace{\underbrace{\underbrace{(p_1.graduation \approx p_2.graduation}_{graduation} \wedge \theta)}_{weighted\_graduation} \dot{\vee} \underbrace{\underbrace{(p_1.admission \approx p_2.admission}_{admission} \wedge \neg\theta)}_{weighted\_admission}}_{retrieval\_part}$$

Using Equation 4.16 the arithmetic evaluation of $q^{\Theta}(p_1,p_2)$ yields:

$$\begin{aligned} weighted\_graduation &\leftarrow graduation \cdot \theta \in [0,1] \\ weighted\_admission &\leftarrow admission \cdot (1-\theta) \in [0,1] \\ retrieval\_part &\leftarrow weighted\_graduation + weighted\_admission \in [0,1] \\ db\_part &\leftarrow school \in \{0,1\} \end{aligned}$$

$$\Downarrow$$

$$db\_part \cdot retrieval\_part \in [0,1]$$

2. **Weights based on Influence**
   we reformulate the query q in Equation 4.17 by assigning weights to each atomic condition.

$$q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta_1,\theta_2} admission(p_1,p_2)\Big) \quad (4.20)$$

To perform the evaluation of this query, the weighted conjunction needs to undergo transformation following the rules outlined in Section 4.3:

$$(p_1.school = p_2.school) \wedge$$
$$((p_1.graduation \approx p_2.graduation \vee \neg\theta_1) \wedge (p_1.admission \approx p_2.admission \vee \neg\theta_2))$$

To facilitate reading, the query will be subdivided into parts that are evaluated separately.

$$\underbrace{\underbrace{(p_1.school = p_2.school)}_{school}\wedge}_{db\_part}$$
$$\underbrace{\underbrace{(\underbrace{(p_1.graduation \approx p_2.graduation}_{graduation} \vee \neg\theta_1)}_{weighted\_graduation} \wedge \underbrace{(p_1.admission \approx p_2.admission}_{admission} \vee \neg\theta_2))}_{weighted\_admission}}_{retrieval\_part}$$

Using Equation 4.16 the arithmetic evaluation of $q^{\Theta}(p_1, p_2)$ yields:

$$\begin{aligned} weighted\_graduation &\leftarrow \Big(graduation + (1-\theta_1) - graduation \cdot (1-\theta_1)\Big) \in [0,1] \\ weighted\_admission &\leftarrow \Big(admission + (1-\theta_2) - admission \cdot (1-\theta_2)\Big) \in [0,1] \\ retrieval\_part &\leftarrow weighted\_graduation \cdot weighted\_admission \in [0,1] \\ db\_part &\leftarrow school \in \{0,1\} \end{aligned}$$

$$\Downarrow$$

$$db\_part \cdot retrieval\_part \in [0,1]$$

Weighted disjunction

1. **Complimented form of a weight**
   We reformulate the query q in Equation 4.18 by assigning a weight in a disjunction to each atomic condition in a complimented form of one against other.

$$q^{\Theta}(p_1, p_2) = school(p_1, p_2) \wedge \Big(graduation(p_1, p_2) \,\dot{\vee}_{\theta}\, admission(p_1, p_2)\Big) \quad (4.21)$$

But the logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. That means,

$$school(p_1,p_2) \wedge (graduation(p_1,p_2) \wedge_\theta admission(p_1,p_2)) = school(p_1,p_2) \wedge (graduation(p_1,p_2) \dot{\vee}_\theta admission(p_1,p_2))$$

Therefore, the formulation and evaluation are the same as described in 4.4.

2. **Weights based on Influence**
   For more than one weighting variable, we reformulate the query q in Equation 4.18 by assigning weights to each atomic condition.

$$q^\Theta(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \vee_{\theta_1,\theta_2} admission(p_1,p_2)\Big) \quad (4.22)$$

To perform the evaluation of this query, the weighted conjunction needs to undergo transformation following the rules outlined in Section 4.3:

$$(p_1.school = p_2.school) \wedge ((p_1.graduation \approx p_2.graduation \wedge \theta_1) \vee (p_1.admission \approx p_2.admission \wedge \theta_2))$$

To facilitate reading, the query will be subdivided into parts that are evaluated separately.

$$\underbrace{\underbrace{(p_1.school = p_2.school)}_{school} \wedge}_{db\_part}$$
$$\underbrace{\underbrace{\underbrace{(p_1.graduation \approx p_2.graduation}_{graduation} \wedge \theta_1)}_{weighted\_graduation} \vee \underbrace{\underbrace{(p_1.admission \approx p_2.admission}_{admission} \wedge \theta_2)}_{weighted\_admission}}_{retrieval\_part}$$

Using Equation 4.16 the arithmetic evaluation of $q^\Theta(p_1,p_2)$ yields:

$$\begin{aligned} db\_part &\leftarrow school \in \{0,1\} \\ weighted\_graduation &\leftarrow \Big(graduation \cdot \theta_1\Big) \in [0,1] \\ weighted\_admission &\leftarrow \Big(admission \cdot \theta_2\Big) \in [0,1] \\ retrieval\_part &\leftarrow \Big(weighted\_graduation + weighted\_admission \\ &\qquad - weighted\_graduation \cdot weighted\_admission\Big) \in [0,1] \end{aligned}$$

$$\Downarrow$$

$$db\_part \cdot retrieval\_part \in [0,1]$$

The query condition is executed on $P \times P$. The evaluation of $q^{\Theta}(p_1, p_2)$ is realized by the function $eval(q^{\Theta}(p_i, p_j), w)$ (see Definition 4.4) which calculates a similarity value from $[0,1]$ for every pair of objects $(p_i, p_j) \in P \times P$ and hence provides a similarity matrix $S = (s_{ij})$.

4.5 Non-TI Clustering Approach

The flowchart in Figure 4.2 depicts our approach for discovering clusters based on similarity values.

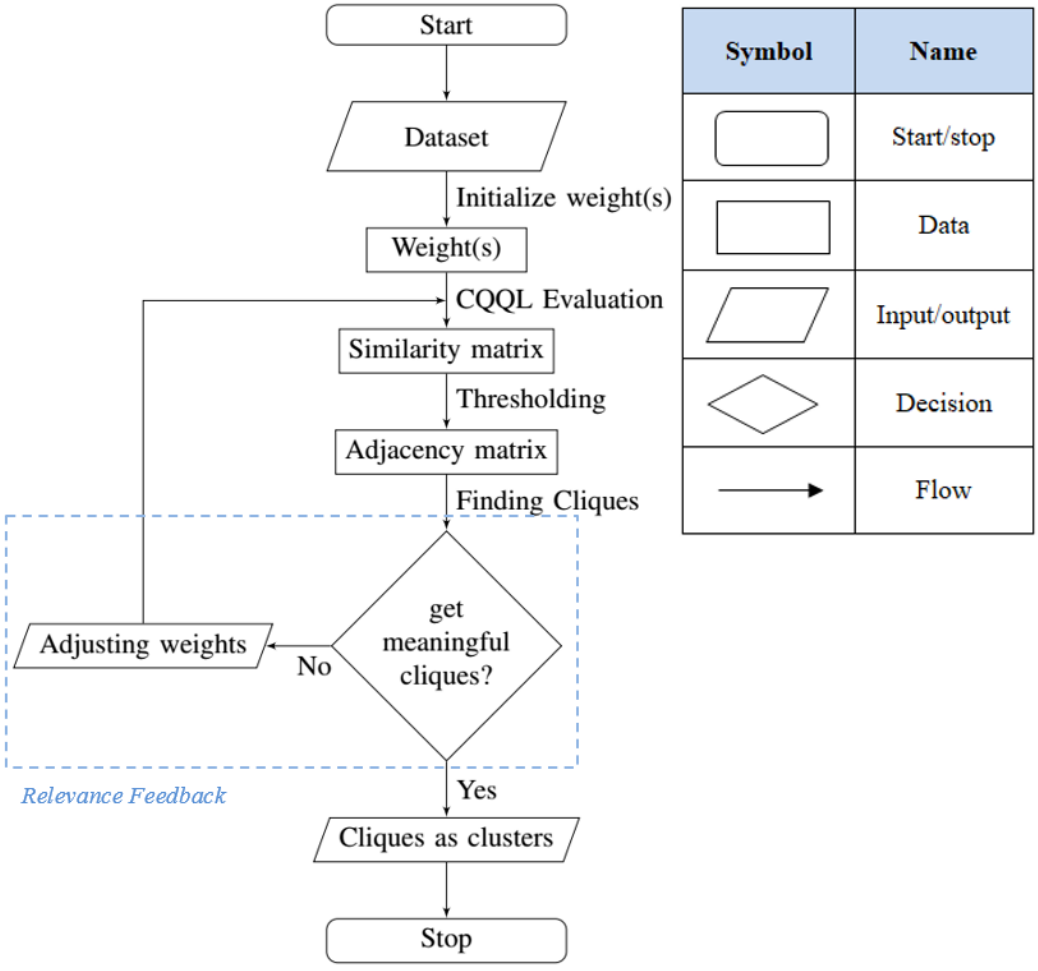


Fig. 4.2 Non-TI Clustering.

In our approach, we initially calculate the similarities between objects, forming a similarity matrix. We then derive an adjacency matrix from the similarity matrix based on a

threshold value, th. Using the clique approach described in Section 3.4.3, we identify the clusters. Later we incorporate weight(s) on each atomic condition, taking into account their relative importance in calculating the similarities between objects. Additionally, for improved user satisfaction, we adjust the weight(s) based on relevance feedback from the user, leading to the discovery of more meaningful clusters (for more information on relevance feedback, read Chapter 5).

As an illustration, we can recall the Example 4.1 and take into account a weighted CQQL query $q^{\Theta}(p_1, p_2)$ that has weight variables $\Theta = \{\theta_i\}$. The weighting function w assigns a value in the range of $[0,1]$ to every weight variable $\theta_i \in \Theta$. Therefore,

1. **Considering weighted conjunction**

   - **Complimented form of a weight**
     Assigning a weight to each atomic condition in a complimented form of one against other, the arithmetic evaluation of Equation 4.19 yields:.

     $$school(p_1,p_2) \cdot \Big(graduation(p_1,p_2) \cdot \theta + admission(p_1,p_2) \cdot (1-\theta) \Big)$$

   - **Weights based on Influence**
     Considering the weighted query $q^{\Theta}(p_1, p_2)$ with two weight variables $\theta_1, \theta_2 \in \Theta$, the arithmetic evaluation of 4.20 yields:

     $$school(p_1,p_2) \cdot \Big((graduation(p_1,p_2) + (1-\theta_1) - graduation(p_1,p_2) \cdot (1 - \theta_1)) \cdot (admission(p_1,p_2) + (1-\theta_2) - admission(p_1,p_2) \cdot (1-\theta_2)) \Big)$$

2. **Considering weighted disjunction**

   - **Complimented form of a weight**
     The logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. Therefore, the arithmetic evaluation of Equation 4.21 yields:.

     $$school(p_1,p_2) \cdot \Big(graduation(p_1,p_2) \cdot \theta + admission(p_1,p_2) \cdot (1-\theta) \Big)$$

   - **Weights based on Influence**
     Considering the weighted query $q^{\Theta}(p_1, p_2)$ with two weight variables $\theta_1, \theta_2 \in \Theta$, the arithmetic evaluation of 4.22 yields:

     $$school(p_1,p_2) \cdot \Big((graduation(p_1,p_2) \cdot \theta_1 + admission(p_1,p_2) \cdot \theta_2 - graduation(p_1,p_2) \cdot admission(p_1,p_2) \cdot \theta_1 \cdot \theta_2) \Big)$$

The query condition is executed on $P \times P$. The evaluation of $q^{\Theta}(p_1, p_2)$ is done through the use of the function $eval(q^{\Theta}(p_i, p_j), w)$ which determines a similarity value in the range of [0,1] for each pair of objects $(p_i, p_j) \in P \times P$ and generates a similarity matrix $S = (s_{ij})$. The adjacency matrix $Adj = \{a_{ij}\}$ can be derived from the similarity matrix S by using Equation 4.23 with a set threshold value th.

$$a_{ij} = \begin{cases} 1 & \text{if } s_{ij} \geq th \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

Next, an undirected graph G is constructed using the adjacency matrix Adj. With the algorithm presented by Bron and Kerbosch [1973], maximal cliques can be discovered in G, which are lists of vertex subsets that have two properties: every pair of vertices in one of the listed subsets is connected by an edge, and no additional vertices can be added to the listed subset while preserving its complete connectivity. Cliques are considered as clusters, though it is important to note that they can overlap.

4.6 Clustering Properties

Let $P = \{p_1, p_2, ..., p_n\}$ be a finite set of actors (objects) having two attributes *admission* and *graduation* and $S = \{p_{ij}\}$ be a similarity matrix which consists of similarity values for pairs of actors.. For simplicity, we use A for *admission* and G for *graduation*. Hence an adjacency matrix $Adj = \{a_{ij}\}$ can be derived from the similarity matrix S by applying Equation 4.23 with threshold th where $a_{ij} = 1$ means there is an edge (similar) between p_i and p_j and $a_{ij} = 0$ means no edge (dissimilar).

Property 4.1 (Set Operation). If $E_A \subseteq P \times P$ is a set of edges based on A and $E_G \subseteq P \times P$ is a set of edges based on G, then $E_{A \wedge_\theta G}$ is a set of edges of weighted conjunction or $E_{A \vee_\theta G}$ is a set of weighted disjunction of A and G with weighted variable $\theta \in [0,1]$ so that $E_{A \wedge_\theta G}, E_{A \vee_\theta G} \subseteq E_A \cup E_G$ holds.

Proof. Using Equation 4.16 it can be demonstrated easily that if $\theta = 1$ then $E_{A \wedge_\theta G} = E_A = E_{A \vee_\theta G}$ and if $\theta = 0$ then $E_{A \wedge_\theta G} = E_G = E_{A \vee_\theta G}$ and for other values of θ within $[0,1]$, $E_{A \wedge_\theta G}, E_{A \vee_\theta G} \subseteq E_A$ or $E_{A \wedge_\theta G}, E_{A \vee_\theta G} \subseteq E_G$ or $E_{A \wedge_\theta G}, E_{A \vee_\theta G} \subseteq E_A \cap E_G$. □

4.7 Summary

In summary, this chapter focused on the mathematical formalism of CQQL based on quantum mechanics. It described the construction and arithmetic evaluation of a CQQL query. It

explains how weights can be incorporated into a query based on influences to personalize the CQQL query. With an example, we explained how CQQL can be used as a multimodal logical query language. Using a flow diagram, we explained our proposed non-TI clustering approach. This chapter concluded with clustering properties.

Chapter 5

Relevance Feedback-based Learning of Personalized CQQL Queries

Personalization is the act of customizing information outcomes to meet an individual's specific information needs (IN). This is crucial because a single query result may not fulfill the user's expectations. Personalization has a long history in information retrieval (IR), with many methods having been developed over the years. These methods can be divided into two types: explicit and implicit [Hearst 2009, cf. Ch. 9]. Explicit personalization is when the user directly provides information about their IN, such as in the case of relevance feedback (RF). Implicit personalization is when the user's IN is inferred based on their feedback history or query log data.

Commercial web IR engines often use implicit personalization techniques, such as considering the user's location and profile, or reordering results based on data analysis of query logs or click-through records. Explicit personalization strategies include recommendations, such as frequently viewed clusters or term options for the user to select from. Personalization is linked to query reformulation, which entails modifying a query after reviewing the initial results. In IR, query reformulation can be facilitated through term suggestions or reformulation (RF).

For clarity, we group personalization and RF into the same category since RF in CQQL does not modify the logical structure of a query. The term "query reformulation" refers only to alterations made to the logical structure of a query.

5.1 Relevance Feedback

Introduced in the mid-1960s, relevance feedback (RF) is a technique that helps users in improving the quality of their query statements. Users can select nodes expected to reside in the same cluster to their needs and submit that information to the IR system. The information can then be utilized to find clusters that satisfy the user's needs [Ruthven 2001]. RF is a cyclical process: a set of clusters retrieved in response to an initial query are presented to the user, who indicates whether clusters are meaningful or not. The system uses this information to generate a modified query, which is then used to retrieve a new set of clusters for the user to view. This process is known as an iteration of RF and repeats until it satisfies the user's needs as shown in Figure 5.1.

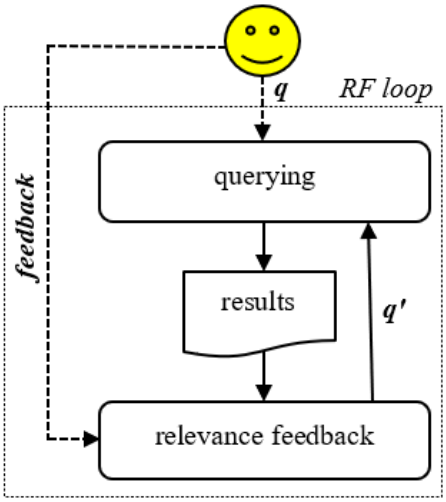


Fig. 5.1 Relevance feedback.

5.2 Types of Relevance Feedback

The types which are mainly discussed in Relevance Feedback are explicit feedback, implicit feedback, and pseudo feedback.

5.2.1 Explicit Feedback

Assessors of relevance provide explicit feedback on the relevance of nodes retrieved for a query. Only when the assessors (or other system users) are aware that the feedback provided is understood as relevance judgments is this form of input classified as explicit.

Users can use a binary or graded relevance system to express their significance. For a particular query, binary relevance feedback tells whether a document is relevant or irrelevant. On a scale of numbers, letters, or descriptions, graded relevance feedback reveals the relevance of a document to a query (such as "not relevant", "somewhat relevant", "relevant", or "very relevant"). Graded relevance can alternatively be expressed as a cardinal ordering of documents established by an assessor, in which the assessor arranges documents in a result set in order of (typically descending) significance.

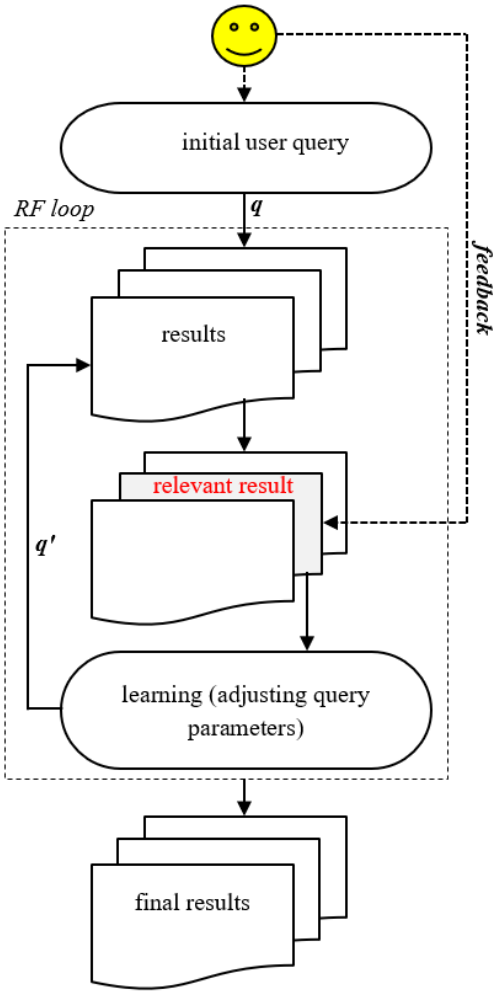


Fig. 5.2 Explicit relevance feedback.

Figure 5.2 shows the flowchart of a typical Content based image retrieval process with relevance feedback [Liu et al. 2007]. Based on the above general assumptions, a typical scenario for relevance feedback in content-based image retrieval is as below [Liu et al. 2007; Zhu and Huang 2003]:

1. The system provides initial retrieval results given query examples.

2. The user determines how relevant (positive examples) or irrelevant (negative examples) the preceding results are to the query.

3. The user's feedback is used to train a machine learning algorithm to create a new ranking model. Return to Step (2) and repeat the process.

Steps $(2) - (3)$ are repeated till the user is satisfied with the results.

Step (3) is comparably the most important step and different approaches can be used to learn the new query. A few generally adopted approached are introduced in the following.

Re-Weighting Approaches

The explicit personalization technique of assigning weights to different parts of a query has been used since the early days of Information Retrieval (IR). Researchers such as Lee [1994] have explored extended Boolean models, such as Fuzzy Set, Waller-Kraft, Paice, P-Nonn, and Infinite-One, which enable users to express the significance of their query terms by assigning weights at the time the query is formulated (QFT). Assigning weights allows the user to express the significance of the co-occurrence of query terms within the text.

A typical approach in step (3) is to automatically adjust the weights of low-level features to accommodate the users' need, rather than asking the user to specify the weights as adopted in earlier content based image retrieval systems. This re-weighting step dynamically updates the weights embedded in the query (not only the weights to different types of low level features such as color, texture, shape, but also the weights to different components in the same feature vector) to model the high-level concepts and perception subjectivity [Rui et al. 1998].

Query Point Movement Approaches

The query-point-movement (QPM) approach [Liu et al. 2007] is another method. It enhances the query point's estimation by moving it toward positive examples and away from negative examples. The Rocchio's formula [Rocchio 1971] is a widely used query point moving technique.

Machine Learning Approaches

The use of machine learning techniques is also common. An active learning algorithm based on *support vector machines* (SVM) can be used to conduct relevance feedback in image retrieval. This algorithm identifies the most informative images to query a user and quickly learns a boundary that distinguishes the images meeting the user's query concept from the

remaining dataset. SVM offers the advantages of (i) high generalization ability and (ii) minimal training sets.

5.2.2 Implicit Feedback

For completeness, implicit relevance feedback is noted even though it is not explored in detail in this dissertation. The user's behavior is used to indicate this type of feedback. Such as the type of document they view or do not view, and the amount of time they spend viewing a document. The user is not a participant in the feedback process. Instead, as shown in Figure 5.3, the system derives the feedback information implicitly. There are two basic approaches to compiling feedback information: it derives feedback information from the top-ranked documents in the result set, which is known as *local analysis*, or (ii) derive feedback information from external sources, such as a thesaurus or term relations extracted from document collections, which is known as *global analysis* (read Baeza-Yates and Ribeiro-Neto [2011], cf. Chapter 5 for details).

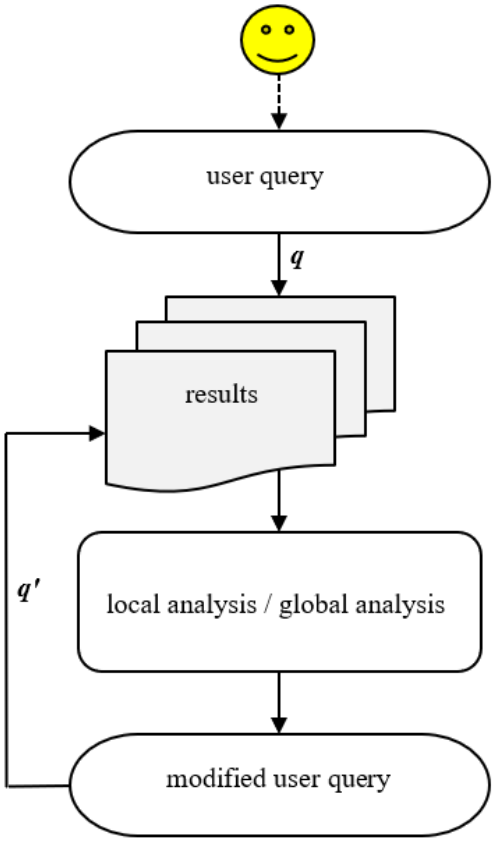


Fig. 5.3 Implicit relevance feedback.

5.2.3 Pseudo Feedback

Pseudo relevance feedback, often called *blind relevance feedback*, is a technique for performing automatic local analysis. This Feedback method automates the manual part of the feedback, resulting in better results for the user. The system locates relevant documents and assumes that the top k rated documents are those that are relevant, using the relevance feedback methodology. The procedure of Relevance feedback is:

1. From the initial user query q, the system generates a result set, which is a ranking of relevant documents.
2. The top k best ranked documents in the result set are used to extract and rank the candidate expansion terms.
3. The top m terms, or groups of terms, are added to the original query q to create a new system query q', depending on the ranking technique adopted.
4. The system generates a final result set, which it then sends to the user.

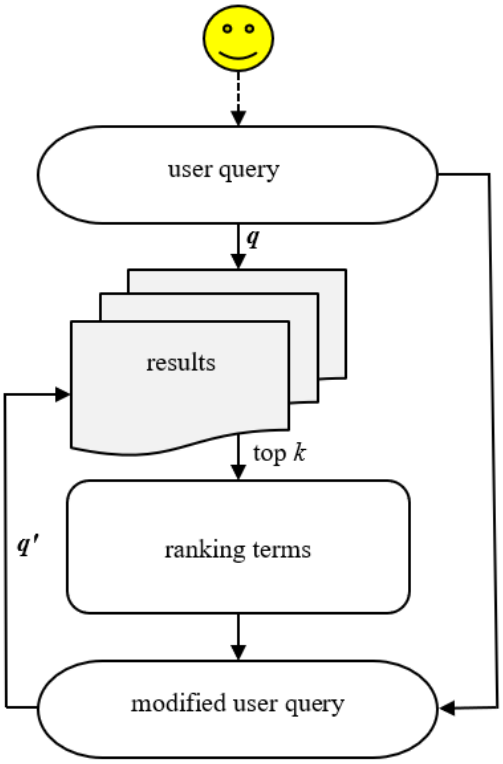


Fig. 5.4 Pseudo relevance feedback.

Several approaches, such as local clustering [Attar and Fraenkel 1977] and local context analysis [Xu and Croft 1996], can be used to rank such candidate expansion terms. In the first scenario, terms are clustered based on their frequency of co-occurrence within the top k best ranked documents, and each term is given a score based on its distance from query terms inside the cluster more closely connected to the query. In the second scenario, groupings of terms are derived from the top k best ranked documents and scored using a variant of $tf - idf$ ranking based on their similarity to the entire query q.

5.3 Relevance Feedback within the CQQL-based Non-TI Clustering

The core idea of relevance feedback is to modify and learn weights through user interactions that rely completely on user feedback. The idea behind user feedback is simple: the user is given the opportunity to modify resulting clusters.

We consider a weighted CQQL query, denoted as q^{Θ}, applied to pairs of objects with weight variables represented by $\Theta = \theta_i$. A weighting function, denoted as w, assigns values within the range of $[0,1]$ to the weight variable $\theta_i \in \Theta$. We use w^I to refer to an initial weighting function that sets the weight variable(s) to an initial value, which can be either 0.5 or 1. The query is executed on pairs from n objects. The evaluation of q^{Θ} is accomplished through the function $eval(q^{\Theta}, w)$. Executing $eval(q^{\Theta}, w)$ results in a similarity value within the range of $[0,1]$ for each pair of objects $(o_i, o_j) \in O \times O$, thereby generating a similarity matrix $S = (s_{ij})$ with respect to the query and the specified weighting scheme. Additionally, we make the assumption that there exists an unknown target function, denoted as w^T, associated with q^{Θ}. This target function produces a similarity matrix that optimally matches the user's intended semantics of the query..

During the initial query formulation, w^T is usually unknown. This presents a challenge because the initial weighting scheme w^I often generates clusters that do not align well with the user's expectations. Consequently, there is a need to progressively approximate the target clusters that would be generated by w^T based on user feedback.

Definition 5.1 (Feedback). Let C be a set of clusters of objects O, $th \in [0,1]$ be a threshold, $c \in C$ be a cluster and $co \subseteq O$ be a set of objects. Depending on the user interaction, every feedback falls into one of the two feedback categories below:

(i) Feedback for constructing a new cluster: A feedback f^{co} requires all objects $o \in co$ to reside in a same cluster i.e. $co \subseteq c \in C$ produced by a weighted query q^{Θ}. A weighting scheme w fulfills the feedback f^{co} if and only if

$$\forall_{o_1,o_2} \in co : eval(q^{\Theta}(o_1,o_2),w) \geq th \tag{5.1}$$

holds. That is there are edges between all object pairs.

(ii) Feedback for removing an object from a cluster: A feedback $f^{o,c}$ requires an object $o \in c$ should be outside of the cluster c i.e. $o \notin c$ produced by a weighted query q^{Θ}. A weighting scheme w fulfils the feedback $f^{o,c}$ if and only if

$$\exists_{o_1} \in c : eval(q^{\Theta}(o_1,o),w) < th \tag{5.2}$$

holds.

For example, if we take the complementary form of a weight from Equation 4.16, Figure 5.5 illustrates the results of evaluating the CQQL query $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta} admission(p_1,p_2)\Big)$. This evaluation considers various weight values for the weighting variable θ and includes three conditions related to attributes like *school*, *graduation*, and *admission*. These attributes represent the similarities between two friends, p_1 and p_2, with $s_{p_1,p_2}(1,0.6,0.3)$, where $school(p_1,p_2) = 1$, $graduation(p_1,p_2) = 0.6$, and $admission(p_1,p_2) = 0.3$. The threshold th is set at 0.5 (indicated by the red horizontal line).

Considering the feedback f^{p_1,p_2}, it becomes evident that with an initial weight value of 0.5, there is no connection between p_1 and p_2 (highlighted in yellow). This implies that, initially, they cannot be placed in the same cluster. However, if the user requests them to be in the same cluster, it is achievable with a weight value of $w(\theta) \leq 0.33$.

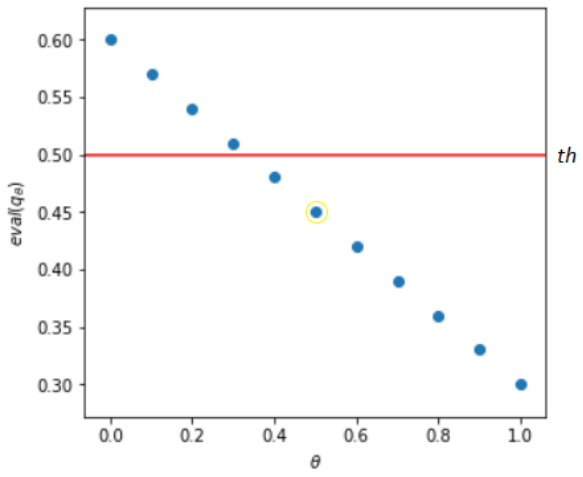


Fig. 5.5 Evaluation of $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta} admission(p_1,p_2)\Big)$ with respect to θ.

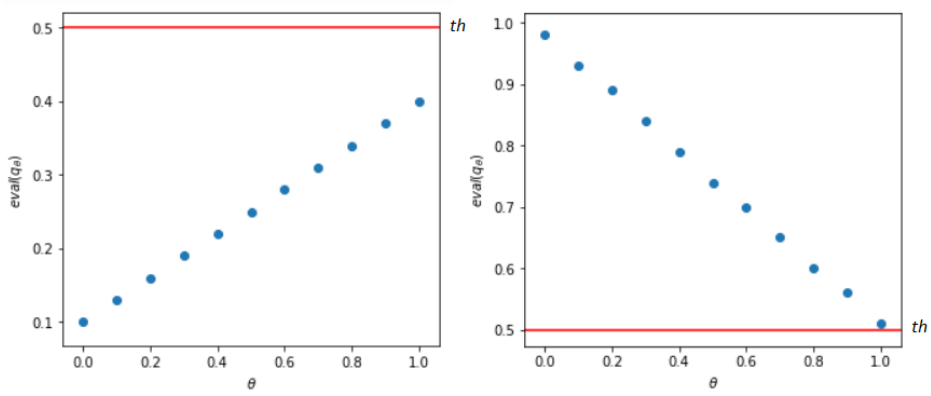


Fig. 5.6 Inconsistent case of feedback $f^{\{p_1,p_2\}}$ (left) and $f^{p_1,\{p_1,p_2,\cdots\}}$ (right) for two specific objects.

Every feedback with respect to a weighted query can be:

- *Inconsistent:* If no $w(\Theta) \in [0,1]$ can satisfy f^{co} or $f^{o,c}$ (see Figure 5.6).
- *Consistent:* Otherwise, the feedback is called consistent (see Figure 5.5).

To provide an example of an *inconsistent* feedback, consider a user feedback f^{p_1,p_2}. We can demonstrate inconsistency by setting the threshold to $th = 0.5$ and defining the similarities between two friends, p_1 and p_2, with respect to attributes like *school*, *graduation*, and *admission* as $s_{p_1,p_2}(1,0.1,0.4)$, where $school(p_1,p_2) = 1$, $graduation(p_1,p_2) = 0.1$, and $admission(p_1,p_2) = 0.4$.

Similarly, we can show an *inconsistent* feedback $f^{p_1,p_1,p_2,\cdots}$ by setting the threshold to $th = 0.5$ and defining the similarities as $s_{p_1,p_2}(1,0.98,0.51)$, where $school(p_1,p_2) = 1$, $graduation(p_1,p_2) = 0.98$, and $admission(p_1,p_2) = 0.51$.

Both the feedback f^{p_1,p_2} and $f^{p_1,p_1,p_2,\cdots}$ cannot be satisfied because there is no weight that can be found, as illustrated in Figure 5.6.

In the process of relevance feedback, clusters are shown to the user, allowing them to make modifications based on their subjective needs. The initial point of relevance feedback is a structured query, which, in our context, is a weighted CQQL query. Figure 5.7 shows the state diagram of relevance feedback, and the states and transitions are defined as follows:

- *Start*: The system starts and initializes weight(s) $w = w^I$.
- *User interaction*: A set of clusters is shown to user so that he/she can provide feedback to modify clusters, if needed. For instance, new clusters can be stated or existing ones may be removed.

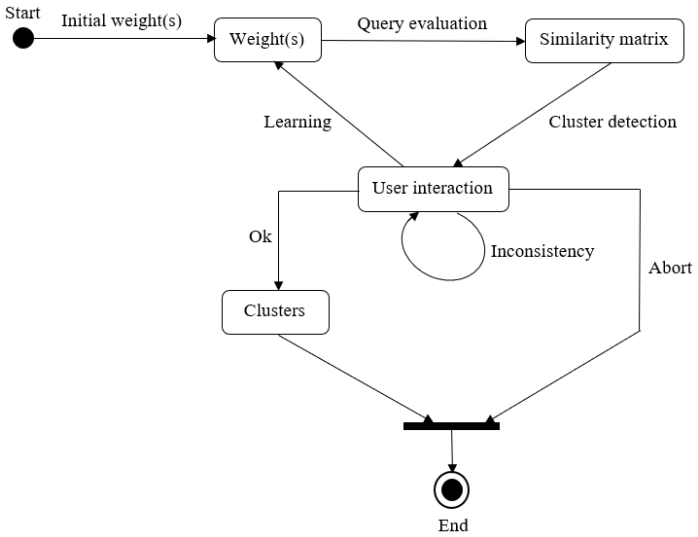


Fig. 5.7 State diagram of relevance feedback.

- *Inconsistency*: If the user feedback does not fulfil the constraint, the user is notified that the modification of clusters is not possible.
- *Learning*: Learning is used to find weight(s) $w(\Theta) \in [0,1]$ that fulfils the feedback f^{co} or $f^{o,c}$ by satisfying the constraint.
- *Weight(s)*: The results of the learning is a weight(s) for a given, weighted CQQL, which express the user-defined feedback.
- *Query Evaluation*: The query evaluation is the actual matching step which takes the learnt weight to produce clusters.
- *Similarity matrix*: The results of the query evaluation is a symmetric similarity matrix.
- *Cluster detection*: It yields a set of clusters from the current similarity matrix.
- *Ok*: User stops altering clusters because of satisfaction.
- *Clusters*: A set of final clusters are presented to the user.
- *Abort*: User terminates the system.
- *End*: The system ends.

Following the user's adjustments to the clusters, it is necessary to inspect them for inconsistencies according to Definition 5.1. When the user modifies one or more clusters,

it typically results in the formation of new clusters. This process may also lead to the disappearance of old clusters while introducing new ones. If the user decides to stop altering these clusters either due to satisfaction with the results or discontinuation of the process, the system reaches a stable state. At this stage, the clusters no longer undergo any further changes, and it's important to note that in each round, all feedback must be gathered, forming a set of feedbacks. Consequently, the constraints become progressively stricter.

5.4 Weight Learning

The aim of the learning process is to determine weight values that establish a new set of clusters that align with user feedback. As an illustration, taking the complimented form of a weight as described in Equation 4.16, Figure 5.8 displays the clusters generated for clustering C_0 using an initial weight value ($\theta = 0.5$).

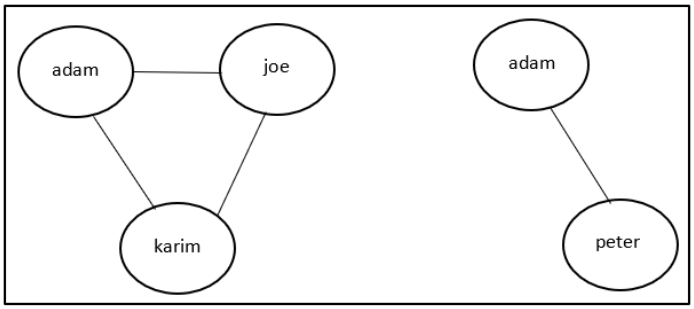


Fig. 5.8 Clusters (C_0).

A user observes that *karim* and *peter* are currently placed in separate clusters, even though they know each other and should ideally be in the same cluster. To address this, the user provides feedback, denoted as $f^{\{karim,peter\}}$, with the requirement that *karim* and *peter* should be in the same cluster.

The learning process attempts to find a weight value, represented as $w(\theta)$, within the range of $[0,1]$, which would establish an edge between *karim* and *peter* in the clustering. However, if it's not possible to establish such an edge, the user is informed with a message stating that modifying the clusters isn't possible due to an unsatisfiable constraint. In the case where an edge is possible, the learning process infers the appropriate weight value for a weighted CQQL query that meets the user's feedback.

As previously mentioned, in each round of the process, every feedback must be included in a feedback set, and this set represents a collection of constraints. In each round, it's imperative to ensure that the entire set of constraints is met. If it's not possible to find weight

values that satisfy the complete set of constraints, the system proceeds to examine subsets of constraints. In such cases, the system provides the user with multiple sets of alternative potential feedbacks that can be fulfilled. The steps of this procedure are outlined in Algorithm 3.

Algorithm 3: Algorithm for a new feedback.

Data: f `/* new feedback */`
Result: $w(\Theta)$, F_{alt}
1 $isPossible := true$
2 $F := F \cup f$ `/* F is the set of feedbacks */`
3 $isPossible :=$ check if F is satisfiable
4 **if** $\neg isPossible$ **then**
5 Report "no weight(s) is found that fulfils the feedbacks F"
6 $k := |F| - 1$
7 **do**
8 $F_{alt} := \{\}$ `/* set of sets of alternate possible fulfilled feedbacks */`
9 $S :=$ generate all k-subsets of F containing f
10 **for** *every subset* F_s *in* S **do**
11 $isPossible :=$ check if F_s is satisfiable
12 **if** $isPossible$ **then**
13 $F_{alt} := F_{alt} \cup F_s$
14 **end**
15 **end**
16 **if** $F_{alt} \neq \emptyset$ **then**
17 Report "F_{alt} are the set of sets of alternate possible feedbacks that can be fulfilled"
18 **end**
19 $k := k - 1$
20 **while** $k \neq 1$
21 **end**
22 **else**
23 Report "weight(s) $w(\Theta)$ fulfils the feedback F"
24 **end**

As an illustration, suppose a user provides a set of feedbacks $F = \{f_1, f_2\}$ and a weight value $w(\Theta) \in [0,1]$ has been found that satisfies the related constraints. If the user then provides a new feedback f_3, and the system determines that no weight value satisfies the

constraints for the set of feedbacks $F = \{f_1, f_2, f_3\}$, it will then check the constraints for the *2-subsets* containing feedback f_3 ($\{f_1, f_3\}$ and $\{f_2, f_3\}$) and present the user with a set of alternate possible feedback sets F_{alt} that can be satisfied.

Weighted conjunction based on influence

Let's revisit the example from Section 4.1 and take into account the weighted query $q^{\Theta}(p_1, p_2)$, which involves two weight variables, $\theta_1, \theta_2 \in \Theta$. We can reformulate the query in Equation 4.17 as:

$$q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$$

Using Equation 4.7, the arithmetic evaluation of $q^{\Theta}(p_1, p_2)$ yields:

$$school(p_1,p_2) \cdot \Big((graduation(p_1,p_2) + (1-\theta_1) - graduation(p_1,p_2) \cdot (1-\theta_1)) \cdot (admission(p_1,p_2) + (1-\theta_2) - admission(p_1,p_2) \cdot (1-\theta_2))\Big) \in [0,1]$$

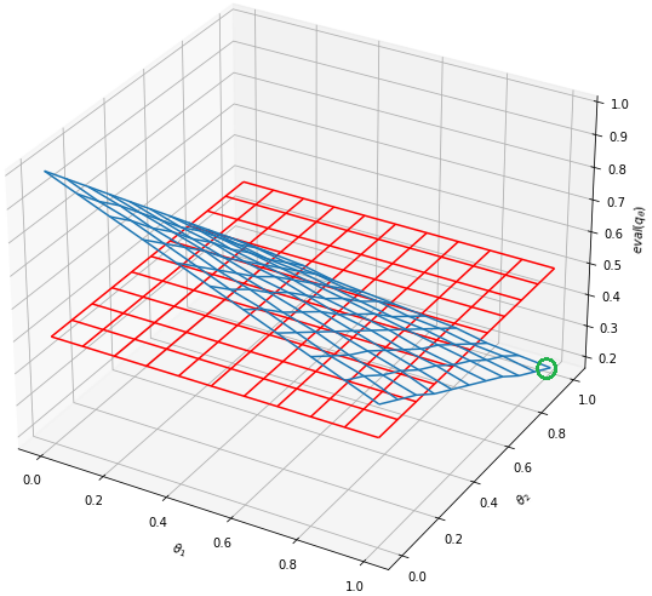


Fig. 5.9 Evaluation of $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$ by two weights θ_1 and θ_2.

Figure 5.9 illustrates the results of evaluating a given CQQL query, $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \wedge_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$, using various weight values for the weighting variables θ_1 and θ_2. There are three conditions related to the attributes

of *school*, *graduation* and *admission* that represent the similarities between two friends, p_1 and p_2, defined as $s_{p_1,p_2}(1,0.6,0.3)$, where $school(p_1,p_2) = 1$, $graduation(p_1,p_2) = 0.6$ and $admission(p_1,p_2) = 0.3$. Additionally, there's a threshold th set at 0.5 (indicated by the red frame). Considering the feedback $f^{\{p_1,p_2\}}$, it is visible that with the initial weight values set at 1, there is no edge (indicated by the green circle) between p_1 and p_2. This implies that, initially, they cannot be placed in the same cluster. However, if the user requests that they should be in the same cluster, it becomes apparent that this feedback can indeed be fulfilled, as the plane intersects the threshold level when certain weight values are applied.

Weighted disjunction based on influence

Again, lets revisit the example in Section 4.1 and consider the weighted query $q^{\Theta}(p_1,p_2)$ in Equation 4.22 with two weight variables $\theta_1, \theta_2 \in \Theta$ where the CQQL conditions with respect to *graduation* and *admission* are in conjunctive form. We can reformulate the query in Equation 4.17 as:

$$q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \vee_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$$

Using Equation 4.8, the arithmetic evaluation of $q^{\Theta}(p_1,p_2)$ yields:

$$school(p_1,p_2) \cdot \Big(graduation(p_1,p_2) \cdot \theta_1 + admission(p_1,p_2) \cdot \theta_2 - graduation(p_1,p_2) \cdot admission(p_1,p_2) \cdot \theta_1 \cdot \theta_2\Big) \in [0,1]$$

Figure 5.10 visualizes the results of the evaluation of the given CQQL query $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \vee_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$ using various values for the weighting variables θ_1 and θ_2 and with three conditions with respect to attributes *school*, *graduation* and *admission* representing the similarities of two friends p_1 and p_2 as $s_{p_1,p_2}(1,0.6,0.3)$ where $school(p_1,p_2) = 1$, $graduation(p_1,p_2) = 0.6$ and $admission(p_1,p_2) = 0.3$ and threshold th is 0.5 (red frame). if the user provides a feedback $f^{\{p_1,p_2\}}$ that they should be in the same cluster, it becomes apparent that this feedback can indeed be fulfilled, as the plane intersects the threshold level when certain weight values are applied.

It is clear that we face a non-linear optimization issue, because different weights can be present within the CQQL query. Therefore, sequential least squares quadratic programming (SLSQP) method implemented by Kraft [1988] can be used to solve non-linear optimization problem.

SLSQP uses Sequential Least Squares Programming to minimize an objective function of several variables with any combination of bounds, equality and inequality constraints.

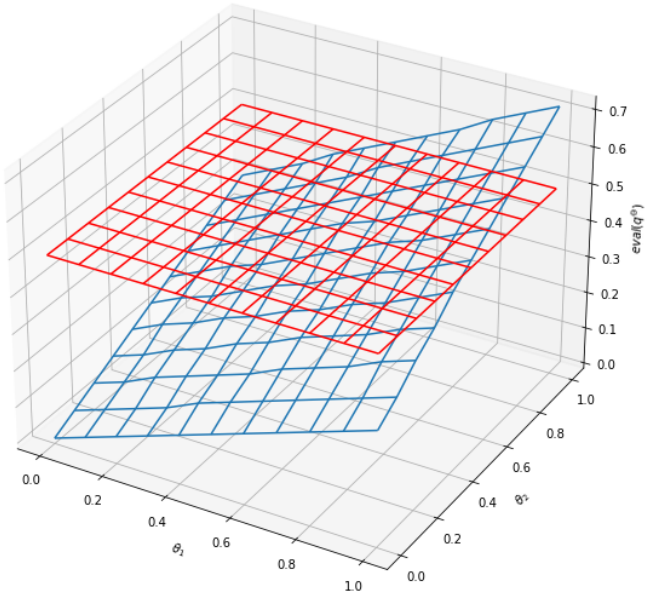


Fig. 5.10 Evaluation of $q^{\Theta}(p_1,p_2) = school(p_1,p_2) \wedge \Big(graduation(p_1,p_2) \vee_{\theta_1,\theta_2} admission(p_1,p_2)\Big)$ by two weights θ_1 and θ_2.

The method wraps the SLSQP Optimization subroutine. Note that the wrapper handles infinite values in bounds by converting them into large floating values. The method optimizes successive second-order (quadratic/least-squares) approximations of the objective function (via BFGS updates), with first-order (affine) approximations of the constraints. Figure 5.11 shows the flow chart of SLSQP.

For example, consider $x_1 = s_{p_1p_2}(graduation)$ be a similarity value with respect to *graduation year* and $x_2 = s_{p_1p_2}(admission)$ be a similarity value with respect to *admission year* of two people p_1 and p_2. Set threshold $th = 0.5$.

- **A complimented form of a weight**

  Logical transformation shows that the weighted conjunction is exactly equals to weighted disjunction. Therefore,

  Initialization: $\theta = 0.5$
  Objective function: $x_1 \cdot \theta + x_2 \cdot (1-\theta)$
  Constraint: $x_1 \cdot \theta + x_2 \cdot (1-\theta) \geq th$

- **Weights based on influence**

  For weighted conjunction:

  Initialization: $\theta_1 = 0.5, \theta_2 = 0.5$
  Objective function: $(x_1 + (1-\theta_1) - x_1 \cdot (1-\theta_1)) \cdot (x_2 + (1-\theta_2) - x_2 \cdot (1-\theta_2))$
  Constraint: $(x_1 + (1-\theta_1) - x_1 \cdot (1-\theta_1)) \cdot (x_2 + (1-\theta_2) - x_2 \cdot (1-\theta_2)) \geq th$

  For weighted disjunction:

  Initialization: $\theta_1 = 0.5, \theta_2 = 0.5$
  Objective function: $(x_1 \cdot \theta_1) + (x_2 \cdot \theta_2) - (x_1 \cdot x_2 \cdot \theta_1 \cdot \theta_2)$
  Constraint: $(x_1 \cdot \theta_1) + (x_2 \cdot \theta_2) - (x_1 \cdot x_2 \cdot \theta_1 \cdot \theta_2) < th$

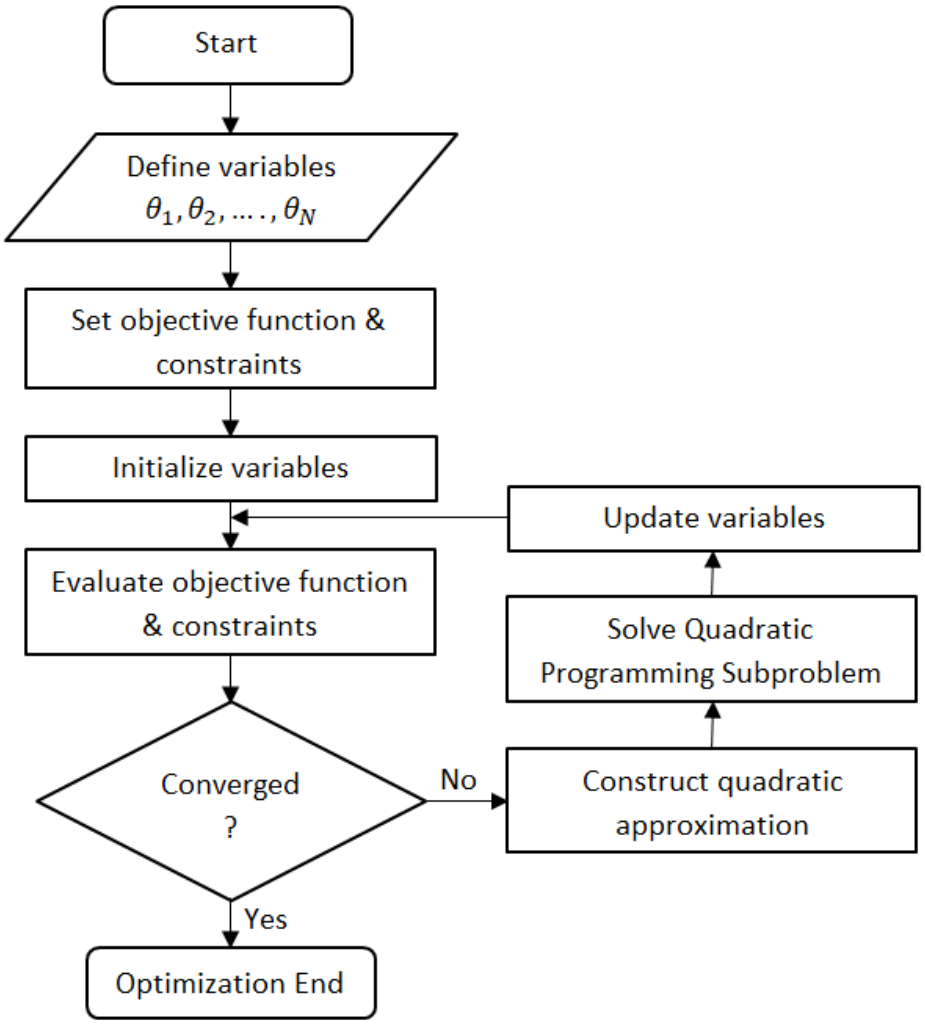


Fig. 5.11 Non-linear optimization.

5.5 Summary

Relevance Feedback is only used for user communication, whereas numeric weights are used for internal representation. The system redetects a set of clusters based on these weights, which must be consistent with the user's subjective perception of similarity to a given query.

While interacting with the system, the user is given the opportunity to reform the clusters by expressing which objects reside within the same clusters and which objects do not belong to a specific cluster. Based on this feedback, the system is able to learn numeric weight values for a query. When applied iteratively, this feedback results in an approximation of the desired clustering.

Chapter 6

Comparing Clusterings for Non-TI Clustering

Clustering objects involves grouping them based on how similar they are in their attribute values. Objects that have a high similarity value are considered more similar to each other and vice versa. The effectiveness of the clustering depends on the similarity within each group and the difference between groups. To evaluate the similarity between objects, a measure is used to determine the degree of resemblance. This leads to the creation of a similarity matrix, which quantifies the similarity between all pairs of objects.

The traditional clustering algorithms such as $k-$means, hierarchical clustering, and DBSCAN group objects based on their similarities in attributes, creating subsets where members are more alike within the same subset than between different subsets. The aim is to maximize cohesion within the subset and separation between subsets, but this can be problematic if the distance function is inadequate, as it may separate objects that should be together or vice versa. In such cases, a desired feature template or gold standard can be used to guide the separation process.

Comparing the similarity between sets of clusters produced by different algorithms has not received much attention. Clustering is used in many applications to reflect human intuition, physical, biological, or social connections, or laws, but incorporating human interpretation and biases into information retrieval tasks, such as web search, is a challenging problem. In such scenarios, human judgment is the standard for measuring relevance, raising the question of how to compare the performance of common clustering techniques and distance measures against a human-generated gold standard partitioning. Despite the lack of research into techniques for comparing clusterings, some significant work has been done, such as by Falkowski et al. [2007], Rand [1971] and Meilă [2003].

Several measures, such as Rand [1971] Index and Fowlkes and Mallows [1983] Index, have been proposed for comparing clustering pairs. Clustering performance can be evaluated using measures comparing the results of different algorithms on a dataset. These measures are often displayed using a 2×2 contingency matrix for ease of comparison. A pair counting method is combined in this research to populate the contingency matrix, offering a straightforward way to summarize the relationship between two sub-cluster memberships.

6.1 Clustering Comparison Criteria

In addition to selecting or creating a clustering method, once a collection of clusters has been established, there is still the concern of membership assignment quality in relation to the initial purpose of the clustering. This can be addressed through *Internal criteria*, *external criteria*, and *relative criteria*, according to Halkidi et al. [2001]. Relative and internal criteria evaluate whether the clustering is significantly different from chance through the use of Monte Carlo methods, while external criteria are used to compare the membership and structure of two clusterings, as noted by Theodoridis and Koutroubas [1999]. This dissertation focuses solely on external criteria.

Internal criteria are assessments of the clustering or its producing algorithm based on the attributes of the data set, such as the proximity matrix. These measurements evaluate characteristics like cohesion, separation, distortion, and likelihood. The results of these assessments can be greatly influenced by pre-determined parameters, such as the number of required clusters or minimum density, making them sensitive to both the quality of the clustering and the evaluation criteria used.

Relative criteria are used to evaluate and rank a clustering solution by comparing it to other clustering results generated by the same algorithm using different input parameters. These criteria are specifically chosen to match the algorithm and the data being analyzed.

External criteria serve the purpose of comparing one clustering to another clustering, a predefined gold standard, or a template representing desired features. Consequently, they create measures that are not influenced by the algorithm used for clustering, any predefined clustering assessment, the dataset itself, or problem-specific criteria.

The task of comparing clusterings is not straightforward as it involves more than just comparing two clusterings using similarity/dissimilarity measurements or algorithm traits. The meaning of comparing clusterings must be established.

Additionally, when comparing clustering pairs without a reference point such as a gold standard or desirable feature template, the comparison will only be numerical and will not

assess the quality of the clustering or its individual clusters. A sense of "goodness" must be introduced through a gold standard or desirable feature template.

6.2 Common Approaches in Comparing Clusterings

Evaluating the results of different clustering algorithms that have been applied to the same data set is crucial for improving the algorithms, quantifying their results, and determining their accuracy. This evaluation can be performed through the use of external criteria, which involves comparing two partitions and determining their similarity. There are several methods for performing this comparison, including *pair counting*, *set matching*, and *information-theoretic* approaches. This section will briefly cover these different clustering comparison methods.

A clustering C is a set $\{c_1, c_2, ..., c_k\}$ of non-empty subsets called clusters of a data set D of elements n such that $\cup_{i=1}^{k} c_i = D$. Assume that $c_i > 0$ for all $i = 1, 2, ..., k$. Let $C' = \{c'_1, c'_2, ..., c'_l\}$ be the second clustering of the same dataset D such that $\cup_{j=1}^{l} c_j = D$. Note that the two clustering may have different numbers of clusters.

The confusion matrix $M = (m_{ij})$ (or contingency table) of the pair C, C' is a $k \times l$ matrix whose ij-th entry equals the number of elements in the intersection of the clusters c_i and c'_j:

$$m_{ij} = |c_i \cap c'_j|$$

6.2.1 Clustering Comparison by Counting Pairs

Pair counting is a method of comparing pairs of entities based on the magnitudes of their attributes, traits, and other characteristics. It was first introduced scientifically by Thurstone [1927] in the Law of Comparative Judgement. In this approach, comparisons are made between pairs of entities. To apply pair counting to a confusion matrix, the members of one clustering are incrementally paired based on certain properties.

- *Symmetric:* $(p_1, p_2) = (p_2, p_1)$

  where $p_1, p_2 \in C$

The comparison of the pairs is made with the members of the other clustering that are similarly paired. The values in the confusion matrix are determined based on the relationships between the pairs from the two partitions.

T_{11} = pairs in the same cluster under C and C'

T_{00} = pairs in different clusters under both C and C'

T_{10} = pairs in the same cluster under C but not under C'

T_{01} = pairs in different clusters under C' but not under C

Note that the sum λ of these four factors equals the number of all possible point pairs

$$\lambda = T_{11} + T_{00} + T_{10} + T_{01} = \frac{n(n-1)}{2}.$$

Below is a general overview of the key techniques that utilize the pair counting method:

Chi Squared Coefficient

The earliest methods for comparing clusterings were created for statistical purposes. One well-known example is the Chi Squared Coefficient, which is considered representative of these methods. The measure is expressed as:

$$X(C,C') = \sum_{i=1}^{k}\sum_{j=1}^{l}\frac{(m_{ij}-E_{ij})^2}{E_{ij}} \text{ where } E_{ij} = \frac{|c_i||c'_j|}{n}$$

The Chi Squared Coefficient, developed by Pearson [1900], is a well-known measure for comparing clusterings and evaluating similarity between them. However, using the coefficient for this purpose requires assuming independence of the two clusterings, which is not always true. The coefficient measures the deviation from the independence assumption, and a high number indicates similar clusterings, as discussed in the work of Mirkin [2001].

Fowlkes and Mallows Index

The Fowlkes and Mallows Index is a metric used to evaluate the similarity between two hierarchical clusterings. It was introduced by Fowlkes [1983] and is calculated by counting the intersections of two hierarchical trees at each clustering level and summing up the counts of the intersections. The index ranges from 0 (maximum dissimilarity) to 1 (maximum similarity) and represents the cumulative similarity between the different levels of clustering. The measure is expressed as:

$$FM(C,C') = \frac{T_{11}}{\sqrt{(T_{11}+T_{10})(T_{11}+T_{01})}}$$

However, this measure has a drawback: for small numbers of clusters, the index is very high, even for independent clusterings, which can result in a maximum value for small numbers of clusters.

Rand Index

The Rand Index, proposed by Rand [1971], is a metric for evaluating the similarity between two different clusterings. It can be used to compare any two clusterings, whether they are produced by different algorithms, have different numbers of clusters, or are evaluated against different ground truth datasets or criteria. The measure is expressed as:

$$R(C,C') = \frac{2(T_{11}+T_{00})}{n(n-1)}$$

The Rand Index ranges from 0 (no similar pairs) to 1 (identical clusterings), and represents the likelihood that two objects will be treated similarly in both clusterings.

Adjusted Rand Index

The Rand Index of two arbitrary partitions is not constant as expected. To address this issue, Hubert and Arabie [1985] introduced an adjustment based on the null hypothesis of a generalized hyper-geometric distribution. The Adjusted Rand Index is calculated by subtracting the expected value under the null hypothesis of random clusterings with a fixed number of clusters and elements in each cluster from the Rand Index. The Adjusted Rand Index normalizes the difference and provides a more accurate representation of the similarity between the two partitions. It is defined as follows:

$$AR(C,C') = \frac{T_{11} - \frac{(T_{11}+T_{10})(T_{11}+T_{01})}{\lambda}}{\frac{2T_{11}+T_{10}+T_{01}}{2} - \frac{(T_{11}+T_{10})(T_{11}+T_{01})}{\lambda}}$$

The Variation of Information (VI) measure has an expected value of 0 for independent clusterings and a maximum value of 1 for identical clusterings. However, its significance is questionable due to the strong assumptions it makes about the distribution. Some pairs of clusterings may result in negative values, as noted by Meilă [2003].

Precision, Recall and F-measure

Another way of comparing clusterings is to use the well known precision and recall measures. For a gold standard P, then:

$$Precision(C,C') = \frac{T_{11}}{T_{11}+T_{10}} \text{ and } Recall(C,C') = \frac{T_{11}}{T_{11}+T_{01}}$$

The *F-measure* is a symmetric measure that combines precision and recall. The *F-measure* is defined as:

$$F\text{-}measure(C,C') = \frac{2T_{11}}{2T_{11}+T_{10}+T_{01}}$$

6.2.2 Clustering Comparison by Set Matching

Meilă and Heckerman [2001] computed the criteria H: a best match for each cluster of C in C' is done by scanning the elements m_{ij} of the contingency table in decreasing order. The largest of them, call it m_{ab}, entails a match between c_a and c'_b, the second largest not in row a or column b entails the second match, and so on until $min(k,l)$ matches are made. The index of the cluster c'_j in C' that matches cluster c_i is denoted by $match(i)$. Then

$$H(C,C') = \frac{1}{n}\sum_{j=match(i)} m_{ij}$$

The index is symmetric and takes value 1 for identical clusterings. But the criteria suffer from the "problem of matching". It first find the "best match" for each cluster and then add up the contributions of the match found. In doing so, it completely ignore what happens to the "unmatched" part of each cluster.

6.2.3 Clustering Comparison based on Information Theoretic Approaches

The field of information theory aims to enhance the description and quantification of data by utilizing the minimum amount of information for storage and communication. This is achieved through the use of information entropy, a measure that expresses the average number of bits required for data storage or communication. To compare the difference between two partitions, information-theoretic methods employ entropy in different ways. Two examples of these methods are Meila's Variation of Information [Meilă 2003] and Normalized Mutual Information [Strehl and Ghosh 2003 and Fred and Jain 2003].

Entropy can be described as the information conveyed by the uncertainty that a randomly selected point belongs to a certain cluster. In the context of clustering Entropy is defined as:

$$H(C) = -\sum_{i=1}^{k} P(i)\log P(i) \text{ where } P(i) = \frac{|c_i|}{n}$$

Informally, entropy of a clustering C is used to quantify the uncertainty of a randomly selected element's cluster. The entropy of a trivial clustering, which is one with only one cluster or n clusters, is zero since we already know the cluster of a randomly selected element.

Normalized Mutual Information

The mutual information between two clusterings C and C' is a measure of how much the knowledge of one clustering can reduce the uncertainty about a randomly picked element's cluster in the other clustering. It is calculated as the average amount of information that can be gained about the cluster of an element when the cluster of that element in the other clustering is known. It is defined as:

$$I(C,C') = \sum_{i=1}^{k} \sum_{j=1}^{l} P(i,j) \log_2 \frac{P(i,j)}{P(i)P(j)}$$

where $P(i,j)$ is the probability that an element belongs to cluster c_i in C and to cluster c'_j in C':

$$P(i,j) = \frac{|c_i \cap c'_j|}{n}$$

The Mutual Information I is a metric used to compare two clusterings, however, it is not limited to a constant value, making interpretation difficult. The bound of Mutual Information between two clusterings is determined by their entropies:

$$I(C,C') \leq \min\{H(C), H(C')\}$$

The Normalized Mutual Information (NMI) is a metric on the space of all clusterings, normalized to be in the range [0,1]. It was proposed by Strehl and Ghosh [2003] and Fred and Jain [2003] to address the difficulty in interpreting the mutual information due to its unbounded value. The NMI normalizes the mutual information between two clusterings by the geometric or arithmetic mean of their entropies, providing a meaningful interpretation of the similarity between two clusterings. The normalized mutual information between two clusterings is defined as:

$$NMI_1(C,C') = \frac{I(C,C')}{\sqrt{H(C)H(C')}}$$

This index has expected value 1 for identical clustering and 0 for independent clusterings if $P(i,j) = 0$ or $P(i,j) = P(i) \cdot P(j)$ for all i, $1 \leq i \leq k$, and for all j, $1 \leq j \leq l$.

The expected value of this index is zero for clusterings that are independent and its maximum value is 1 for identical clusterings.

According to Fred and Jain [2003], obtaining a dependable clustering of a set of elements requires utilizing a combination of multiple clusterings instead of relying on a single algorithm. Hence, the normalized mutual information between two clusterings is as follows:

$$NMI_2(C,C') = \frac{2I(C,C')}{H(C)+H(C')}$$

This index has expected value 1 for identical clustering and 0 for independent clusterings if $P(i,j) = 0$ or $P(i,j) = P(i) \cdot P(j)$ for all i, $1 \leq i \leq k$, and for all j, $1 \leq j \leq l$.

Variation of Information

The Variation of Information (VI) measure was introduced by Meilă [2003] to evaluate the similarity between two clusterings of the same data. It calculates the amount of information lost or gained when changing from one clustering C to another clustering C'. The measure is positive, symmetric, and transitive, and Meilă considers it a metric. However, it is not normalized, which would enhance its comparability to other measures.

$$VI(C,C') = H(C) + H(C') - 2I(C,C')$$

6.3 Clustering Comparison for Non-TI Clustering

Comparing clusterings presents a significant challenge because there is no available ground truth data for reference. As a result, it's not possible to assess performance quantitatively due to the absence of such ground truth data. Nevertheless, for the purpose of testing our approach, we make an assumption that involves considering a social network with clustering done by humans as an ideal solution. We use this human-generated clustering in our approach to evaluate a clustering distance, which measures the dissimilarity between two different sets of clusters.

To avoid any potential confusion, it's important to distinguish between a *cluster*, which refers to a group of data points, and *clustering*, which denotes the set of clusters produced through a clustering process.

The method of comparing clusterings using pair counting involves counting the number of pairs of objects that are grouped similarly in both clusterings. The process begins by pairing all members of one cluster and then comparing them to the correspondingly paired members in the other clustering. A contingency matrix is subsequently used to assign values based on different relationships between the pairs in each of the clusterings.

Given a human generated (ideal solution) clustering C_I and a clustering C produced by our approach of the data set $P = \{p_i\}_{i=1}^{n}$ with n peoples, we first define data set pairs $T_P = \{(p_i, p_j) | p_i, p_j \in P \text{ and } i < j\}$ as all the pairs realizable from the complete data set. Second, clustered pairs $T_{C_I} = \{(p_1, p_2) \in T_P | \{p_1, p_2\} \subseteq c_I \in C_I\}$ and $T_C = \{(p_1, p_2) \in T_P | \{p_1, p_2\} \subseteq c \in C\}$ that cluster together in C_I and C respectively. Using T_{C_I}, T_C and T_P the values of the four quadrants of the contingency matrix are realized as:

$T_{11} = |T_{C_I} \cap T_C|$ $\quad \left\{\text{pairs in } C_I \text{ and } C\right\}$

$T_{10} = |T_C \setminus T_{C_I}|$ $\quad \left\{\text{pairs in } C \text{ but not in } C_I\right\}$

$T_{01} = |T_{C_I} \setminus T_C|$ $\quad \left\{\text{pairs in } C_I \text{ but not in } C\right\}$

$T_{00} = |T_P \setminus (T_{C_I} \cup T_C)|$ $\quad \left\{\text{pairs not in both } C \text{ and } C_I\right\}$

Note that the sum λ of these four quadrants equals the number of all possible people pairs

$$\lambda = |T_P| = T_{11} + T_{10} + T_{01} + T_{00} = \binom{n}{2} = \frac{n(n-1)}{2}$$

In Table 6.1, each column of the matrix represents the instances within a cluster generated by humans (referred to as the "Ideal Solution"), while each row corresponds to the instances within a cluster produced by the system. By comparing the ideal solution against the produced it is easy to see if a system is confusing two clusters (i.e. commonly mislabelling one as another).

Table 6.1 Contingency matrix

	Human generated (Ideal solution)	
	Pairs in C_I	Pairs not in C_I
Produced (cluster)		
Pairs in C	T_{11}	T_{10}
Pairs not in C	T_{01}	T_{00}

The *Accuracy* $\mathscr{A}$ for a clustering C concerning a certain human generated clustering C_I indicates how *good* the clustering C describes the clustering C_I. It can be defined as:

$$\mathscr{A}(C_I, C) = \frac{T_{11} + T_{00}}{\lambda}$$

where $\mathscr{A}$ ranges from 0 (no pair classified in the same way under both clusterings) to 1 (identical clusterings).

We can derive the clustering distance $d(C_I, C)$ from the similarity measure $\mathscr{A}(C_I, C)$ by applying $\mathscr{A}(x, y) = e^{-d(x,y)}$ as:

$$d(C_I, C) = -ln(\mathscr{A}(C_I, C))$$

where d ranges from 0 (identical clusterings) to ∞ (no pair classified in the same way under both clusterings).

We propose Algorithm 4 to calculate the clustering distance d between C_I and C.

Algorithm 4: Algorithm for clustering distance.

Data: C, C_I
Result: d

$T_C := \{\}$
$T_{C_I} := \{\}$
for *each cluster c in C* **do**
  if $|c| > 1$ **then**
    $t_c :=$ generate all 2-subsets of c
    $T_C := T_C \cup t_c$
  end
end
for *each cluster* c_I *in* C_I **do**
  if $|c_I| > 1$ **then**
    $t_{c_I} :=$ generate all 2-subsets of c_I
    $T_{C_I} := T_{C_I} \cup t_{c_I}$
  end
end
$\lambda = \frac{n \cdot (n-1)}{2}$
$T_{11} := |T_{C_I} \cap T_C|$
$T_{10} := |T_C \setminus T_{C_I}|$
$T_{01} := |T_{C_I} \setminus T_C|$
$T_{00} := \lambda - T_{11} - T_{10} - T_{01}$
$\mathscr{A} := \frac{T_{11}+T_{00}}{\lambda}$
$d := -ln(\mathscr{A})$
return d

6.4 Summary

This chapter focuses on comparing the memberships of subgroups in two different clusterings. A contingency matrix with two columns and two rows (see Table 6.1) is used to evaluate the relationships between the members of the two subclusters. The matrix summarizes the pairwise relationships between the memberships of the two subclusters and is used to perform pair counting measures. The contingency matrix, typically used for comparing two populations, is used here to compare two partitioned spaces by assigning values to the matrix using a pair counting approach. The contingency matrix is used to identify the

important connections between two clusterings by examining the presence of member pairs, non-member pairs, and member pairs that are not shared.

Chapter 7

Implementation and Evaluation

When developing a Non-TI clustering approach, two questions usually arise: how fast is it and how good and meaningful are its results? Measuring running times in a reproducible manner is not easy because they depend on a variety of factors, but given a specific computer system and input instance, it is quite simple. Measuring the quality of the results, on the other hand, is much more difficult, even given a specific input graph. Depending on the application, smaller or larger cluster sizes, or more or less uniform cluster sizes, may be desired.

Graphs from applications are frequently unavailable due to data security or corporate interests. Graphs from applications where we know which clusters we want to find are even rarer, and even when they are, it is not always clear whether those clusters can be discovered solely from the graph structure.

We quantify the qualitative performance of the Non-TI clustering approach in various social and information networks by comparing it to state-of-the-art clustering methods. We evaluate the performance of the approach by calculating the clustering distance of the detected clustering when compared to the gold-standard, ground-truth clustering.

We conducted experiments on social networks, varying the sizes of the networks, to validate our proposed non-TI clustering approach. In this thesis, we describe datasets of four social networks with $N = 6$, $N = 20$, $N = 50$, and $N = 100$ people, where we have network information as well as node attributes, for our evaluation. We have explicit ground-truth cluster labels in addition to networks and attributes. The availability of such ground-truth allows us to evaluate clustering methods by quantifying the degree of similarity between the detected clustering and ground-truth clustering.

In this chapter, we start by describing our experimental setup and then continue with the results on the social networks by varying network sizes. We discuss the result of the proposed approach by comparing it with the shortcomings of the traditional clustering methods. To

evaluate the performance, we calculate the clustering distance of the detected clustering by comparing it with the ground-truth clustering.

7.1 Experimental Setup

The experiments were performed on a HP Pavilion x360 Convertible Laptop equipped with an Intel Core i5-10210U CPU running at a clock speed of 1.60 GHz (8 CPUs), 2.1 GHz and with 8 GB RAM. The operating system was Windows 11 Home 64-bit and the algorithms were implemented using Python 3.8.5. We use the Fruchterman-Reingold algorithm to visualize social networks by creating 2D representations based on the adjacency matrix.

7.2 Datasets

Social networks typically contain a massive amount of content and linkage data that can be analyzed. These data are further classified as unstructured or structured, depending on whether they are organized in a predefined manner or not. Graph-structured data is the most common type of structured data. In the simplest framework, they are represented as a graph $G = (V, E)$, where V is a set of nodes or entities (e.g., people, organizations, and products) and E is a set of edges or relationships that connect the nodes through patterns of interactions. This type of data is measured using social network analysis, a graph analytics application that focuses on extracting intelligence from such interconnected data. Unstructured data, on the other hand, refers to content data such as text, images, videos, tweets, product reviews, and other multimedia data shared on online social networks, also known as User Generated Content (UGC). Unfortunately, sometimes unstructured data from some social networks is not available to all. Therefore, we generate datasets of social network types of small (6 people), medium (20 people and 50 people), and large sizes (100 people and so on), respectively.

7.3 Experiments and Results

To assess the effectiveness of the proposed method, we conducted experiments on social networks of varying sizes. The process involve obtaining a similarity matrix S by using a conjunctive or disjunctive query q on the attributes of people pairs, such as *school*, *admissionyear*, and *graduationyear*. From this similarity matrix, we derive an adjacency matrix Adj, setting a threshold value of 0.5. This adjacency matrix is used to create an

undirected graph using the undirected graph drawing algorithm proposed by Fruchterman and Reingold [1991]. For comparison purposes, traditional clustering algorithms are also applied. To evaluate the performance of the proposed clique-guided approach, we calculate a clustering distance metric by comparing the detected clustering with the ground-truth clustering using the pair counting approach outlined in Section 6.3.

In user interaction, a user provides feedback based on his needs. In every round, the system adds the feedback to a feedback set that must be fulfilled. In addition, it provides the user with alternate feedbacks that can be fulfilled if no weight(s) is found that satisfies the corresponding constraints of the feedback set.

We conducted several experiments by varying the social network size. For a better understanding, we use a single experiment performed on a small social network of **6** people. We discuss three more experimental details and results from a social network of **20**, **50**, and **100** people in Appendix C. Within each experiment, we perform traditional clustering methods, an unweighted Non-TI clustering approach, and a weighted clustering approach with *(i)* a complimented form of a weight and *(ii)* weights based on influence, respectively. For comparing the similarities between ground-truth clustering and detected clustering, we calculate the clustering distance.

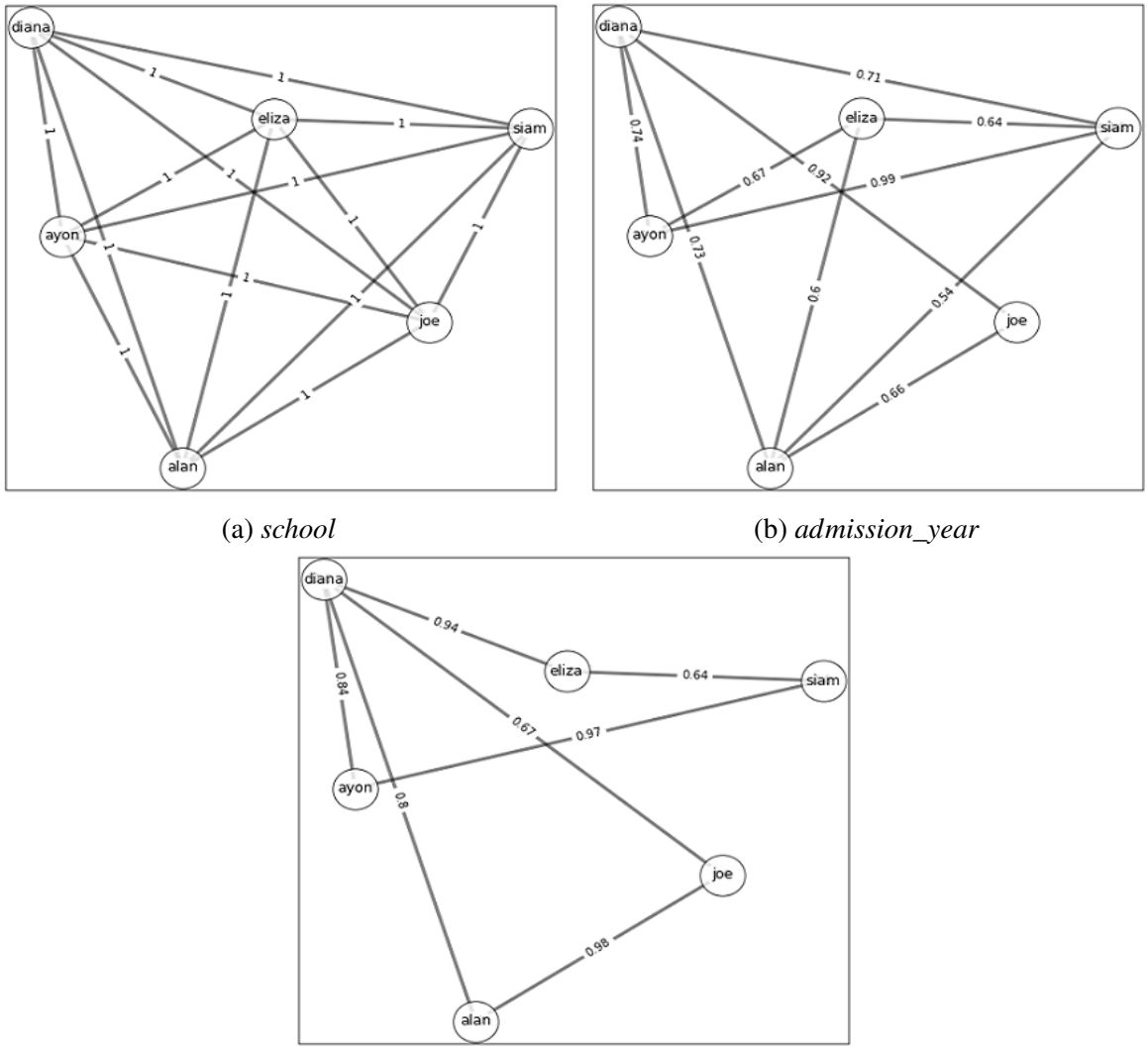
7.3.1 Clustering of a social network

Figure 7.1 illustrates the network structure of three attributes of a small social network of **six** people. The weighted edge between two people reflects the similarities among them. The corresponding weight of an edge is the similarity value obtained by the evaluation of a CQQL query.

We subdivide the experiment into two parts based on the operand conjunction ($\wedge$) or disjunction ($\vee$) contained inside the CQQL query. In the following, both conjunction and disjunction are taken into account.

7.3.2 CQQL query with conjunction ($\wedge$)

First, we execute the CQQL query of Example 4.1 on a small social network dataset of six people ($N = 6$) that returns a similarity matrix. Following the flow diagram in Figure 4.2, we generate a graph structure and perform clustering approaches on it to detect meaningful clusters. Later, we incorporate atomic condition weights into a CQQL query based on the effects of attributes on user interaction.



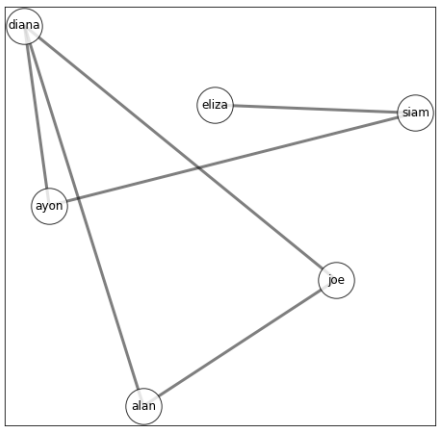
(a) *school* (b) *admission_year*

(c) *graduation_year*

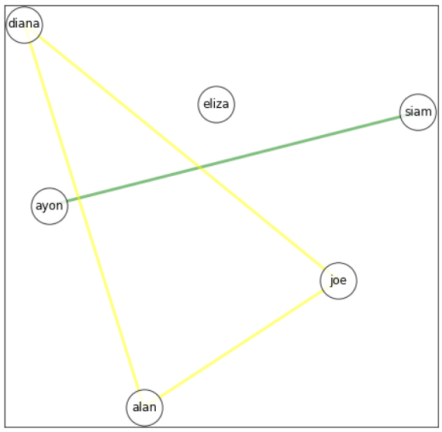
Fig. 7.1 Structure of a six-people network.

Clustering approaches

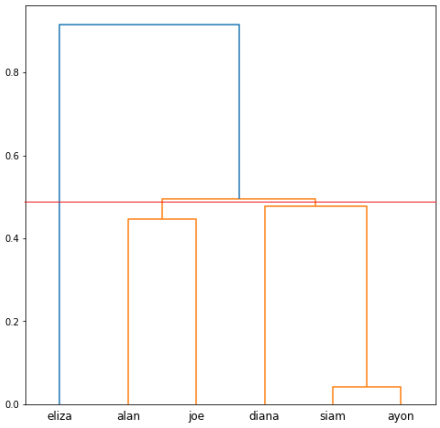
The first experiment in Figure 7.2 analyzes how clusters can be found by applying both traditional clustering techniques and our proposed Non-TI clustering approach. After executing the unweighted query $q(p_i, p_j) = school(p_i, p_j) \wedge \underbrace{\Big(graduation(p_i, p_j) \wedge admission(p_i, p_j)\Big)}_{conjunction}$ where $i = j = \{1, 2, 3, ..., N\}$, a similarity matrix is obtained. Figure 7.1 depicts the network structure constructed by the adjacency matrix derived from that similarity matrix with $th = 0.5$. An edge between a pair of nodes indicates that they know each other. It is noted



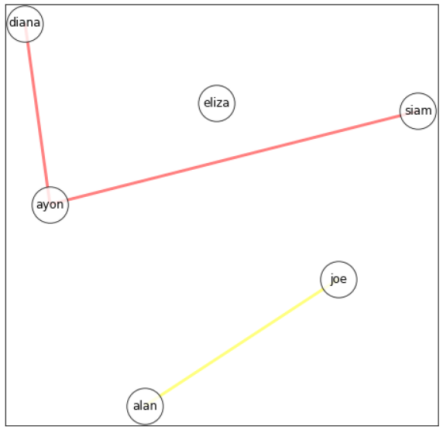
(a) Unweighted



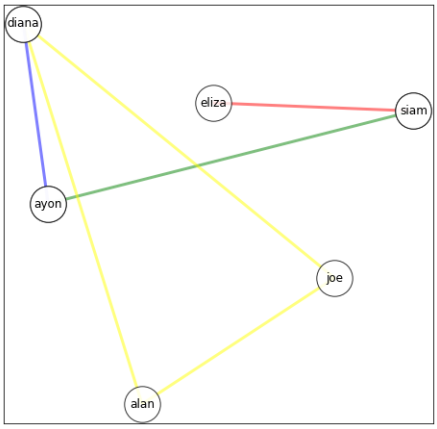
(b) k-medoids ($k = 3$) clustering



(c) Dendogram



(d) Hierarchical clustering



(e) Non-TI

Fig. 7.2 Clustering on a six-people network in *conjunction*.

that the edges among objects in the resulting clusters are taken over from the original graph to visualize relationships.

At first we apply k-medoids clustering method with $k = 3$ where k is the number of clusters. The starting point of the k-medoids method is the dissimilarity matrix. After executing the CQQL query, we get a similarity matrix. We obtain the dissimilarity matrix from the similarity matrix using transformation. Three clusters can be found: $\{eliza\}$, $\{siam, ayon\}$ and $\{alan, joe, diana\}$. Compared with the unweighted network structure, it is clear that the detected clusters do not have any clusters with $\{diana, ayon\}$ or $\{eliza, siam\}$ even if they know each other. Therefore, the detected clusters are not meaningful.

Applying single-linkage hierarchical clustering algorithm, a cluster hierarchy is created, often known as a dendrogram as shown in Figure 7.2c. Figure 7.2d shows that three clusters $\{diana, ayon, siam\}$, $\{alan, joe\}$ and $\{eliza\}$ can be found by drawing a horizontal line (red line as depicted in Figure 7.2c) using the threshold value 0.5. But the cluster $\{diana, ayon, siam\}$ violates the TI property because there is no edge within the pair $\{diana, siam\}$ which is visible in Figure 7.2d. Also, there are some pairs who know each other but detected clusters do not reflect their relationship.

Meaningful clusters can be found by applying our proposed Non-TI clustering approach where intra-cluster objects are close to each other (i.e., they are connected to each other with an edge) as shown in Figure 7.2e. Clustering methods with their corresponding detected clusters are summarized in Table 7.1.

Table 7.1 Clusters after clustering on a six-people network in *conjunction*

Methods	**Clusters**
k-medoids (k=3)	*{eliza}, {siam, ayon}, {alan, joe, diana}*
Hierarchical (Single-linkage)	*{siam, ayon, diana}, {eliza}, {alan, joe}*
Non-TI	*{siam, eliza}, {siam, ayon}, {diana, joe, alan}, {diana, ayon}*

User interaction

The next experiment analyzes how meaningful clusters can be found using our proposed approach with user interaction. To understand the concepts and performance of relevance feedback with user interaction, we consider both weighting strategies (see Section 4.3) in our experiments. In the following, we describe weighting strategies individually.

Complimented form of a weight

We conduct the experiment using the CQQL query with a weighted conjunction $q^{\Theta}(p_i, p_j) = school(p_i, p_j) \wedge \Big(graduation(p_i, p_j) \wedge_{\theta} admission(p_i, p_j)\Big)$ where $i = j = \{1, 2, 3, ..., N\}$. A

user provides feedback of expecting either *(i) a pair of people to reside within the same cluster* or *(ii) a person should be outside of a specific cluster*. The user provides feedback until he stops or is satisfied with the resulting clusters. The following describes the experiments with results and performance compared the results with ground-truth clustering in detail.

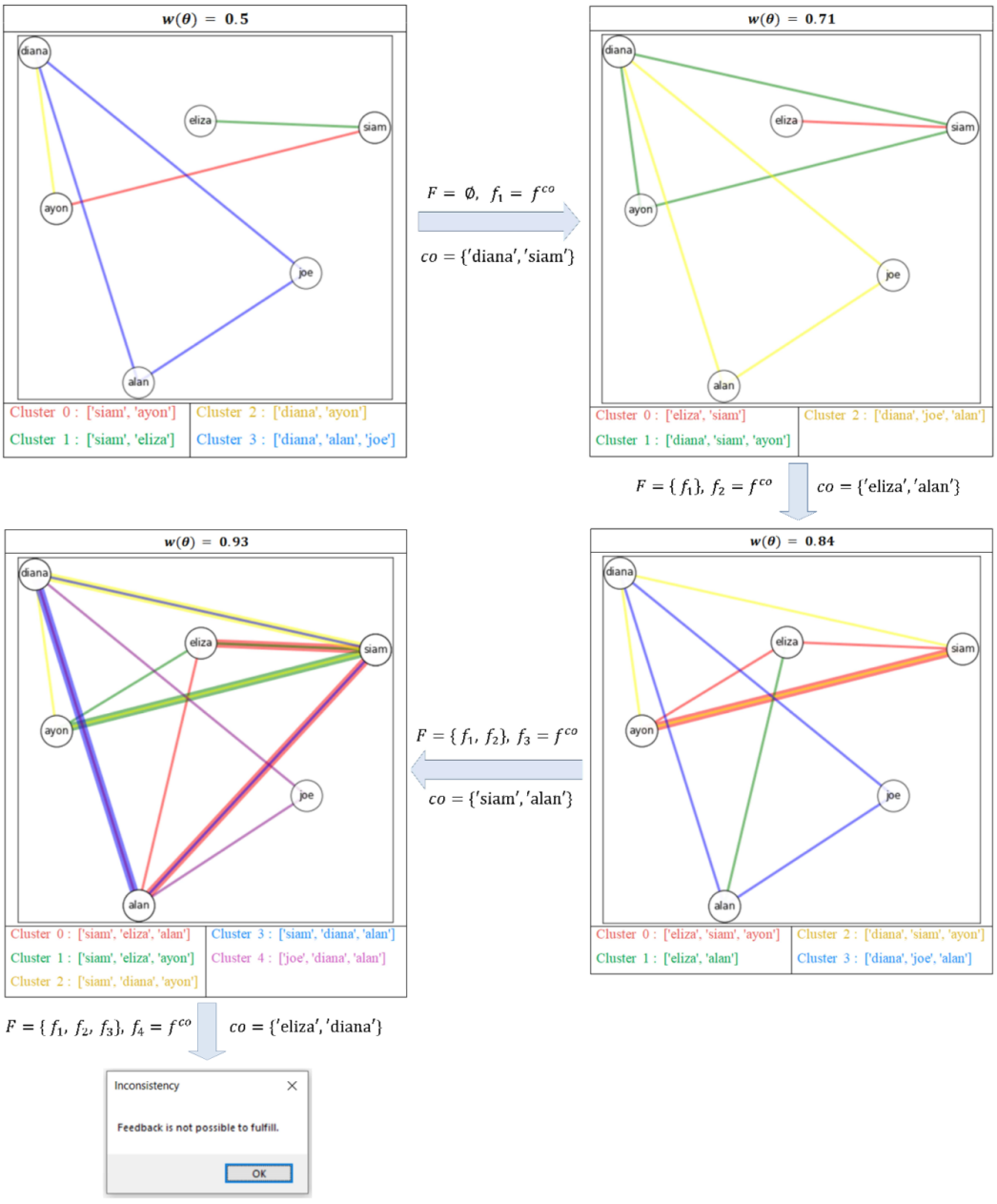


Fig. 7.3 Clustering based on f^{co} on a six-people network in *conjunction*.

1. **Feedback for constructing a new cluster**
   The experiment starts by assuming equal importance between the two attributes (by setting

$\theta = 0.5$) while conducting a conjunction ($\wedge$) on query q^{Θ} as seen in Figure 7.3 (top left). If the user provides feedback $f_1 = f^{co}$, requesting that the pair $co = \{'diana', 'siam'\}$ be placed in the same cluster, it can be seen from Figure 7.3 (top right) that a weight of $w(\theta) = 0.71$ has been found that satisfies the constraint and clusters the pair. This feedback is added to the set $F = \{f_1\}$. The user provides a second feedback $f_2 = f^{co}$, requesting that $co = \{'eliza', 'alan'\}$ be placed in the same cluster and a weight of $w(\theta) = 0.84$ has been found that satisfies the feedback and all previous feedbacks in F to form clusters, as seen in Figure 7.3 (bottom right in second row). This feedback is added to $F = \{f_1, f_2\}$. The user then provides a third feedback $f_3 = f^{co}$, requesting that $co = \{'siam', 'alan'\}$ be in the same cluster, and a weight of $w(\theta) = 0.93$ has been found that satisfies the constraint and clusters the group, as shown in Figure 7.3 (bottom left in second row). This feedback is added to the set $F = \{f_1, f_2, f_3\}$. Finally, the user provides a feedback $f_4 = f^{co}$, requesting that $co = \{'eliza', 'diana'\}$ be placed in the same cluster. In this case, no weight $w(\theta) \in [0, 1]$ that satisfies the feedback along with all previous feedbacks in F can be found, and no alternative feedbacks are possible. The user is notified of this as seen in Figure 7.3 (bottom left in third row).

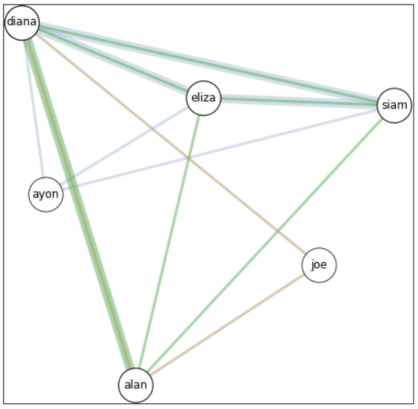


Fig. 7.4 Ideal Solution.

Table 7.2 Ground-truth clustering

Clusters
{diana, eliza, siam, ayon},
{diana, eliza, siam, alan},
{diana, joe, alan}

To test our approach, we evaluate a clustering distance of detected clustering compared with ground-truth clustering d in every round of our experiment in Figure 7.3. The interactions conform to the ideal solution. Figure 7.4 is our ideal solution and corresponding clusters of ground-truth clustering are listed in Table 7.2. It is visible in Figure 7.5 that the clustering distance d in every round decreases and converges towards the ideal solution.

2. **Feedback for removing a person from a cluster**
   The next experiment shows user feedback (category $f^{o,c}$) requiring that a person be outside of a cluster. If the user provides feedback by expecting to see $o =' alan'$ outside

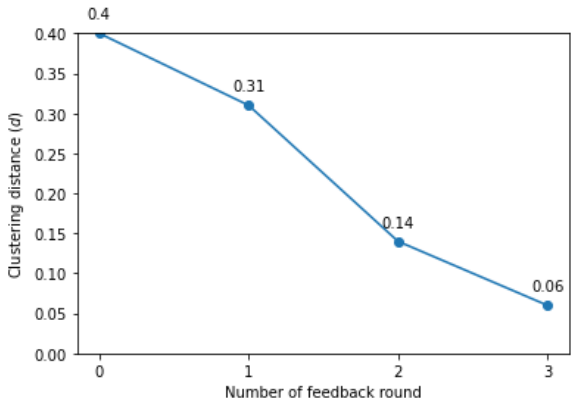


Fig. 7.5 Clustering distances based on f^{co} in Figure 7.3.

of $c = cluster\ 1$, Figure 7.6 demonstrates that $w(\theta) = 0.82$ is found, which fulfills the feedback by satisfying the constraint and detecting clusters.

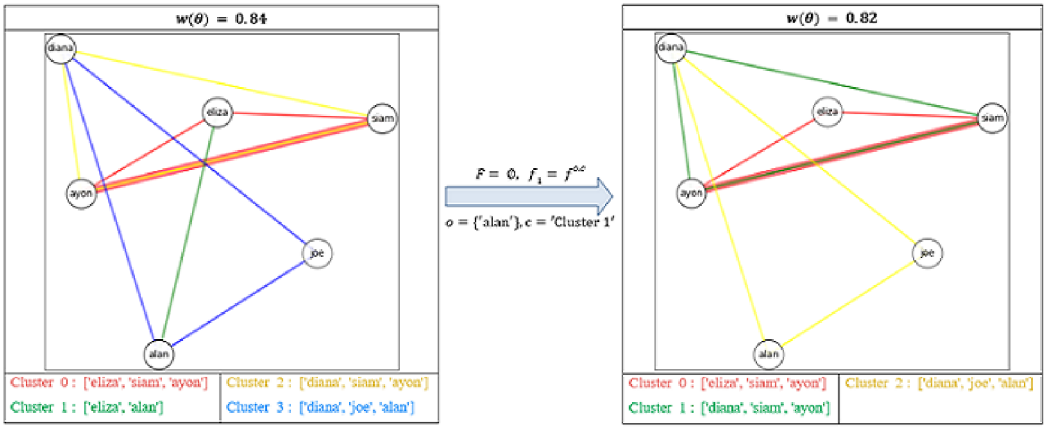


Fig. 7.6 Clustering based on $f^{o,c}$ on a six-people network in *conjunction*.

Weights based on Influence

We expand our experiment using two weighting variables $\theta_1, \theta_2 \in \Theta$ for a user feedback of both feedback categories f^{co} and $f^{o,c}$.

1. **Feedback for constructing a new cluster**
   The experiment starts with a query, $q^{\Theta}(p_i, p_j) = school(p_i, p_j) \wedge (graduation(p_i, p_j) \wedge_{\theta_1,\theta_2} admission(p_i, p_j))$, where $i = j = \{1, 2, 3, ..., N\}$, and is performed as a conjunction ($\wedge$) with equal weights for both attributes, which is set as $w(\theta_1, \theta_2) = \{1, 1\}$ (refer to Figure 7.7 (top left)). If a user gives feedback $f_1 = f^{co}$ requiring $co = \{'diana', 'siam'\}$ to be

in the same cluster, Figure 7.7 (top right) shows that $w(\theta_1, \theta_2) = \{0.53, 0.7\}$ is found to fulfill the feedback and form clusters. This feedback is added to the set $F = \{f_1\}$.

The user then gives a second feedback $f_2 = f^{co}$ requiring $co = \{'eliza', 'alan'\}$ to be in the same cluster, and $w(\theta_1, \theta_2) = \{0.54, 0.68\}$ is found to fulfill this feedback along with the previous feedbacks F and form clusters, as shown in Figure 7.7 (bottom right in second row). This feedback is added to $F = \{f_1, f_2\}$.

After that, the user gives a third feedback $f_3 = f^{co}$ requiring $co = \{'siam', 'alan'\}$ to be in the same cluster, and $w(\theta_1, \theta_2) = \{0.54, 0.66\}$ was found to fulfill this feedback and form clusters, as shown in Figure 7.7 (bottom left in second row). This feedback is added to the set $F = \{f_1, f_2, f_3\}$.

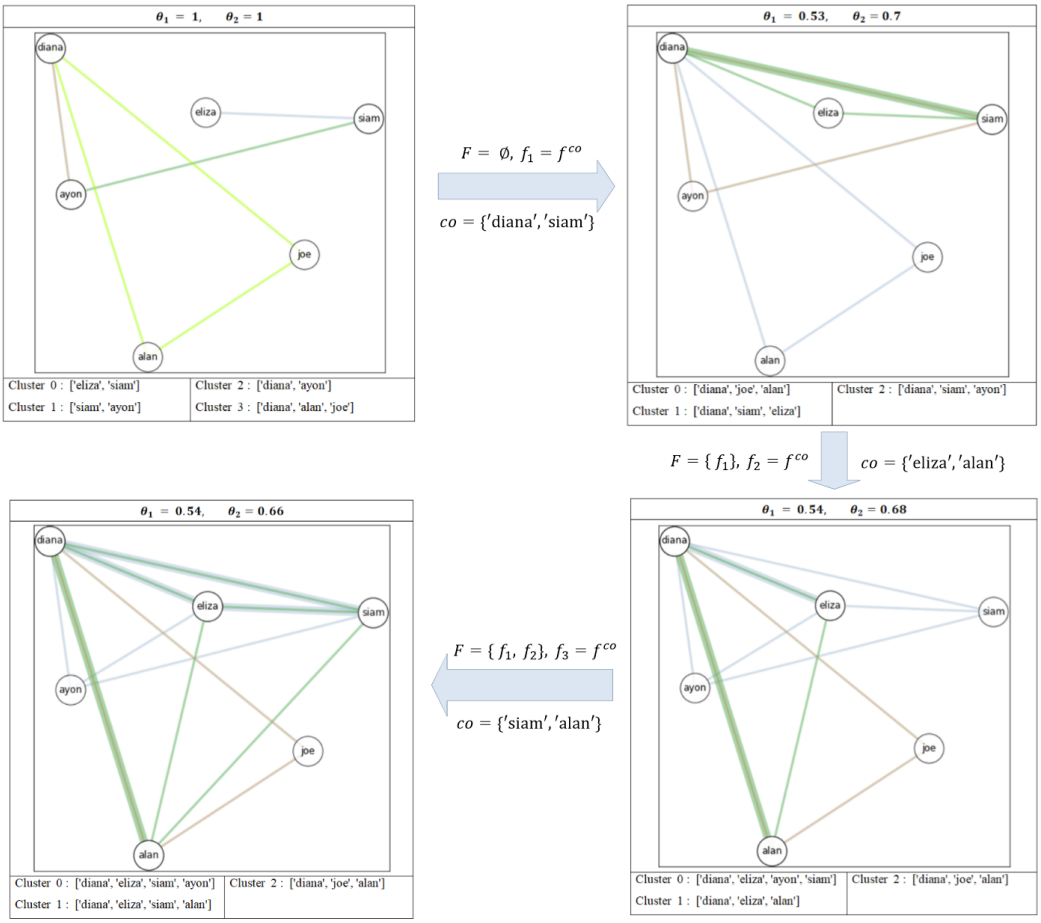


Fig. 7.7 Clustering based on f^{co} on a six-people network in *conjunction*.

To test our approach, we evaluate a clustering distance of detected clustering compared with ground-truth clustering d in every round of our experiment in Figure 7.7. Figure 7.4 is our ideal solution and corresponding clusters of ground-truth clustering are listed in

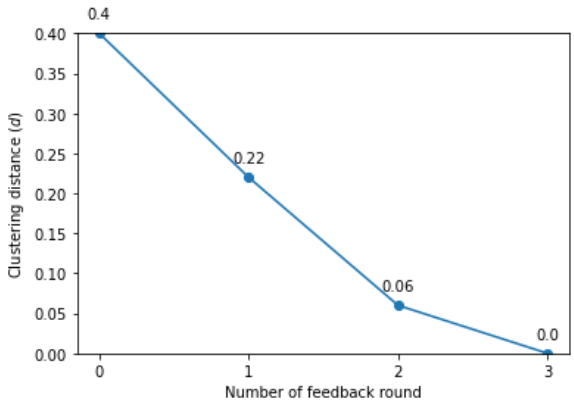


Fig. 7.8 Clustering distances based on f^{co} in Figure 7.7.

Table 7.2. Figure 7.8 shows that the clustering distance d in every round decreases and finally converges to the ideal solution.

2. **Feedback for removing a person from a cluster**
   The next experiment shows user feedback (category $f^{o,c}$) requiring that a person should be outside of a cluster. If the user provides feedback of expecting $o =' alan'$ outside of $c = cluster\ 1$, Figure 7.9 shows that $w(\theta_1, \theta_2) = \{0.75, 0.5\}$ is found that fulfills the feedback by satisfying the constraint and it finds clusters.

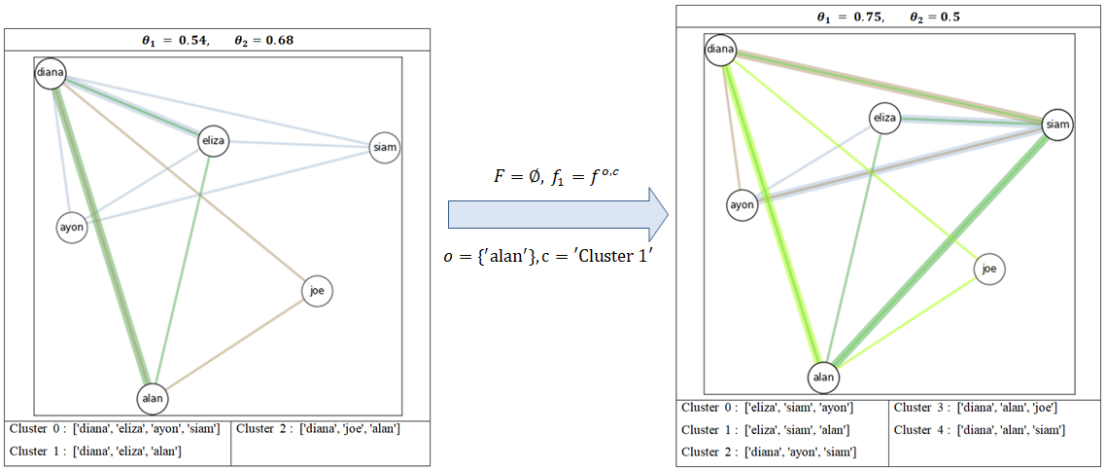


Fig. 7.9 Clustering based on $f^{o,c}$ on a six-people network in *conjunction*.

7.3.3 CQQL query with disjunction ($\vee$)

Alternatively, we can use disjunction ($\vee$) between two attributes, *graduation* and *admission*, in a query $q(p_i,p_j) = school(p_i,p_j) \wedge \underbrace{\Big(graduation(p_i,p_j) \vee admission(p_i,p_j)\Big)}_{disjunction}$ where $i = j = \{1,2,3,...,N\}$. First, we conduct experiments using clustering approaches to detect meaningful clusters and compare the resulting clustering of one approach to the other. Following that, we broaden our experiment by taking user feedback into account in order to re-detect clusters that are relevant to their needs.

Clustering approaches

The first experiment in Figure 7.2 analyzes how clusters can be found by applying both traditional clustering techniques and our proposed Non-TI clustering approach. Figure 7.2a shows the network structure constructed by unweighted query q of Equation 4.18.

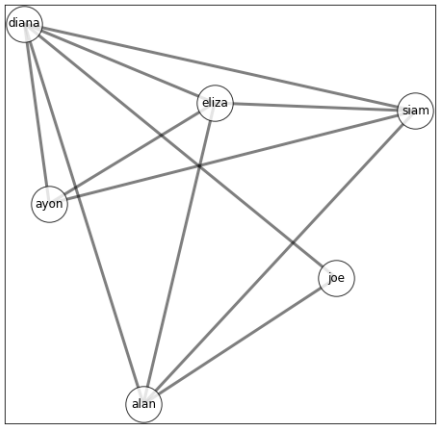
At first we apply k-medoids clustering method with $k = 3$ on the network. Three clusters can be found: $\{eliza\}$, $\{siam, ayon\}$ and $\{alan, joe, diana\}$. Compared with the unweighted network structure, it is clear that the detected clusters do not have any clusters with for example$\{diana, ayon\}$ or $\{eliza, siam\}$ even if they know each other. Therefore, the detected clusters are not meaningful.

A dendogram is created using the single-linkage hierarchical clustering algorithm, as shown in Figure 7.10c. Figure 7.10c demonstrates that a horizontal line (red line) with threshold 0.05 intersects three vertical lines, which means three clusters can be found, as shown in Figure 7.2d.

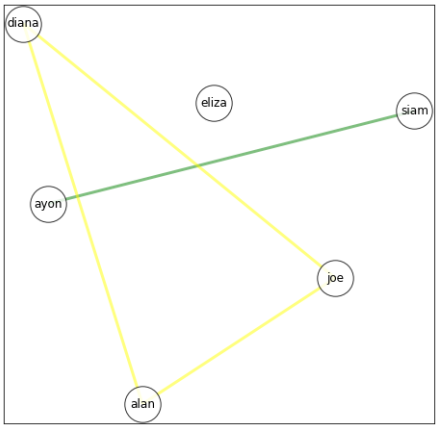
But meaningful clusters can be found by applying our proposed Non-TI clustering approach where intra-cluster objects are close to each other (i.e., they are connected to each other with an edge) as shown in Figure 7.2e. Clustering methods with their corresponding detected clusters are summarized in Table 7.3.

Table 7.3 Clusters after clustering on a six-people network in *disjunction*

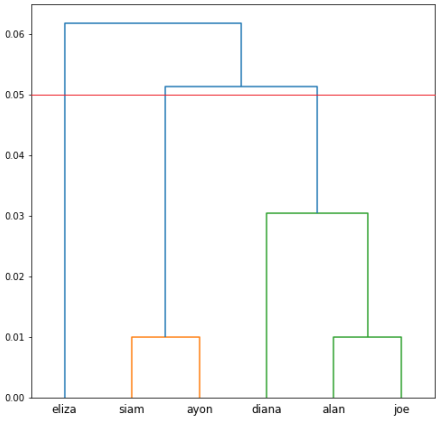
Methods	**Clusters**
k-medoids (k=3)	*{eliza}, {siam, ayon}, {alan, joe, diana}*
Hierarchical (Single-linkage)	*{alan, joe, diana}, {eliza}, {siam, ayon}*
Non-TI	*{diana, alan, eliza, siam}, {diana, alan, joe}, {diana, ayon, eliza, siam}*



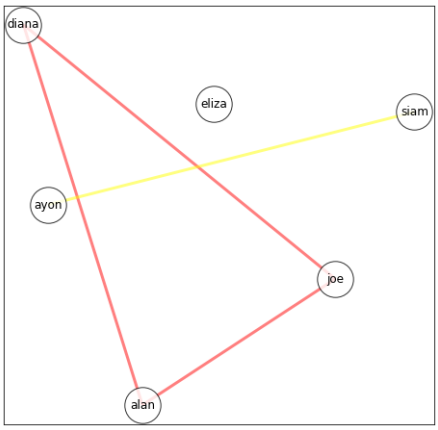
(a) Unweighted



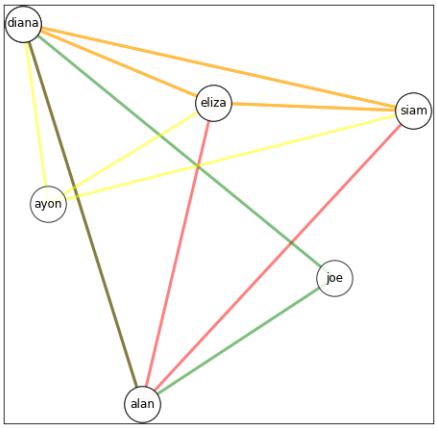
(b) k-medoids ($k = 3$) clustering



(c) Dendogram



(d) Hierarchical clustering



(e) Non-TI

Fig. 7.10 Clustering on a six-people network in *disjunction*.

User interaction

The next experiment analyzes how meaningful clusters can be found with user interaction. We consider both weighting strategies (see Section 4.3) in this experiment as well.

Complimented form of a weight

The logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. Therefore, the experimental procedure and results will be equal for the same parameter settings as described in 7.3.2.

Weights based on Influence

We expand our experiment using two weighting variables $\theta_1, \theta_2 \in \Theta$ for a user feedback of both feedback categories f^{co} and $f^{o,c}$.

1. **Feedback for constructing a new cluster**

   The experiment starts with a query $q^{\Theta}(p_i, p_j) = school(p_i, p_j) \wedge (graduation(p_i, p_j) \vee_{\theta_i,\theta_j} admission(p_i, p_j))$ where $i = j = 1, 2, 3, ..., N$ and $w(\theta_1, \theta_2) = \{0.5, 0.5\}$ while performing disjunction $(\vee)$ (refer to Figure 7.11 top left). The user then provides feedback $f_1 = f^{co}$ that requires $co = \{'diana', 'siam'\}$ to be in the same cluster, which leads to $w(\theta_1, \theta_2) = \{0.71, 0.51\}$ being found as the solution that satisfies the constraint and creates clusters, as shown in Figure 7.11 top right. The feedback is added to the set $F = \{f_1\}$.

   The user continues to provide a second feedback $f_2 = f^{co}$ that requires $co = \{'eliza', 'alan'\}$ to be in the same cluster. $w(\theta_1, \theta_2) = \{0.84, 0.51\}$ is found as the solution that satisfies both the second feedback and the previous feedbacks in the set F, resulting in clusters as shown in Figure 7.11 bottom right in the second row. The feedback is then added to $F = \{f_1, f_2\}$.

   The user provides a third feedback $f_3 = f^{co}$ that requires $co = \{'siam', 'alan'\}$ to be in the same cluster, and $w(\theta_1, \theta_2) = \{0.93, 0.51\}$ is found as the solution that satisfies the constraint and creates clusters, as shown in Figure 7.11 bottom left in the second row. The feedback is added to the set $F = \{f_1, f_2, f_3\}$.

   The user then provides a fourth feedback $f_4 = f^{co}$ that requires $co = \{'eliza', 'diana'\}$ to be in the same cluster, and $w(\theta_1, \theta_2) = \{0.93, 0.54\}$ is found as the solution that satisfies both the fourth feedback and the previous feedbacks in the set F, resulting in clusters as shown in Figure 7.11 bottom right in the second row. The feedback is added to the set $F = \{f_1, f_2, f_3, f_4\}$.

   To test our approach, we evaluate a clustering distance of detected clustering compared with ground-truth clustering d in every round of our experiment in Figure 7.7. Figure 7.4

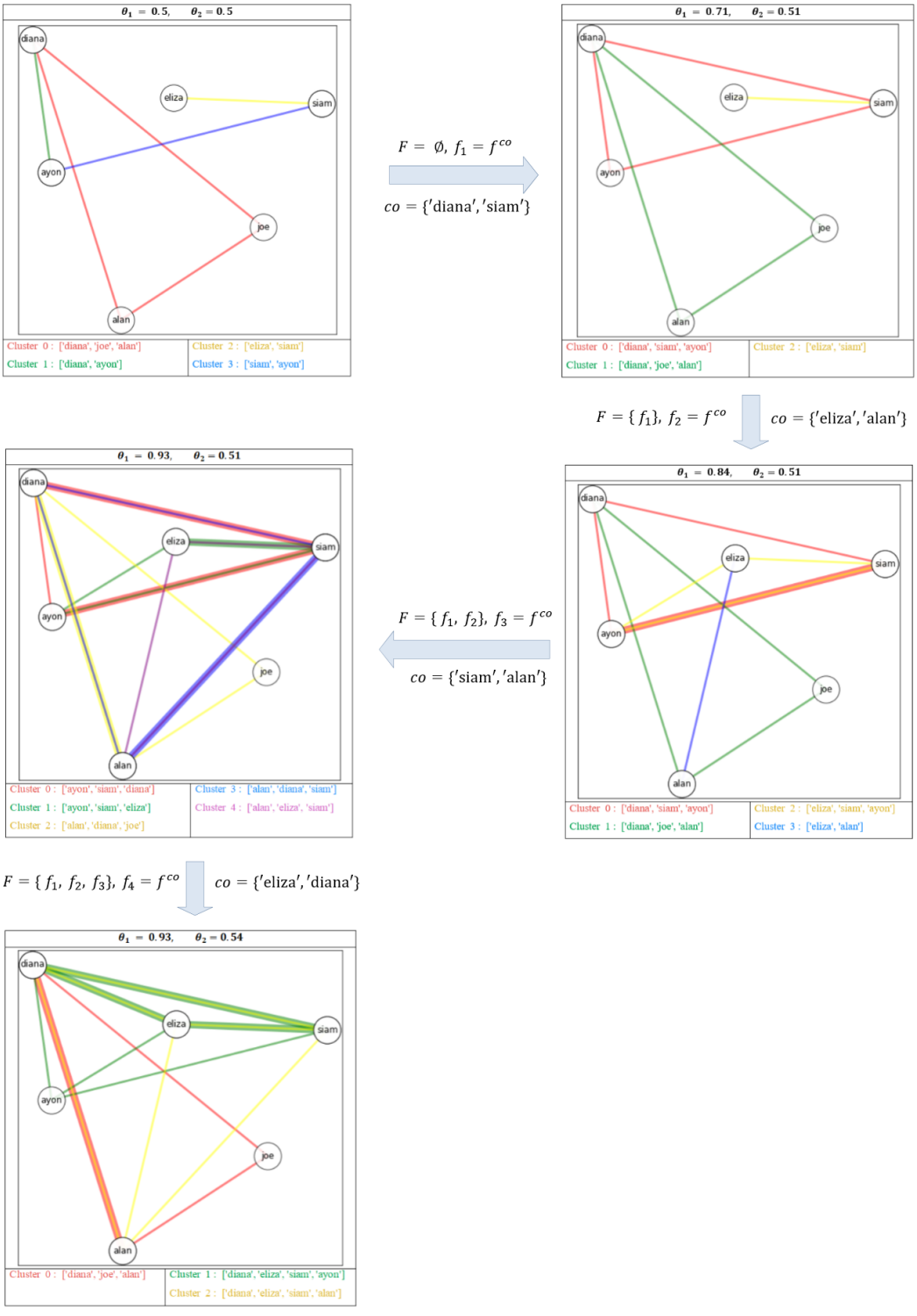


Fig. 7.11 Clustering based on f^{co} on a six-people network in *disjunction*.

is our ideal solution and corresponding clusters of ground-truth clustering are listed in

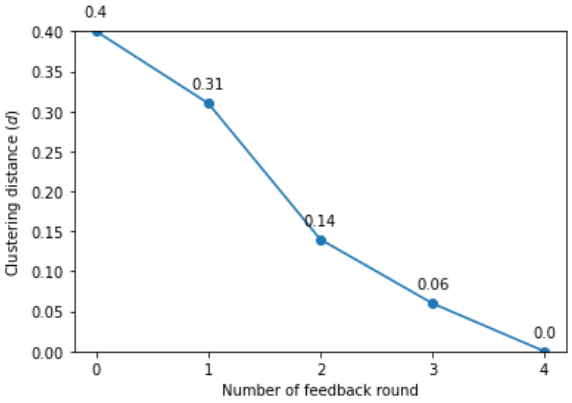


Fig. 7.12 Clustering distances based on f^{co} in 7.11.

Table 7.2. Figure 7.12 shows that the clustering distance d in every round decreases and finally converges to the ideal solution.

2. **Feedback for removing a person from a cluster**

The experiment demonstrates that user feedback in category $f^{o,c}$ requires a person to be outside of a cluster. If the feedback is that a person named $o =' alan'$ should be outside of $c = cluster\ 1$, the figure shown in Figure 7.13 indicates that the weight $w(\theta_1, \theta_2) = \{0.83, 0.51\}$ satisfies this requirement by fulfilling the constraint and locating clusters.

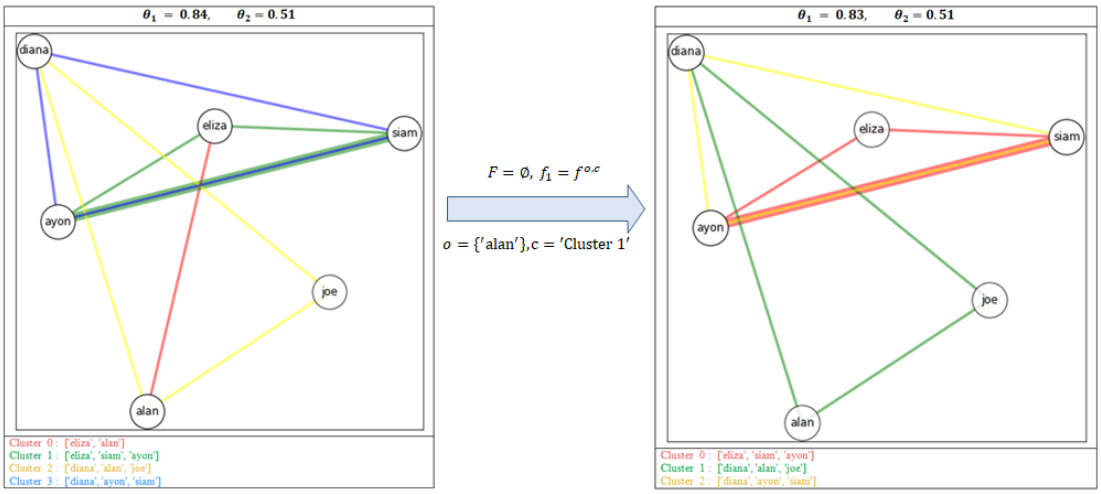


Fig. 7.13 Clustering based on $f^{o,c}$ on a six-people network in *disjunction*.

7.4 Summary

In this chapter, we first calculated similarity values among people in a social network using CQQL, which returns a similarity matrix. A adjacency matrix was derived from the similarity matrix using a threshold value of $th = 0.5$. The Fruchterman-Reingold algorithm was utilized to generate two-dimensional representations of social networks by using the adjacency matrix for visualization purposes.

The traditional clustering techniques like $k-$medoids and hierarchical clustering were applied to the network, however their results may be impacted by the violation of the TI property. To address this issue, our proposed clique-guided non-TI clustering approach was used to detect meaningful clusters.

It is not obvious that the resulting clusters satisfy user needs. Therefore, we extended our non-TI clustering approach by incorporating weights in CQQL conditions based on the influence of the attributes of a person's similarity. The user provides feedback by interacting with the system, and it checks for weights that satisfy the corresponding constraints of the feedback. To evaluate the performance of our relevance feedback based non-TI clustering approach, we computed the clustering distance of the resulting cluster and compared it with the ground-truth clustering.

One limitation of this work is the complexity ($O(3^{n/3})$) of the clique algorithm. We tested our approach on social networks with a maximum size of 1000 people. The complexity will certainly increase with the increase of network sizes. Nevertheless, the promising performance of the proposed approach indicates its potential in the detection of meaningful clusters and indicates its prospects for future developments.

Chapter 8

Conclusions

For analyzing experimental data from a social network, a large collection of clustering algorithms is available. The scientific literature continues to be flooded with new clustering algorithms. Most clustering algorithms work by taking the distances between objects as input and then discovering clusters of objects. However, distance functions may sometimes not satisfy the triangle inequality property, which can greatly impact the quality of the resulting clusters.

In this thesis, we presented three major contributions towards the clustering of a social network: one for detecting clusters based on a non-TI clustering approach; one for adapting clusters with user interaction by providing feedback; and one for evaluating the non-TI approach. Firstly, we proposed a CQQL-based clique-guided non-TI clustering approach that detects meaningful clusters. Secondly, we proposed a weighted CQQL query where weights reflect the influence of similarity conditions based on feedback from user interaction. Finally, we calculated the clustering distance by comparing the detected clustering with ground truth clustering. The clustering distance evaluates the performance of our approach compared with the ideal solution.

The findings of this thesis have been published in three peer-reviewed conference papers [listed in Appendix D]. Below, we summarize our main contributions, research findings and suggest future research directions in social network clustering.

8.1 Contributions and Research Findings

Non-TI clustering approach

Our proposed non-TI clustering approach, based on CQQL, aims to detect meaningful clusters. The approach first calculates the similarity between objects using CQQL, resulting

in a similarity matrix with dimensions of $N \times N$. This similarity matrix is then transformed into an adjacency matrix using a threshold value. Finally, the clique algorithm is applied to detect clusters.

Our proposed approach outperforms traditional clustering algorithms to detect clusters. Traditional clustering algorithms rely on distance functions that satisfy various properties, including the triangle inequality (TI). However, if the distance function violates the TI property, it may negatively impact the quality of the resulting clusters. Despite this violation, our proposed approach still successfully detects meaningful clusters.

Relevance feedback in user interaction

The concept behind using feedback is straightforward: the user is given the chance to change the initial cluster results by transferring objects from one cluster to another or by taking an object out of a cluster. A CQQL query uses weights that are based on the impact of similarity criteria. The main idea is to change and improve these weights through user feedback. As a result, two types of feedback were proposed: (i) feedback for creating a new cluster and (ii) feedback for removing an object from a cluster. The type of feedback depends on the user's interaction and can either be considered *consistent* if the weights do meet the constraints or *inconsistent* if the weights do not meet the constraints.

Each feedback received in every round is added to a feedback set—a set of constraints. This set must be satisfied in every round. If it is not possible to find a weight(s) that satisfies all constraints, the system checks subsets of constraints and provides the user with a set of alternate possible sets of feedback that can be fulfilled.

Clustering distance

We tested our approach by calculating a clustering distance by comparing the resulting clustering with a ground truth clustering. The clustering distance measures the difference between two clusterings. We used the concept of counting pairs of object approach to compare clusterings.

8.2 Future Works

Despite the fact that this thesis made significant contributions to social network clustering research, some limitations remain that offer directions for future research, as discussed below.

Reducing complexity of clique algorithm

The Bron-Kerbosch algorithm with pivoting takes $O(3^{n/3})$ time in the worst case. We performed our experiment by varying node sizes from up to 1000 nodes. More nodes increase running time. The main concern of our thesis is not to reduce the complexity of the clique algorithm. Therefore, reduction of the complexity of the Clique algorithm will be a prominent work in the future. One probable way of reducing the complexity is by weakening the clique condition, for example, by discovering cliques with a size of at least 3 instead of all maximal cliques. Another way is to parallelize or distribute the approach across multiple processors or machines. This can significantly reduce the time it takes to locate all cliques in a social network.

Graphical user interface

Our future work will focus on creating a graphical user interface that helps the user refine queries as a result of the promising outcomes of our experiments. The theoretical core of the user interface will be built on the exploitation of feedback regions and the corresponding weight maxima to provide useful suggestions during the refinement process.

Star Satellite clustering approach

Another clustering approach that can be used to find meaningful clusters based on the distance between nodes is the star satellite clustering approach. The approach starts by choosing one node in the network as the central node (a star), and then grouping all of the nodes that are nearby it (satellites). This process is repeated for each satellite by selecting each one as the new star node and identifying its close neighbors as satellites, until all nodes have been assigned to a cluster. The result is a set of clusters, each of which is centered around a star node and contains the satellites that are close to it. The time complexity of the Star Satellite approach is $O(n+mlogn)$, where n is the number of nodes in the graph and m is the number of edges.

Appendix A

Mathematical Foundations

The purpose of this appendix is to provide an overview of the mathematical foundations needed to comprehend the thesis. It is not meant to substitute for a comprehensive introduction to the relevant topics, particularly in statistics and probability theory, which are more thoroughly covered by Mendenhall et al. [2020], Miller et al. [1999], or Ash [2008].

Several authors, including Bondy and Murty [1976], provide basic overviews of the mathematical concepts of graph theory.

A.1 Numbers, Sets, Relations and Functions

Numbers

The set of natural numbers that includes zero is represented as $\mathbb{N} = \{0, 1, 2, ...\}$. The real numbers are symbolized by $\mathbb{R}$. The non-negative real numbers are referred to as $\mathbb{R}_{\geq 0}$, and the positive real numbers are denoted as $\mathbb{R}^{+}$.

Sets

A *set* is an aggregation of objects known as elements. The elements of a set can be any kind of objects, such as numbers, letters, or even other sets. A set is defined as a collection of distinct elements, and the order of the elements within the set is usually not important. The logical statement "a is a member of the set A" is written as

$$a \in A$$

Likewise, its logical negation "a is not a member of the set A" is written $a \notin A$. Therefore, exactly one of these two statements is true.

Some important sets are:

- $\mathbb{N} = \{0, 1, 2, 3, ...\}$ is the set of natural numbers with 0.
- $\mathbb{Z} = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ is the set of integers.
- $\mathbb{Q}$ is the set of rational numbers.
- $\mathbb{R}$ is the set of real numbers.
- $\mathbb{C}$ is the set of complex numbers.

Empty Set. A set with no elements is called *empty set* (or *null set*, or *void set*), and is represented by $\emptyset$ or $\{\}$.

Subset. A set S is considered a subset of another set A, or contained in A, if all elements of S are also elements of A. This relationship is represented mathematically as $S \subseteq A$. This means that for all elements x in S, it must also be an element of A.

A proper subset of a set A, represented as $S \subset A$, is a subset that satisfies $S \subseteq A$ but is not equal to A. This means that there exists at least one element in A that is not in S. In other words, a proper subset is a subset that is strictly contained within another set.

Definition A.1. For a finite set A, the *cardinality* $|A|$ equals the number of elements in A.

Definition A.2. We say that A *equals* B (denoted $A = B$) if, for all x, $x \in A$ iff $x \in B$. This means that $A = B \iff \forall x((x \in A) \leftrightarrow (x \in B))$.

Ordered Pairs. A set is a collection of distinct elements. An ordinary pair, represented as a, b, is a set with two elements where the order of the elements is irrelevant. This means that $a, b = b, a$ and no distinction is made between the elements in terms of their arrangement.

On the other hand, an ordered pair, represented as (a, b), is a different object used when the order of the elements is relevant. In this case, $(a, b) \neq (b, a)$ unless $a = b$. The elements in an ordered pair are arranged in a specific order, and this order is important.

Two ordered pairs are equal if and only if both the first and second elements are equal. In mathematical terms, $(a, b) = (a', b')$ if and only if $a = a'$ and $b = b'$.

Partition. A *partition* of a set A is a grouping of its elements into non-overlapping and non-empty subsets such that their union is equal to the original set A. In other words, a partition of a set A is a collection of subsets of A such that each element in A belongs to exactly one subset in the collection, and the subsets in the collection cover all elements of A. The subsets in the partition are called "blocks".

Sets of Sets. So far, most of our sets contain atomic elements (such as numbers or strings) or tuples (e.g. pairs of numbers). Sets can also contain other sets. For example, $\{\mathbb{Z}, \mathbb{Q}\}$ is a set containing two infinite sets. $\{\{a, b\}, \{c\}\}$ is a set containing two finite sets.

Set Operations

Let A, B be any two sets.

Definition A.3. The *union* of A and B (denoted $A \cup B$) is the set of elements in either A or B. That means $A \cup B = \{x \in A \text{ or } x \in B\}$ is also defined by

$$x \in A \cup B \iff (x \in A) \vee (x \in B)$$

Definition A.4. The *intersection* of A and B (denoted $A \cap B$) is the set of elements in both A and B. That means $A \cap B = \{x \in A \text{ and } x \in B\}$ is also defined by

$$x \in A \cap B \iff (x \in A) \wedge (x \in B)$$

Two sets are said to be *disjoint* if $A \cap B = \emptyset$.

Definition A.5. The *set difference* between A and B (denoted $A - B$ or $A \setminus B$) is the set of elements in A but not in B. That means

$$x \in A - B \iff (x \in A) \wedge (x \notin B)$$

If there is some implied *universal set* U, then the *complement* (denoted A^c) is defined by $A^c = U - A$.

Definition A.6. The *Cartesian Product*, denoted $A \times B$, of two sets is the set of ordered pairs $(a,b)|a \in A, b \in B$. For $n-$tuples taken from the same set, the notation An denotes the $n-$fold product $A \times A \times ... \times A$.

An example of Cartesian product is the real plane $\mathbb{R}^2$, where $\mathbb{R}$ is the set of real numbers.

Binary Relations

A *binary relation* $\sim$ between elements of the set A is defined by the pairs $(x,y) \in A \times A$ for which the relation holds. Specifically, the binary relation is defined by the subset of ordered pairs $E \subseteq A \times A$ where the relation $a \sim b$ holds; so $x \sim y$ iff $(x,y) \in E$. A binary relation on A is said to be:

1. *Reflexive:* if $x \sim x$ holds for all $x \in A$
2. *Symmetric:* if $x \sim y$ implies $y \sim x$ for all $x, y \in A$
3. *Transitive:* if $x \sim y$ and $y \sim z$, then $x \sim z$ for all $x, y, z \in A$

Functions

A *function* $f : X \to Y$ from X to Y is defined by a subset $F \subset X \times Y$ such that $A_x = \{y \in Y | (x,y) \in F\}$ has exactly one element for each $x \in X$. The value of f at $x \in X$, denoted $f(x)$, is the unique element of Y contained in A_x.

A.2 Vectors and Matrices

Vectors

A $k-$dimensional vector a is an ordered collection of k real numbers $a_1, a_2, ..., a_k$, and is written as $a = \langle a_1, a_2, ..., a_k \rangle$. The numbers a_j where $j = 1, 2, ..., k$ are called the components of the vector a.

Definition A.7. A *dot product* is a way of multiplying two vectors to get a number, or scalar. If $b = \langle b_1, b_2, ..., b_k \rangle$ is also a vector then the dot product of two vectors a and b is defined as

$$a \cdot b = \sum_{j=1}^{k} a_j b_j$$

Definition A.8. The *norm* of a vector, written as $|a|$ or $||a||$ is the square root of the dot product of a with itself, is defined as

$$|a| = \sqrt{a \cdot a}$$

Matrices

A *matrix* is a group of numbers (elements) that are arranged in rows and columns. In general, an $m \times n$ matrix is a rectangular array of mn numbers (or elements) arranged in m rows and n columns. If $m = n$ the matrix is called a *square matrix*. Sometime we use the abbreviation $S = (s_{ij})$ for a matrix *where* s_{ij} would be the element at i^{th} row and j^{th} column.

Definition A.9. A square matrix $S = (s_{ij})$ is called a *symmetric matrix* if $s_{ij} = s_{ji}$, i.e. the elements of the matrix are symmetric with respect to the main diagonal.

Definition A.10. Two matrices P and Q are *equal* if and only if they have the same size $m \times n$ and their corresponding elements are equal.

A.3 Graphs

Throughout this thesis, unless specified otherwise, we consider simple undirected graphs as defined below. The set of all such graphs is symbolized as G.

Definition A.11 (Graph). A graph $G = (V,E)$ is composed of a finite set of vertices $V = 1,2,...,n$ and a finite set of edges $E \subseteq V \times V$, which are pairs of distinct vertices. Two vertices u and v are said to be *adjacent* (or *neighbors*) if $(u,v) \in E$. The *neighborhood* of a vertex v, denoted $N(v)$ (or $N_G(v)$ when the graph needs to be mentioned explicitly), is defined as $N(v) = \{u \in V | (u,v) \in E\}$. The degree of a vertex v is the number of vertices adjacent to it and is denoted as $deg(v)$.

Definition A.12 (Subgraph). A graph $G' = (V',E')$ is considered a *subgraph* of a graph $G = (V,E)$, denoted as $G' \subseteq G$, when $V' \subseteq V$ and $E' \subseteq E$. If G' is a subgraph of G, then G is referred to as a supergraph of G'. If a subgraph $G' \subseteq G$ is not equal to G, it is referred to as proper and written as $G' \subset G$.

Definition A.13 (Clique). A *clique* of a graph, also called a complete subgraph, is a set $C \subseteq V$ of pair-wise adjacent vertices.A clique C is considered *maximal* if there is no vertex in $V \setminus C$ that is connected to all vertices in C..

The order of a graph G is defined as the number of its vertices, and represented as $|G| = |V(G)|$. The size of G is represented by $||G|| = |E(G)|$, which is the number of its edges. If a graph G has $|G| = 0$, it is referred to as an empty graph. The adjacency matrix of a graph G of order n is a symmetric matrix $A \in 0,1^{n \times n}$ that reflects the presence of edges between vertices. The matrix element a_{ij} is equal to 1 if $v_i v_j \in E(G)$, and 0 otherwise, where v_i and v_j are the vertices that are numbered.

Appendix B

Transformation of CQQL Queries

"A CQQL condition (or query) in a specific syntactical form can be evaluated by means of simple, straightforward arithmetics [Schmitt 2008]. The algorithm for transforming an arbitrary CQQL query *e* is given in Figure B.1.

input: CQQL expression e
output: e without overlaps on retrieval attributes

(1) transform expression e into
disjunctive normal form $\hat{x}_1 \vee \ldots \vee \hat{x}_m$
where $\hat{x}_i$ are conjunctions of literals
(2) simplify expression e by applying
idempotence and invertibility rules
(3) if there is an overlap on a retrieval
attribute between some conjunctions $\hat{x}_i$ then
(3a) let o be a literal of an attribute
common to at least two conjunctions
(3b) replace all conjunctions $\hat{x}_i$ of e
with $(o \wedge \hat{x}_i) \vee (\neg o \wedge \hat{x}_i)$
(3c) simplify e by applying idempotence,
invertibility, and absorption and obtain
$e = (o \wedge \hat{x}_1) \vee \ldots \vee (o \wedge \hat{x}_{m_1}) \vee$
$(\neg o \wedge \hat{x}_{m_1+1}) \vee \ldots \vee (\neg o \wedge \hat{x}_{m_2})$
(3d) replace e with $(o \wedge e_1) \vee (\neg o \wedge e_2)$ where
$e_1 = \hat{x}_1 \vee \ldots \vee \hat{x}_{m_1}, e_2 = \hat{x}_{m_1+1} \vee \ldots \vee \hat{x}_{m_2}$
(3e) continue with step (3) for e_1 and e_2
(4) transform innermost disjunctions to
conjunctions and negations by applying
de-Morgan-law

Fig. B.1 Transformation algorithm.

Example B.1. Consider a collection of XML documents about paintings. Each has a textual content description in the `<desc>` tag, a paint technique in the `<technique>` tag, and the century it was created in the `<century>` tag. The query *'retrieve all oil paintings showing evening twilight painted about in sixteenth century'* combines a database query (`technique='oil'`), a text retrieval query (`desc` is about `'evening twilight'`), and a proximity query (`century` *approx* `16th`).

Next, we demonstrate the evaluation. The atoms of the example query are shown in Table B.1. The condition on a painting's textual description is a text retrieval query, the condition on the century of its creation is a proximity query, and the conditions on the three different painting techniques are traditional database queries.

Table B.1 Atomic conditions

Condition
d : `desc='crucifixion'`
c : `century='16th'`
t_1 : `technique='oil'`
t_2 : `technique='pencil'`
t_3 : `technique='watercolor'`

In the example query shown in Figure B.2, we search for crucifixion paintings or watercolor paintings. If created in the sixteenth century, the crucifixion should be painted in oil, otherwise in pencil. Figure B.2 shows the transformation algorithm in detail as well as the final arithmetic evaluation formula."

[Schmitt et al. 2008]

$$q = (d \wedge ((c \wedge t_1) \vee (\neg c \wedge t_2))) \vee t_3$$

$$\downarrow (1)(2)$$

$$(c \wedge d \wedge t_1) \vee (\neg c \wedge d \wedge t_2) \vee t_3$$

$$\downarrow (3a)(3b)(3c),\ o = c$$

$$(c \wedge d \wedge t_1) \vee (c \wedge t_3) \vee (\neg c \wedge d \wedge t_2) \vee (\neg c \wedge t_3)$$

$$\downarrow (3d)$$

$$(c \wedge ((d \wedge t_1) \vee t_3)) \vee (\neg c \wedge ((d \wedge t_2) \vee t_3))$$

$$\downarrow (4)$$

$$(c \wedge \neg(\neg(d \wedge t_1) \wedge \neg t_3)) \vee (\neg c \wedge \neg(\neg(d \wedge t_2) \wedge \neg t_3))$$

arithmetic evaluation w.r.t. tuple t:

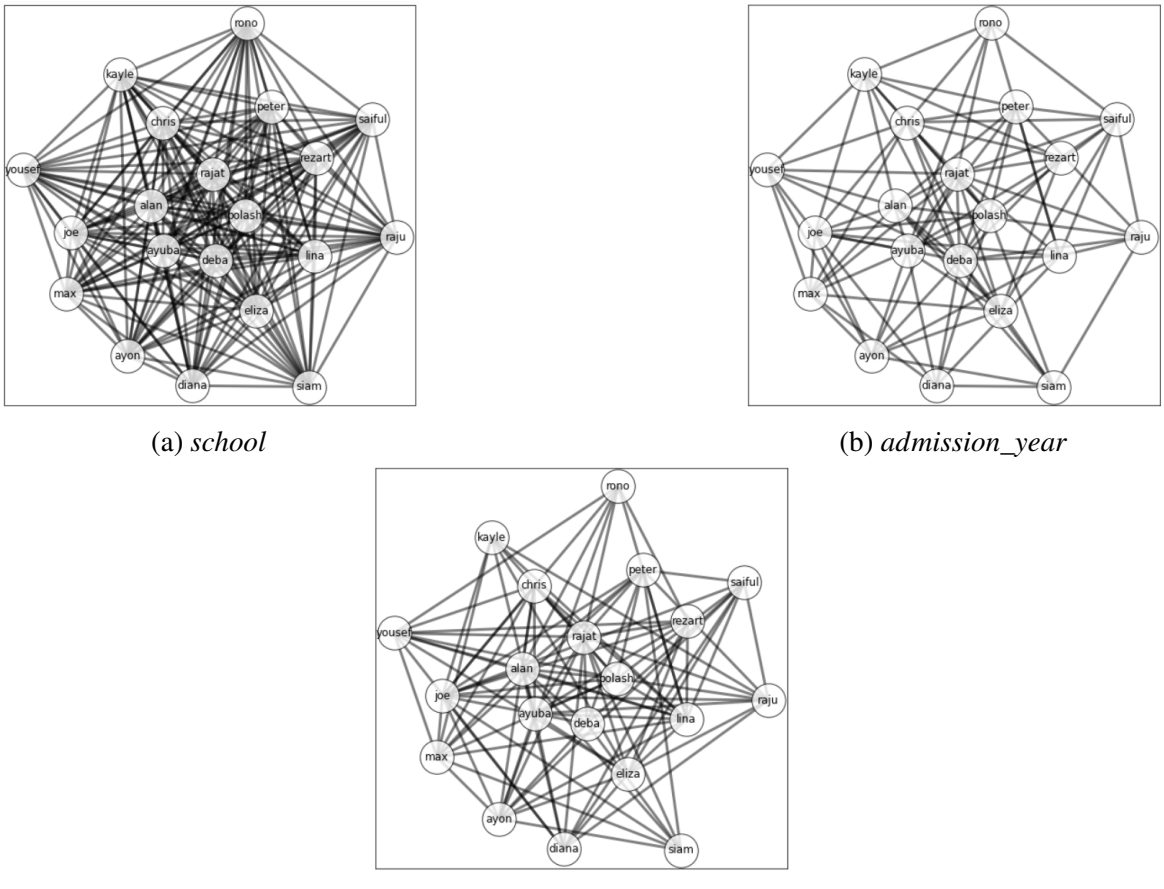
$$\begin{aligned}\mathrm{eval}^t(q) = {} & c^t \left(1 - \left(1 - d^t t_1^t\right)\left(1 - t_3^t\right)\right) \\ & + \left(1 - c^t\right)\left(1 - \left(1 - d^t t_2^t\right)\left(1 - t_3^t\right)\right)\end{aligned}$$

Fig. B.2 Example transformations and arithmetic evaluation.

Appendix C

Experimental Results

C.1 Clustering of a social network (20 people)



(a) *school*

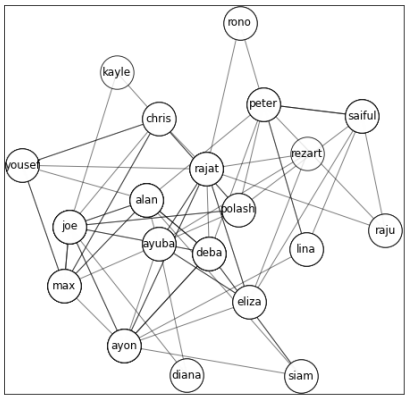
(b) *admission_year*

(c) *graduation_year*

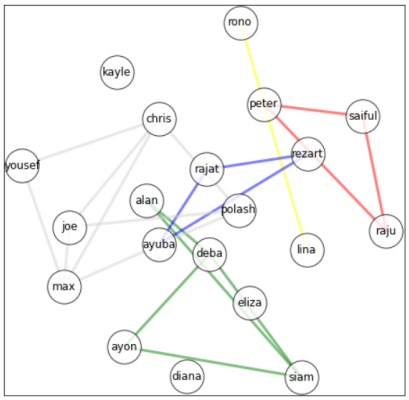
Fig. C.1 Structure of a twenty-people network.

C.1.1 CQQL query with conjunction (∧)

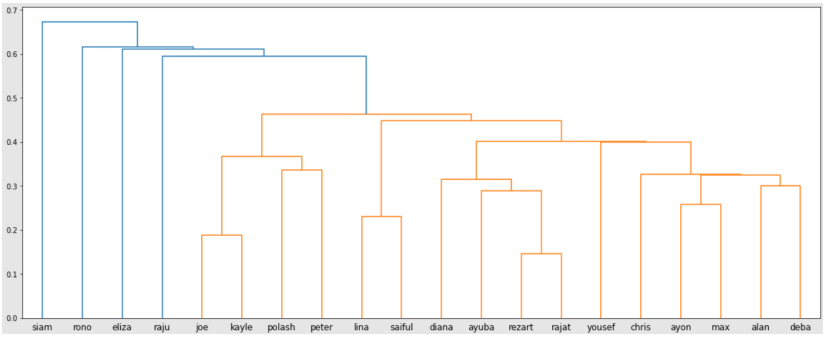
Clustering approaches



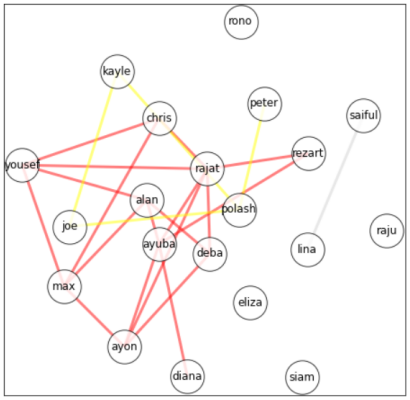
(a) Unweighted



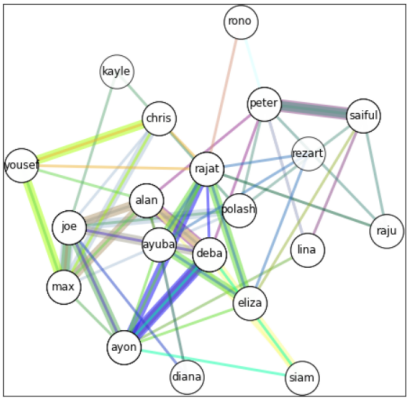
(b) k-medoids ($k = 7$) clustering



(c) Dendogram of Figure C.2d



(d) Hierarchical ($k = 7$) clustering



(e) Non-TI

Fig. C.2 Clustering on a twenty-people network in *conjunction*.

Table C.1 Clusters after clustering on a twenty-people network in *conjunction*

Methods	**Clusters**
k-medoids (k=7)	*{raju, saiful, peter}, {siam, ayon, alan, deba}, {lina, rono}, {ayuba, rezart, rajat}, {eliza}, {diana, kayle}, {joe, yousef, polash, max, chris}*
Hierarchical (Single-linkage k=7)	*{ayon, alan, diana, ayuba, deba, rezart, yousef, rajat, max, chris}, {raju}, {joe, kayle, polash, peter}, {siam}, {eliza}, {rono}, {lina, saiful}*
Non-TI	*{kayle, polash, joe}, {joe, max, polash, chris}, {joe, max, ayon}, {joe, max, alan}, {joe, diana}, {joe, deba, alan}, {joe, deba, ayon}, {rajat, ayuba, eliza, ayon}, {rajat, ayuba, eliza, rezart}, {rajat, rono}, {rajat, yousef, chris}, {rajat, raju}, {rajat, deba, ayon}, {polash, peter, saiful}, {alan, yousef, max}, {alan, peter, deba}, {alan, ayuba}, {alan, siam, deba}, {saiful, peter, raju}, {saiful, peter, lina}, {saiful, eliza}, {lina, ayon}, {lina, rono}, {siam, ayon, deba}, {diana, ayuba}, {max, yousef, chris}*

User interaction

Complimented form of a weight

Feedback for constructing a new cluster

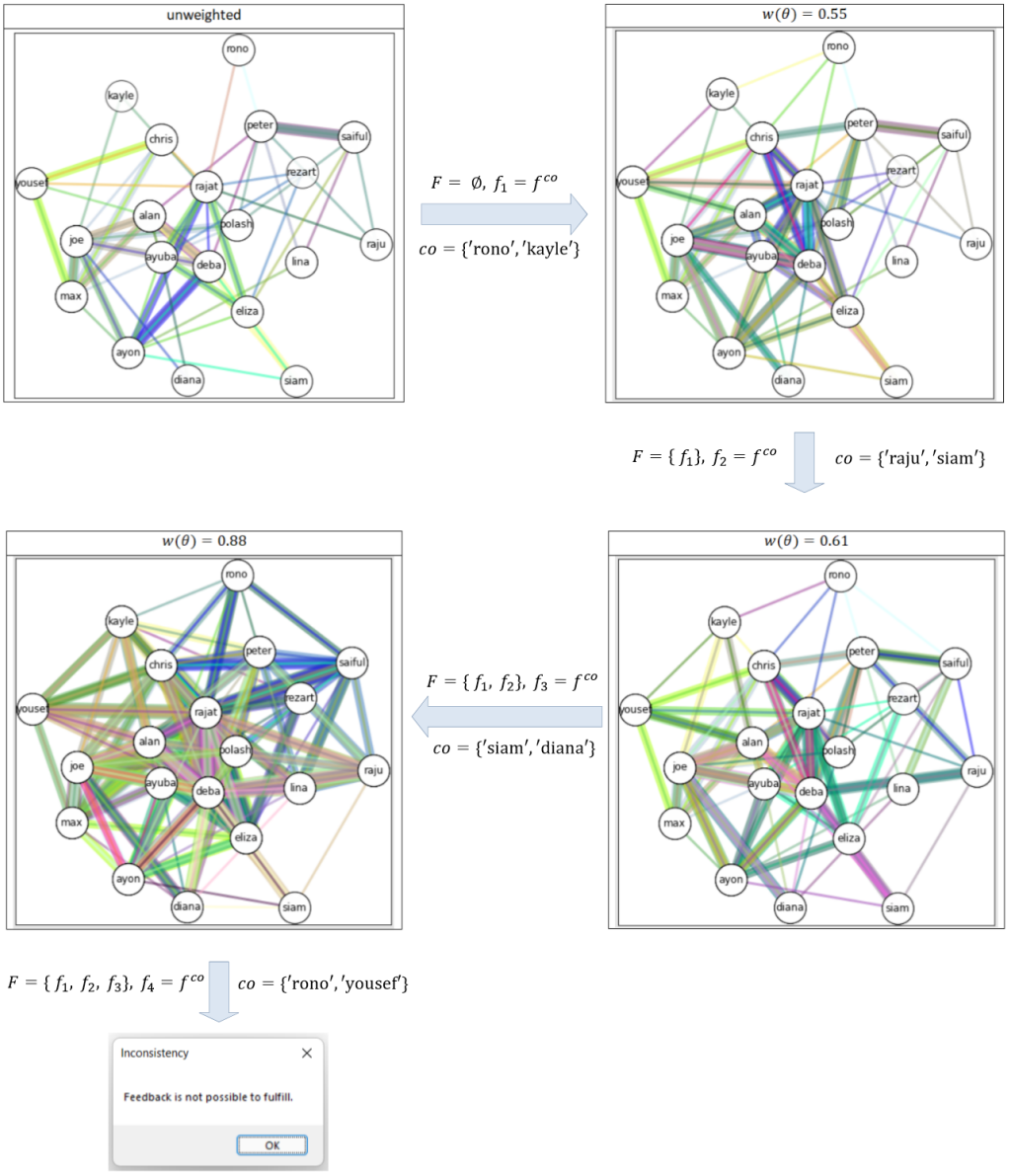


Fig. C.3 Clustering based on f^{co} on a twenty-people network in *conjunction*.

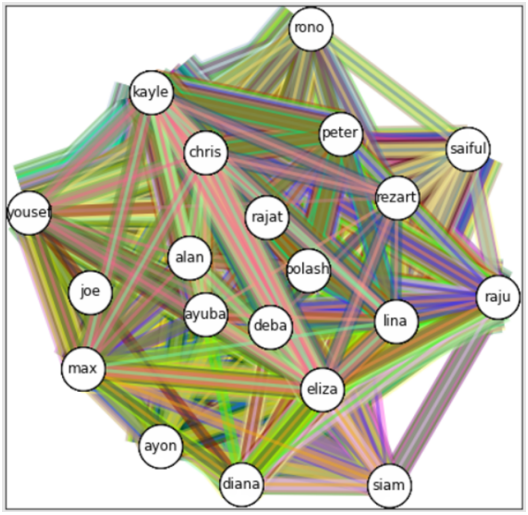


Fig. C.4 Ideal Solution.

Table C.2 Ground-truth clustering

Clusters
{joe, rono, chris, alan, kayle, ayuba}, {joe, rono, chris, alan, kayle, yousef}, {joe, rono, chris, lina, ayon, saiful}, {joe, rono, chris, lina, ayon, ayuba}, {joe, rono, chris, lina, rezart, saiful}, {joe, rono, chris, lina, rezart, ayuba, kayle}, {joe, rono, chris, yousef, ayon}, {joe, rono, chris, yousef, rezart, kayle}, {joe, deba, polash, raju, alan, kayle}, {joe, deba, polash, raju, alan, siam}, {joe, deba, polash, raju, lina, saiful, diana}, {joe, deba, polash, raju, lina, kayle}, {joe, deba, polash, raju, siam, diana} ,{joe, deba, polash, yousef, alan, kayle, chris} ,{joe, deba, polash, yousef, diana} ,{joe, deba, polash, chris, lina, saiful} ,{joe, deba, polash, chris, lina, kayle} ,{joe, deba, ayon, siam} ,{joe, deba, ayon, chris, saiful, lina} ,{joe, deba, ayon, chris, yousef} ,{joe, deba, rezart, lina, saiful, diana} ,{joe, deba, rezart, lina, saiful, chris} ,{joe, deba, rezart, lina, kayle, chris} ,{joe, deba, rezart, yousef, diana} , {joe, deba, rezart, yousef, kayle, chris} , {joe, ayuba, alan, kayle, polash, raju} , {joe, ayuba, alan, kayle, polash, chris} , {joe, ayuba, lina, raju, polash, diana} , {joe, ayuba, lina, raju, polash, kayle} , {joe, ayuba, lina, diana, rezart} , {joe, ayuba, lina, chris, polash, kayle} , {joe, max, polash, chris, kayle, alan, yousef} , {joe, max, polash, chris, kayle, lina} , {joe, max, polash, diana, lina} , {joe, max, polash, diana, yousef} , {joe, max, polash, diana, siam} , {joe, max, polash, siam, alan} , {joe, max, ayon, chris, lina} , {joe, max, ayon, chris, yousef} , {joe, max, ayon, siam} , {rajat, raju, deba, peter, polash, saiful, lina} ,{rajat, raju, deba, peter, polash, alan, siam} , {rajat, raju, deba, eliza, saiful, lina} , {rajat, raju, deba, eliza, alan, siam} , {rajat, raju, ayuba, alan, polash, peter} , {rajat, raju, ayuba, alan, eliza} , {rajat, raju, ayuba, lina, polash, peter} , {rajat, raju, ayuba, lina, eliza} , {rajat, chris, alan, peter, polash, deba} ,{rajat, chris, alan, peter, polash, ayuba} , {rajat, chris, alan, peter, polash, max} , {rajat, chris, alan, ayuba, eliza} , {rajat, chris, alan, ayuba, rono} , {rajat, chris, alan, yousef, rono} , {rajat, chris, alan, yousef, deba, polash} , {rajat, chris, alan, yousef, deba, eliza} , {rajat, chris, alan, yousef, max, polash} , {rajat, chris, alan, yousef, max, eliza} , {rajat, chris, lina, saiful, deba, polash, peter} , {rajat, chris, lina, saiful, deba, eliza, ayon} , {rajat, chris, lina, saiful, deba, eliza, rezart} , {rajat, chris, lina, saiful, rono, ayon} , {rajat, chris, lina, saiful, rono, rezart} , {rajat, chris, lina, ayuba, polash, peter} , {rajat, chris, lina, ayuba, rono, ayon} , {rajat, chris, lina, ayuba, rono, rezart} , {rajat, chris, lina, ayuba, eliza, ayon} , {rajat, chris, lina, ayuba, eliza, rezart} , {rajat, chris, lina, max, polash, peter} , {rajat, chris, lina, max, ayon, eliza} , {rajat, chris, yousef, ayon, eliza, deba} , {rajat, chris, yousef, ayon, eliza, max} , {rajat, chris, yousef, ayon, rono} , {rajat, chris, yousef, rezart, deba, eliza} ,{rajat, chris, yousef, rezart, rono} , {rajat, siam, deba, ayon, eliza} , {rajat, siam, max, ayon, eliza} , {rajat, siam, max, alan, polash, peter} , {rajat, siam, max, alan, eliza} , {diana, eliza, deba, lina, saiful, raju} , {diana, eliza, deba, lina, saiful, rezart} , {diana, eliza, deba, yousef, rezart} , {diana, eliza, deba, siam, raju} , {diana, eliza, ayuba, lina, raju} , {diana, eliza, ayuba, lina, rezart} , {diana, eliza, max, lina}

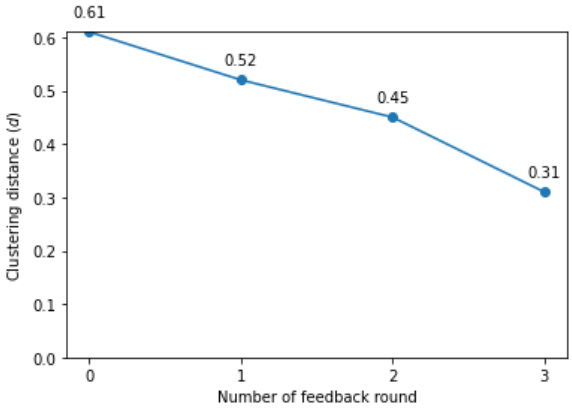


Fig. C.5 Clustering distances based on $f^{o,c}$ in Figure C.3.

Feedback for removing a person from a cluster

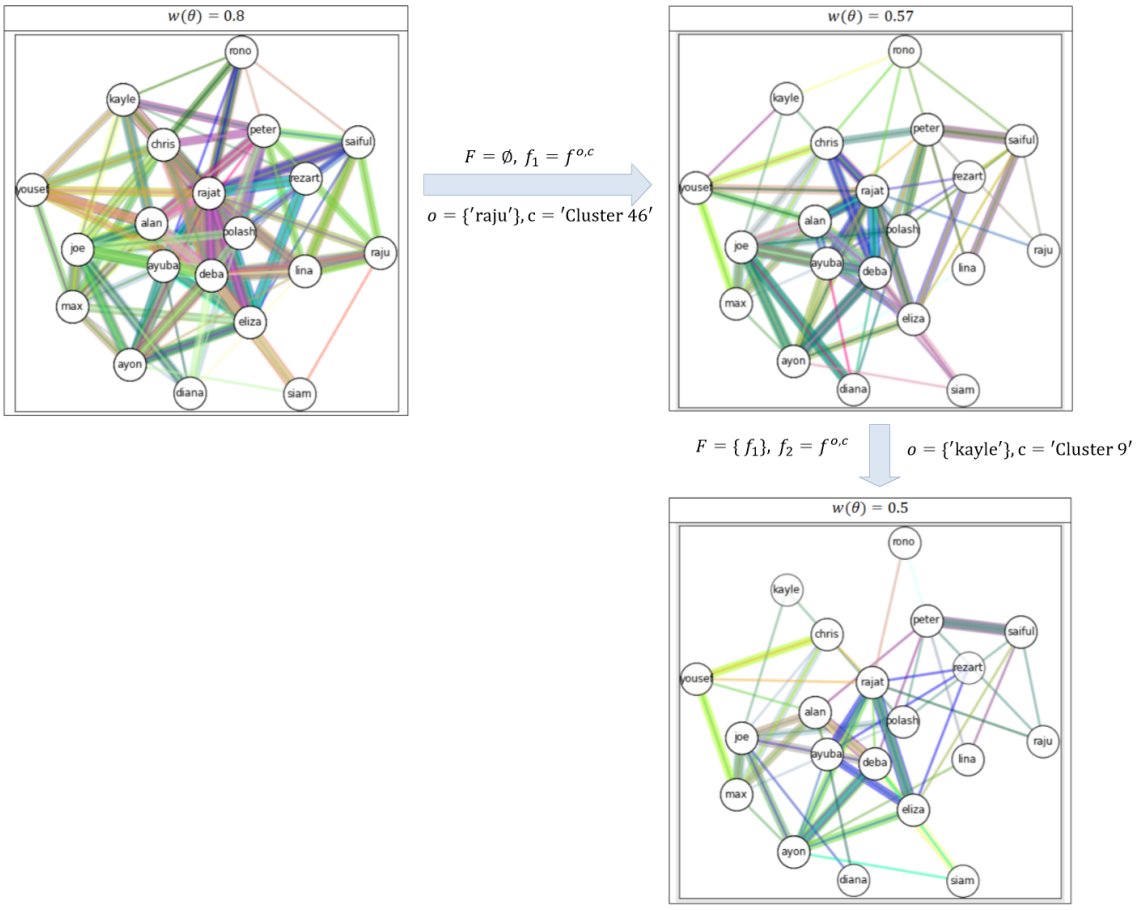


Fig. C.6 Clustering based on $f^{o,c}$ on a twenty-people network in *conjunction*.

Weights based on influence

Feedback for constructing a new cluster

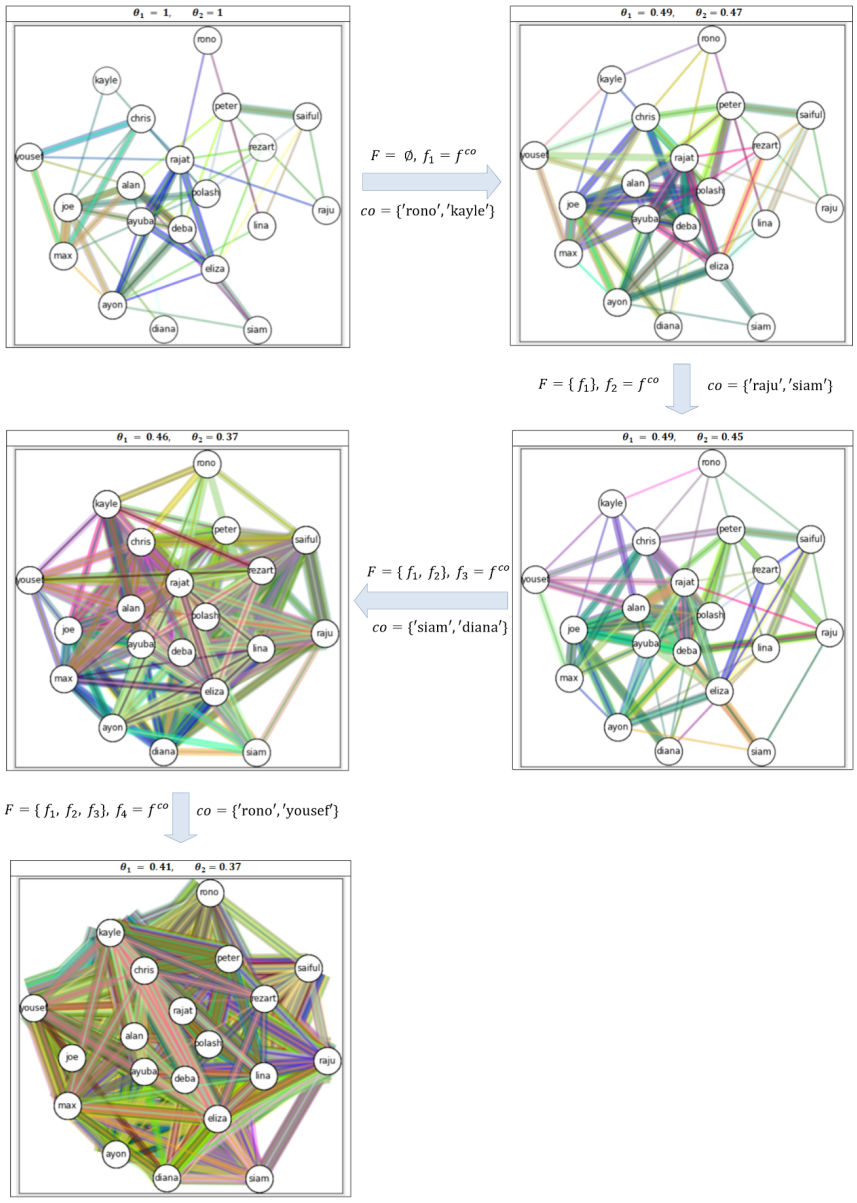


Fig. C.7 Clustering based on f^{co} on a twenty-people network in *conjunction*.

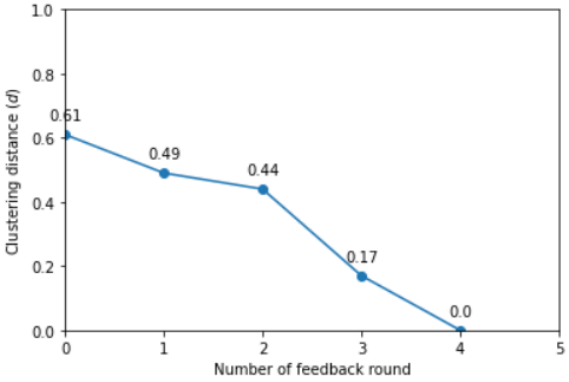
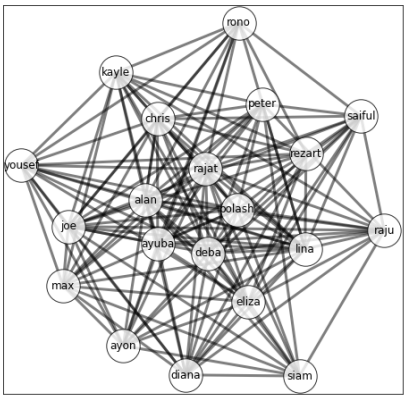


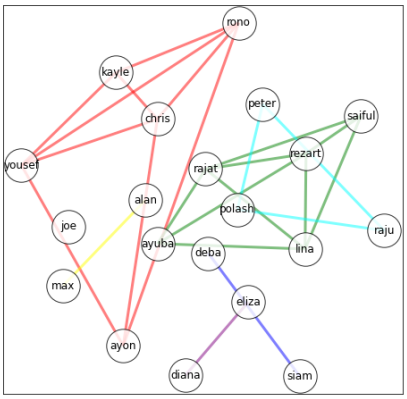
Fig. C.8 Clustering distances based on $f^{o,c}$ in Figure C.7.

C.1.2 CQQL query with disjunction (∨)

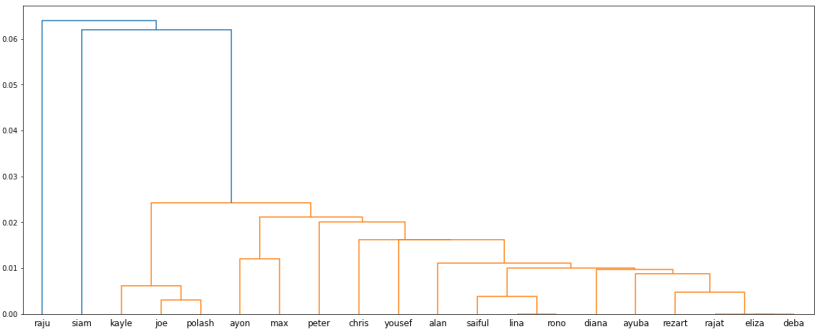
Clustering approaches



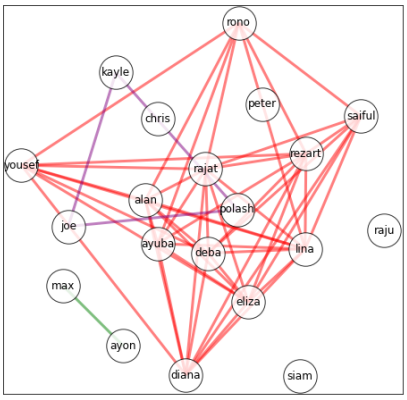
(a) Unweighted



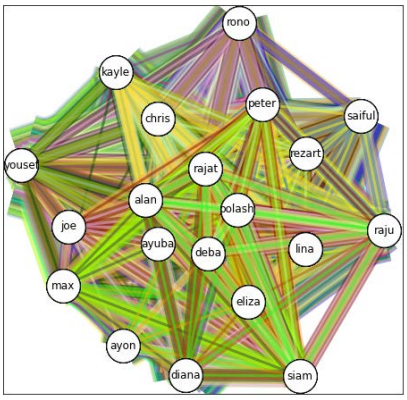
(b) k-medoids ($k = 7$) clustering



(c) Dendogram of Figure C.9d



(d) Hierarchical ($k = 7$) clustering



(e) Non-TI

Fig. C.9 Clustering on a twenty-people network in *disjunction*.

Table C.3 Clusters after clustering on a twenty-people network in *disjunction*

Methods	**Clusters**
k-medoids (k=7)	*{ayon, kayle, yousef, rono, chris}, {lina, ayuba, rezart, saiful, rajat}, {alan, max}, {siam, deba}, {eliza, diana}, {raju, polash, peter}, {joe}*
Hierarchical (Single-linkage k=7)	*{alan, eliza, diana, lina, ayuba, deba, rezart, yousef, rono, saiful, rajat}, {ayon, max}, {chris}, {raju}, {joe, kayle, polash}, {siam}, {peter}*
Non-TI	*{lina, joe, chris, kayle, rezart, yousef, deba}, {lina, joe, chris, kayle, rezart, yousef, rono}, {lina, joe, chris, kayle, rezart, ayuba, rono}, {lina, joe, chris, kayle, alan, polash, yousef, max}, {lina, joe, chris, kayle, alan, polash, yousef, deba}, {lina, joe, chris, kayle, alan, polash, peter, ayuba}, {lina, joe, chris, kayle, alan, polash, peter, max}, {lina, joe, chris, kayle, alan, polash, peter, deba}, {lina, joe, chris, kayle, alan, rono, yousef}, {lina, joe, chris, kayle, alan, rono, ayuba}, {lina, joe, chris, ayon, yousef, max}, {lina, joe, chris, ayon, yousef, deba}, {lina, joe, chris, ayon, yousef, rono}, {lina, joe, chris, ayon, ayuba, rono}, {lina, joe, chris, ayon, saiful, deba}, {lina, joe, chris, ayon, saiful, rono}, {lina, joe, chris, saiful, deba, rezart}, {lina, joe, chris, saiful, deba, polash, peter}, {lina, joe, chris, saiful, rono, rezart}, {lina, joe, diana, rezart, ayuba}, {lina, joe, diana, rezart, deba, yousef}, {lina, joe, diana, rezart, deba, saiful}, {lina, joe, diana, polash, saiful, deba, raju}, {lina, joe, diana, polash, alan, ayuba, raju}, {lina, joe, diana, polash, alan, max, yousef}, {lina, joe, diana, polash, alan, deba, yousef}, {lina, joe, diana, polash, alan, deba, raju}, {lina, joe, raju, polash, peter, kayle, alan, ayuba}, {lina, joe, raju, polash, peter, kayle, alan, deba}, {lina, joe, raju, polash, peter, saiful, deba}, {lina, rajat, chris, ayuba, ayon, eliza}, {lina, rajat, chris, ayuba, ayon, rono}, {lina, rajat, chris, ayuba, rezart, eliza}, {lina, rajat, chris, ayuba, rezart, rono}, {lina, rajat, chris, ayuba, alan, eliza}, {lina, rajat, chris, ayuba, alan, polash, peter}, {lina, rajat, chris, ayuba, alan, rono}, {lina, rajat, chris, max, yousef, ayon, eliza}, {lina, rajat, chris, max, yousef, alan, eliza}, {lina, rajat, chris, max, yousef, alan, polash}, {lina, rajat, chris, max, peter, polash, alan}, {lina, rajat, chris, deba, yousef, eliza, ayon}, {lina, rajat, chris, deba, yousef, eliza, rezart}, {lina, rajat, chris, deba, yousef, eliza, alan}, {lina, rajat, chris, deba, yousef, polash, alan}, {lina, rajat, chris, deba, peter, polash, saiful}, {lina, rajat, chris, deba, peter, polash, alan}, {lina, rajat, chris, deba, saiful, eliza, ayon}, {lina, rajat, chris, deba, saiful, eliza, rezart}, {lina, rajat, chris, rono, yousef, ayon}, {lina, rajat, chris, rono, yousef, rezart}, {lina, rajat, chris, rono, yousef, alan}, {lina, rajat, chris, rono, saiful, ayon}, {lina, rajat, chris, rono, saiful, rezart}, {lina, rajat, diana, eliza, ayuba, rezart}, {lina, rajat, diana, eliza, ayuba, alan, raju}, {lina, rajat, diana, eliza, max, yousef, alan}, {lina, rajat, diana, eliza, deba, yousef, rezart}, {lina, rajat, diana, eliza, deba, yousef, alan}, {lina, rajat, diana, eliza, deba, rezart, saiful}, {lina, rajat, diana, eliza, deba, raju, saiful}, {lina, rajat, diana, eliza, deba, raju, alan}, {lina, rajat, diana, polash, saiful, deba, raju}, {lina, rajat, diana, polash, alan, ayuba, raju}, {lina, rajat, diana, polash, alan, max, yousef}, {lina, rajat, diana, polash, alan, deba, yousef}, {lina, rajat, diana, polash, alan, deba, raju}, {lina, rajat, raju, peter, polash, ayuba, alan}, {lina, rajat, raju, peter, polash, deba, saiful}, {lina, rajat, raju, peter, polash, deba, alan}, {lina, eliza, kayle, chris, yousef, max, alan}, {lina, eliza, kayle, chris, yousef, deba, rezart}, {lina, eliza, kayle, chris, yousef, deba, alan}, {lina, eliza, kayle, chris, ayuba, rezart}, {lina, eliza, kayle, chris, ayuba, alan}, {lina, eliza, kayle, raju, alan, ayuba}, {lina, eliza, kayle, raju, alan, deba}, {siam, joe, max, ayon}, {siam, joe, max, polash, alan, peter}, {siam, joe, max, polash, alan, diana}, {siam, joe, deba, ayon}, {siam, joe, deba, raju, polash, alan, peter}, {siam, joe, deba, raju, polash, alan, diana}, {siam, rajat, max, ayon, eliza}, {siam, rajat, max, alan, eliza, diana}, {siam, rajat, max, alan, polash, peter}, {siam, rajat, max, alan, polash, diana}, {siam, rajat, deba, ayon, eliza}, {siam, rajat, deba, raju, alan, eliza, diana}, {siam, rajat, deba, raju, alan, polash, peter}, {siam, rajat, deba, raju, alan, polash, diana}*

User interaction

Complimented form of a weight

The logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. Therefore, the experimental procedure and results will be equal for the same parameter settings as described in C.1.1.

Weights based on influence

Feedback for constructing a new cluster

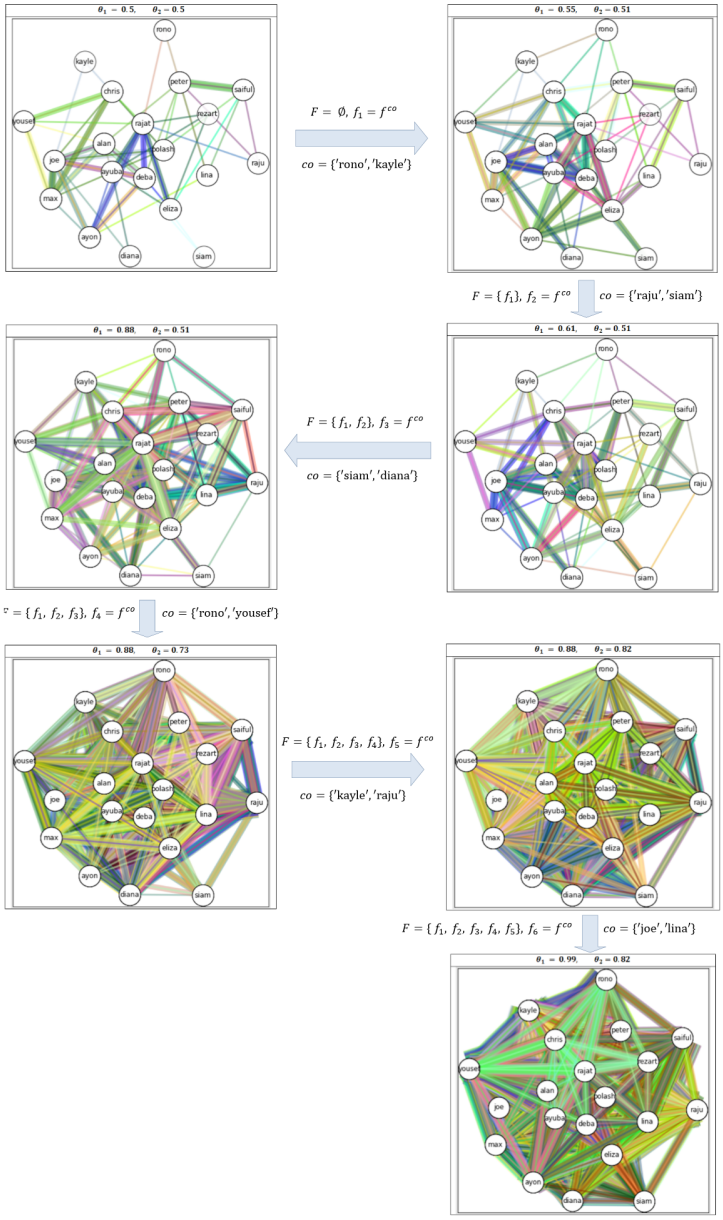


Fig. C.10 Clustering based on f^{co} on a twenty-people network in *disjunction*.

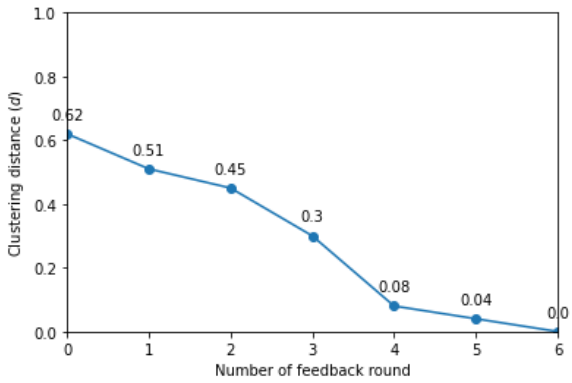


Fig. C.11 Clustering distances based on f^{co} in Figure C.10.

Feedback for removing a person from a cluster

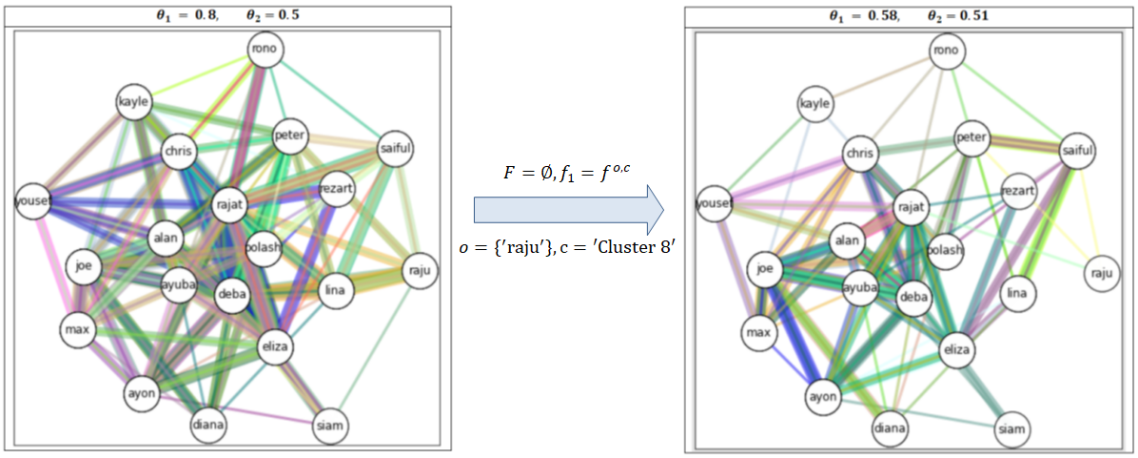
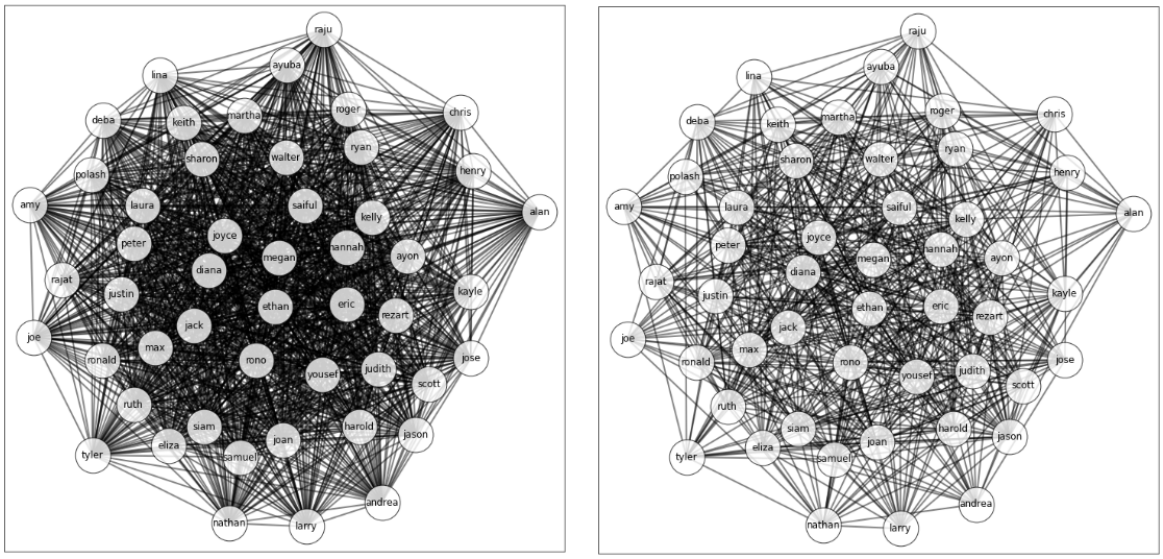
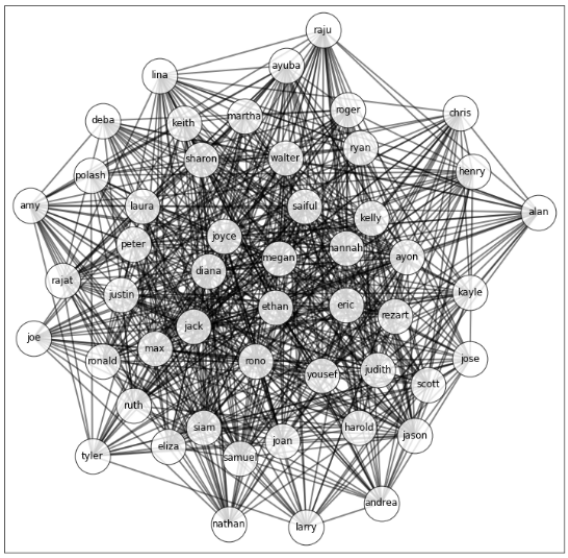


Fig. C.12 Clustering based on $f^{o,c}$ on a twenty-people network in *disjunction*.

C.2 Clustering of a social network (50 people)



(a) *school* (b) *admission_year*

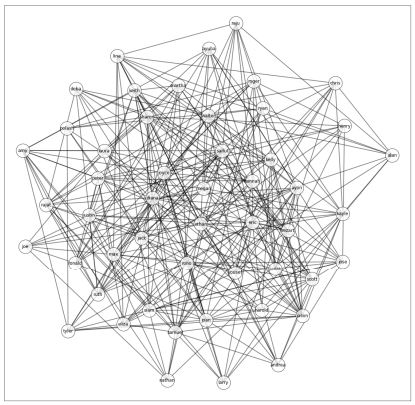


(c) *graduation_year*

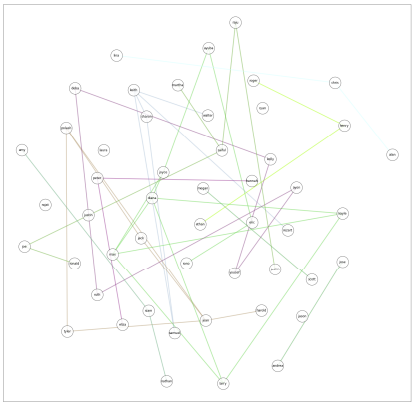
Fig. C.13 Structure of a fifty-people network.

C.2.1 CQQL query with conjunction (∧)

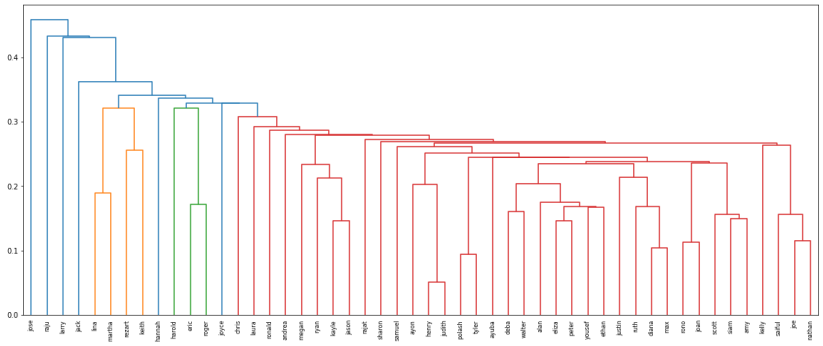
Clustering approaches



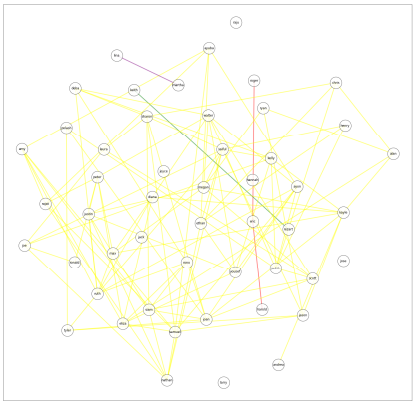
(a) Unweighted



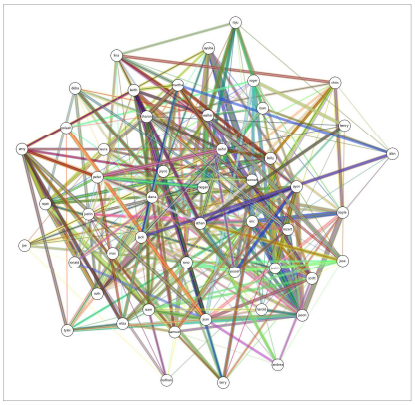
(b) k-medoids ($k = 10$) clustering



(c) Dendogram of Figure C.14d



(d) Hierarchical (single-linkage $k = 10$) clustering



(e) Non-TI

Fig. C.14 Clustering on a fifty-people network in *conjunction*.

Table C.4 Clusters after clustering on a fifty-people network in *conjunction*

Methods	**Clusters**
k-medoids (k=10)	*{rezart, justin, sharon, samuel, walter, keith}, {scott, jose, megan, andrea}, {polash, ryan, jack, tyler, joan, harold}, {henry, ethan, roger}, {diana, ayuba, kayle, rono, max, eric, larry, joyce}, {ayon, deba, yousef, rajat, ruth, kelly}, {joe, raju, saiful, ronald, judith, martha}, {alan, lina, chris, jason}, {siam, amy, nathan}, {eliza, peter, laura, hannah}*
Hierarchical (Single-linkage k=10)	*{eric, harold, roger}, {rezart, keith}, {siam, ayon, alan, joe, eliza, diana, ayuba, deba, kayle, yousef, polash, rono, saiful, rajat, max, peter, chris, ryan, jason, ronald, amy, justin, sharon, scott, laura, samuel, ruth, tyler, henry, nathan, kelly, ethan, joan, walter, megan, judith, andrea}, {joyce}, {lina, martha}, {raju}, {hannah}, {larry}, {jose}, {jack}*
Non-TI	*{roger, justin, eric}, {roger, joan, polash}, {roger, joan, ronald}, {roger, joan, kayle}, {roger, rajat, judith}, {roger, joyce, henry}, {roger, judith, polash}, {roger, judith, henry}, {roger, eric, kayle}, {roger, eric, henry}, {keith, justin, rezart}, {keith, scott, martha}, {keith, scott, walter}, {keith, scott, joan}, {keith, jose, walter}, {keith, jose, ethan}, {keith, sharon, martha}, {keith, sharon, joan}, {keith, sharon, diana, rezart}, {keith, sharon, samuel}, {keith, sharon, ethan}, {keith, amy, samuel}, {keith, amy, diana}, {keith, walter, diana}, {keith, ronald, martha}, {keith, ronald, joan}, {eliza, lina}, {eliza, diana, tyler, amy}, {eliza, peter, amy}, {eliza, peter, hannah}, {eliza, eric, rono, yousef}, {eliza, eric, samuel}, {eliza, eric, tyler}, {eliza, amy, samuel}, {eliza, hannah, yousef}, {eliza, alan}, {kayle, larry, eric}, {kayle, larry, max, diana}, {kayle, jason, joan}, {kayle, jason, megan}, {kayle, jason, diana}, {kayle, jason, kelly}, {kayle, jason, eric, ayuba}, {kayle, andrea, joan}, {kayle, max, megan}, {kayle, chris, kelly}, {kayle, chris, alan}, {kayle, chris, eric}, {kayle, nathan}, {siam, martha, rono}, {siam, martha, scott}, {siam, martha, lina, sharon}, {siam, justin, nathan}, {siam, joan, rono}, {siam, joan, scott}, {siam, joan, sharon}, {siam, joan, harold}, {siam, rajat, amy}, {siam, rajat, sharon}, {siam, rajat, nathan}, {siam, jose, rono}, {siam, jose, harold}, {larry, scott, max}, {larry, ayon, samuel}, {larry, joyce, rono}, {larry, joyce, max, diana}, {larry, eric, rono}, {larry, eric, samuel}, {ayon, ruth, walter}, {ayon, ruth, rajat}, {ayon, jason, judith}, {ayon, jason, joan}, {ayon, jason, ayuba, samuel}, {ayon, jason, ayuba, ethan}, {ayon, jason, rezart}, {ayon, joe, judith}, {ayon, joe, ethan}, {ayon, rajat, judith}, {ayon, rajat, samuel, ayuba}, {ayon, yousef, rezart}, {ayon, yousef, ethan}, {ayon, yousef, walter}, {ayon, lina, raju}, {ayon, lina, ethan}, {ayon, raju, judith, walter}, {yousef, justin, rezart, eric}, {yousef, jose, rono}, {yousef, jose, walter}, {yousef, jose, ethan, andrea}, {yousef, deba, rono}, {yousef, deba, walter}, {yousef, deba, kelly}, {yousef, deba, rezart}, {yousef, joyce, rono}, {yousef, joyce, hannah}, {yousef, eric, ethan}, {yousef, hannah, ethan}, {jose, megan, walter}, {jose, henry, ethan}, {jose, tyler, walter}, {jose, ayuba, ethan}, {deba, ruth, rono}, {deba, ruth, walter}, {deba, ruth, kelly, rajat}, {deba, samuel, sharon, rajat}, {deba, sharon, rezart}, {laura, scott, martha}, {laura, scott, hannah}, {laura, scott, andrea}, {laura, joe}, {laura, harold}, {laura, sharon, martha}, {laura, sharon, hannah}, {laura, amy}, {joyce, jason, ayuba}, {joyce, jason, diana}, {joyce, amy, ayuba}, {joyce, amy, diana}, {joyce, chris}, {rono, ryan}, {polash, joan, tyler}, {polash, joan, sharon}, {polash, joan, jack}, {polash, ryan, jack}, {walter, megan, scott}, {walter, megan, ryan, raju}, {walter, megan, ruth}, {walter, megan, peter}, {walter, diana, raju}, {walter, diana, ruth, tyler}, {walter, judith, scott}, {walter, alan, raju, ryan}, {saiful, martha, kelly}, {saiful, martha, sharon}, {saiful, martha, ronald}, {saiful, martha, jack}, {saiful, ruth, kelly}, {saiful, ruth, megan, max}, {saiful, ruth, megan, ronald}, {saiful, ruth, amy}, {saiful, megan, raju}, {saiful, megan, peter, max}, {saiful, megan, peter, ronald}, {saiful, megan, jason}, {saiful, amy, peter}, {saiful, kelly, peter}, {saiful, kelly, jason}, {saiful, kelly, joe}, {saiful, rezart, jason}, {saiful, rezart, sharon}, {saiful, ethan, jason, jack}, {saiful, ethan, joe, nathan}, {saiful, ethan, sharon, henry}, {saiful, ethan, andrea}, {saiful, ethan, peter}, {saiful, ronald, joe}, {justin, megan, ruth}, {justin, megan, peter}, {scott, joan, andrea}, {scott, megan, max}, {scott, judith, kelly}, {scott, kelly, martha}, {joan, jason, tyler}, {joan, jason, harold, jack}, {harold, alan, jack}, {harold, jason, kelly}, {harold, jason, eric}, {harold, max}, {samuel, jason, ryan}, {samuel, jason, eric, ayuba}, {samuel, rajat, amy, ayuba}, {samuel, rajat, nathan}, {samuel, andrea}, {samuel, eric, chris}, {samuel, chris, sharon}, {chris, judith, kelly}, {chris, eric, ethan}, {chris, lina, kelly}, {chris, lina, sharon, ethan}, {hannah, raju}, {hannah, ethan, peter}, {hannah, ethan, sharon}, {ayuba, ethan, eric, jason}, {ayuba, ronald}, {rajat, peter, kelly}, {rajat, peter, amy}, {rajat, amy, ruth}, {rajat, judith, kelly}, {rajat, jack}, {lina, martha, kelly}, {lina, megan, raju, ryan}, {lina, megan, max}, {lina, andrea, ethan}, {diana, ruth, tyler, amy}, {diana, ruth, max}, {diana, jason, tyler}, {diana, jason, rezart}, {judith, kelly, jason}, {judith, kelly, joe}, {ryan, jason, megan}, {ryan, jason, jack}, {ryan, jack, alan}, {eric, jason, tyler}, {eric, jason, rezart}, {eric, henry, ethan}, {alan, martha, jack}*

User interaction

Complimented form of a weight

Feedback for constructing a new cluster

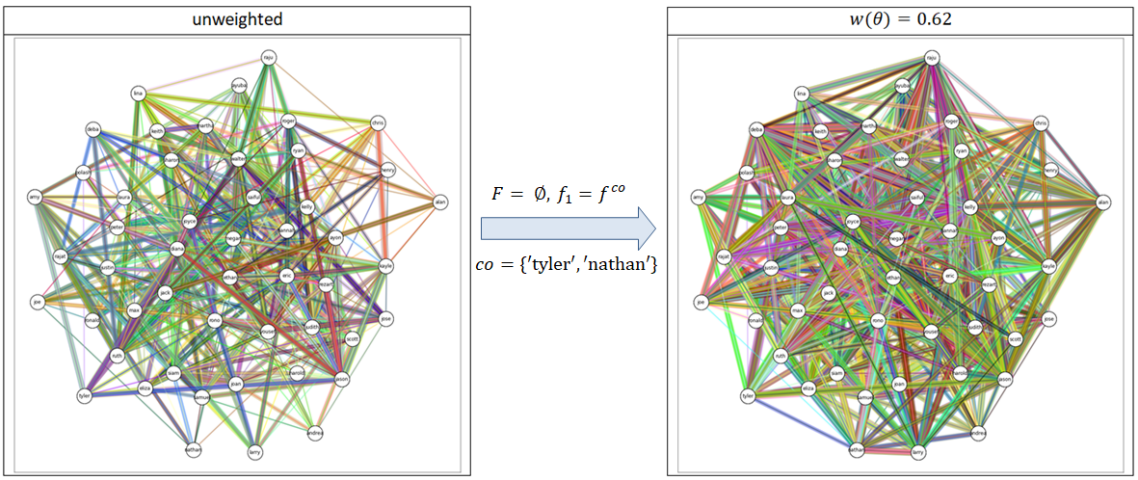


Fig. C.15 Clustering based on f^{co} on a fifty-people network in *conjunction*.

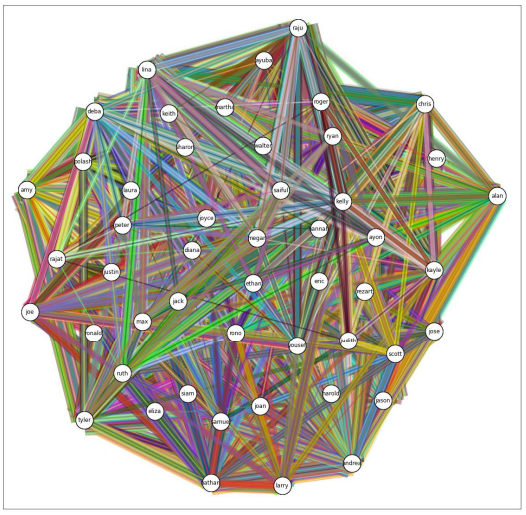


Fig. C.16 Ideal Solution.

Table C.5 Ground-truth clustering

{sharon, samuel, chris, lina}, {sharon, samuel, rajat, jason, joan}, {sharon, samuel, rajat, jason, ayuba}, {sharon, samuel, rajat, joan, polash}, {sharon, samuel, rajat, lina, laura}, {sharon, samuel, rajat, deba, laura}, {sharon, samuel, rajat, deba, polash}, {sharon, samuel, rajat, deba, ayuba}, {sharon, samuel, keith, joan}, {sharon, samuel, keith, justin}, {sharon, samuel, ryan, jason}, {sharon, samuel, ryan, polash}, {sharon, samuel, ryan, lina}, {sharon, samuel, justin, laura}, {sharon, samuel, justin, ayuba}, {sharon, martha, diana, keith, justin}, {sharon, martha, rajat, polash, jack}, {sharon, martha, rajat, polash, deba}, {sharon, martha, rajat, henry, ayuba}, {sharon, martha, rajat, deba, laura}, {sharon, martha, rajat, deba, ayuba}, {sharon, martha, rajat, lina, jack}, {sharon, martha, rajat, lina, laura}, {sharon, martha, rajat, lina, siam}, {sharon, martha, harold, justin, jack}, {sharon, martha, harold, justin, laura}, {sharon, martha, harold, justin, siam}, {sharon, martha, harold, polash, jack}, {sharon, martha, keith, saiful, ethan}, {sharon, martha, ethan, siam, lina}, {sharon, martha, ethan, saiful, jack, lina}, {sharon, martha, ethan, saiful, henry, ayuba}, {sharon, martha, justin, ayuba}, {sharon, martha, saiful, polash, jack}, {sharon, jason, diana, rezart}, {sharon, jason, henry, rajat, ayuba}, {sharon, jason, henry, ryan, saiful}, {sharon, jason, henry, ethan, hannah}, {sharon, jason, henry, ethan, saiful, ayuba}, {sharon, jason, jack, rajat, joan}, {sharon, jason, jack, harold, joan}, {sharon, jason, jack, ethan, hannah}, {sharon, jason, jack, ethan, saiful}, {sharon, jason, jack, ryan, saiful}, {sharon, jason, siam, joan, rajat}, {sharon, jason, siam, joan, harold}, {sharon, jason, siam, ethan}, {sharon, jason, rezart, saiful}, {sharon, chris, siam, lina, ethan}, {sharon, rezart, keith, justin, diana}, {sharon, rezart, keith, saiful}, {sharon, rezart, polash, deba}, {sharon, rezart, polash, saiful}, {sharon, ryan, saiful, polash, jack}, {sharon, ryan, saiful, lina, jack}, {sharon, joan, polash, jack, rajat}, {sharon, joan, polash, jack, harold}, {sharon, hannah, laura, deba}, {walter, joan, samuel, scott, keith}, {walter, joan, samuel, rajat, ayon}, {walter, joan, samuel, rajat, peter}, {walter, joan, yousef, ayon}, {walter, joan, tyler, kayle}, {walter, joan, tyler, peter}, {walter, joan, jack, rajat, ayon}, {walter, joan, jack, rajat, roger}, {walter, joan, jack, kayle, roger}, {walter, joan, roger, scott}, {walter, joan, roger, rajat, peter}, {walter, ruth, max, hannah}, {walter, ruth, max, lina, megan}, {walter, ruth, max, tyler, megan}, {walter, ruth, max, tyler, diana}, {walter, ruth, diana, jose, tyler}, {walter, ruth, rajat, jack, ayon, lina}, {walter, ruth, rajat, samuel, ayon, lina},

{walter, ruth, rajat, samuel, deba}, {walter, ruth, rajat, jose}, {walter, ruth, tyler, megan, jose}, {walter, ruth, megan, deba}, {walter, ruth, hannah, jack}, {walter, ruth, hannah, jose}, {walter, ruth, hannah, deba}, {walter, alan, joe, scott}, {walter, alan, joe, ethan}, {walter, alan, ethan, jack}, {walter, alan, ethan, chris}, {walter, alan, ethan, ayuba}, {walter, alan, kayle, jack, roger}, {walter, alan, kayle, chris}, {walter, alan, kayle, raju}, {walter, alan, kayle, ayuba}, {walter, alan, ryan, raju}, {walter, alan, ryan, roger, jack}, {walter, alan, ryan, roger, scott}, {walter, alan, scott, chris}, {walter, rajat, samuel, ayon, judith}, {walter, rajat, samuel, ayon, laura, lina}, {walter, rajat, samuel, ayon, ayuba}, {walter, rajat, samuel, peter, laura, lina}, {walter, rajat, samuel, peter, laura, deba}, {walter, rajat, samuel, deba, ayuba}, {walter, rajat, roger, judith}, {walter, rajat, roger, deba, peter}, {walter, rajat, jose, ayuba}, {walter, raju, diana, keith}, {walter, raju, diana, kayle}, {walter, raju, diana, yousef}, {walter, raju, judith, ayon}, {walter, raju, judith, kayle}, {walter, raju, judith, hannah}, {walter, raju, ayon, laura, lina}, {walter, raju, ayon, yousef}, {walter, raju, megan, keith}, {walter, raju, megan, lina, laura}, {walter, raju, megan, lina, ryan}, {walter, raju, megan, deba, laura}, {walter, raju, megan, deba, kayle}, {walter, raju, megan, deba, yousef}, {walter, raju, hannah, deba, laura}, {walter, raju, hannah, deba, yousef}, {walter, lina, samuel, scott, chris}, {walter, lina, samuel, scott, laura}, {walter, lina, samuel, scott, ryan}, {walter, lina, max, megan, scott, chris}, {walter, lina, max, megan, scott, laura}, {walter, lina, max, megan, scott, ryan}, {walter, lina, max, megan, peter, laura}, {walter, lina, jack, ethan, ayon}, {walter, lina, jack, ryan}, {walter, lina, ethan, chris}, {walter, lina, ethan, peter}, {walter, jack, hannah, ethan}, {walter, keith, scott, megan}, {walter, keith, jose, megan}, {walter, keith, jose, diana}, {walter, keith, jose, ethan}, {walter, ayon, ayuba, ethan}, {walter, ayon, joe, laura, larry}, {walter, ayon, joe, ethan, judith}, {walter, ayon, joe, ethan, yousef, larry}, {walter, ayon, samuel, larry, laura}, {walter, kayle, max, chris, megan}, {walter, kayle, max, tyler, megan}, {walter, kayle, max, tyler, diana, larry}, {walter, kayle, judith, chris}, {walter, kayle, judith, roger}, {walter, kayle, deba, roger}, {walter, kayle, deba, ayuba}, {walter, kayle, ayuba, tyler}, {walter, joyce, samuel, ryan}, {walter, joyce, samuel, judith, chris}, {walter, joyce, samuel, deba, ayuba}, {walter, joyce, samuel, larry}, {walter, joyce, max, chris, megan}, {walter, joyce, max, hannah}, {walter, joyce, max, tyler, megan}, {walter, joyce, max, tyler, diana, larry}, {walter, joyce, max, ryan, megan}, {walter, joyce, judith, roger}, {walter, joyce, judith, ethan, chris}, {walter, joyce, judith, ethan, joe}, {walter, joyce, judith, ethan, hannah}, {walter, joyce, yousef, megan, chris}, {walter, joyce, yousef, megan, jose}, {walter, joyce, yousef, megan, deba}, {walter, joyce, yousef, diana, jose}, {walter, joyce, yousef, diana, larry},

{walter, joyce, yousef, ethan, chris}, {walter, joyce, yousef, ethan, joe, larry}, {walter, joyce, yousef, ethan, hannah, jose}, {walter, joyce, yousef, deba, joe}, {walter, joyce, yousef, deba, hannah}, {walter, joyce, tyler, jose, megan}, {walter, joyce, tyler, jose, diana}, {walter, joyce, tyler, jose, ayuba}, {walter, joyce, ryan, roger}, {walter, joyce, roger, deba}, {walter, joyce, ayuba, jose, ethan}, {walter, scott, joe, judith}, {walter, scott, joe, laura, larry}, {walter, scott, max, hannah, laura}, {walter, scott, max, larry, laura}, {walter, scott, samuel, judith, chris}, {walter, scott, samuel, larry, laura}, {walter, scott, judith, roger}, {walter, scott, judith, hannah}, {walter, laura, peter, hannah, max}, {walter, laura, peter, hannah, deba}, {walter, laura, peter, deba, megan}, {walter, laura, deba, joe}, {walter, peter, diana, max, tyler}, {walter, peter, ethan, hannah}, {walter, peter, tyler, megan, max}, {ronald, joan, jack, roger, harold}, {ronald, joan, amy, keith}, {ronald, joan, amy, roger, peter}, {ronald, joan, amy, roger, harold}, {ronald, alan, roger, jack, harold}, {ronald, alan, martha, jack, harold}, {ronald, alan, martha, joe}, {ronald, alan, martha, ayuba}, {ronald, harold, megan, laura, martha}, {ronald, harold, laura, amy}, {ronald, harold, eric, roger}, {ronald, laura, hannah, peter}, {ronald, laura, joe, martha}, {ronald, laura, peter, megan}, {ronald, laura, peter, amy}, {ronald, hannah, jack, ruth}, {ronald, saiful, martha, jack}, {ronald, saiful, martha, keith, megan}, {ronald, saiful, martha, joe}, {ronald, saiful, martha, ayuba}, {ronald, saiful, ruth, jack}, {ronald, saiful, ruth, amy}, {ronald, saiful, ruth, megan}, {ronald, saiful, rezart, keith, eric}, {ronald, saiful, amy, keith}, {ronald, saiful, amy, peter}, {ronald, saiful, amy, ayuba}, {ronald, saiful, eric, ayuba}, {ronald, saiful, peter, megan}, {henry, martha, scott, megan}, {henry, martha, scott, alan}, {henry, martha, megan, joyce}, {henry, martha, megan, saiful}, {henry, martha, ayuba, alan, ethan}, {henry, martha, ayuba, joyce, ethan}, {henry, jason, megan, ryan, joyce}, {henry, jason, megan, ryan, saiful}, {henry, jason, rajat, ayon, judith}, {henry, jason, rajat, ayon, amy, ayuba}, {henry, jason, rajat, nathan}, {henry, jason, ryan, andrea, saiful}, {henry, jason, ryan, alan}, {henry, jason, ethan, judith, ayon}, {henry, jason, ethan, judith, hannah, joyce}, {henry, jason, ethan, andrea, ayon, saiful}, {henry, jason, ethan, andrea, nathan, hannah}, {henry, jason, ethan, andrea, nathan, saiful}, {henry, jason, ethan, ayuba, alan}, {henry, jason, ethan, ayuba, ayon, saiful, amy}, {henry, jason, ethan, ayuba, ayon, saiful, eric}, {henry, jason, ethan, ayuba, joyce, amy}, {henry, scott, judith, roger}, {henry, scott, judith, hannah}, {henry, scott, hannah, andrea}, {henry, scott, ryan, andrea}, {henry, scott, ryan, alan, roger}, {henry, scott, ryan, megan}, {henry, roger, rajat, judith}, {henry, roger, rajat, amy}, {henry, roger, eric}, {henry, roger, joyce, judith}, {henry, roger, joyce, amy}, {henry, roger, joyce, ryan}, {henry, jose, megan, joyce}, {henry, jose, rajat, nathan}, {henry, jose, rajat, ayuba}, {henry, jose, ethan, hannah, andrea, nathan}, {henry, jose, ethan, hannah, joyce}, {henry, jose, ethan, ayuba, joyce},

{harold, joan, scott, siam}, {harold, joan, scott, roger}, {harold, joan, amy, jason, siam}, {harold, joan, amy, yousef}, {harold, joan, roger, polash, jack}, {harold, judith, kelly, jason, raju}, {harold, judith, kelly, roger, scott}, {harold, judith, polash, roger, joyce}, {harold, judith, joyce, jason}, {harold, alan, jack, jason}, {harold, alan, kelly, jason, raju}, {harold, alan, kelly, scott, roger}, {harold, alan, kelly, scott, martha}, {harold, jack, roger, justin}, {harold, megan, jason, kelly, raju, siam}, {harold, megan, jason, joyce}, {harold, megan, raju, kelly, yousef}, {harold, megan, raju, justin, laura}, {harold, megan, raju, justin, siam}, {harold, megan, raju, justin, yousef}, {harold, megan, jose, siam}, {harold, megan, jose, joyce, polash}, {harold, megan, jose, joyce, yousef}, {harold, megan, martha, justin, scott, laura}, {harold, megan, martha, justin, scott, siam}, {harold, megan, martha, justin, yousef}, {harold, megan, martha, max, scott, kelly, siam}, {harold, megan, martha, max, scott, laura}, {harold, megan, martha, max, joyce}, {harold, megan, martha, polash, joyce}, {harold, megan, martha, yousef, kelly}, {harold, megan, martha, yousef, joyce}, {harold, amy, kelly, siam, jason}, {harold, amy, kelly, siam, max}, {harold, amy, kelly, roger}, {harold, amy, kelly, yousef}, {harold, amy, laura, max}, {harold, amy, joyce, jason}, {harold, amy, joyce, roger}, {harold, amy, joyce, max}, {harold, amy, joyce, yousef}, {harold, eric, jason}, {harold, eric, justin, roger}, {harold, eric, justin, yousef}, {harold, roger, justin, scott}, {eliza, samuel, rono, larry, laura}, {eliza, samuel, rono, larry, eric}, {eliza, samuel, rono, larry, joyce}, {eliza, samuel, rono, joan}, {eliza, samuel, rono, lina, laura}, {eliza, samuel, rono, lina, eric}, {eliza, samuel, judith, scott}, {eliza, samuel, judith, joyce}, {eliza, samuel, scott, joan}, {eliza, samuel, scott, laura, larry}, {eliza, samuel, scott, laura, lina}, {eliza, samuel, amy, peter, joan}, {eliza, samuel, amy, peter, laura}, {eliza, samuel, amy, joyce}, {eliza, samuel, peter, lina, laura}, {eliza, max, laura, larry, scott}, {eliza, max, laura, larry, rono}, {eliza, max, laura, amy, peter}, {eliza, max, laura, hannah, scott}, {eliza, max, laura, hannah, peter}, {eliza, max, laura, lina, peter}, {eliza, max, laura, lina, martha, scott}, {eliza, max, laura, lina, martha, rono}, {eliza, max, diana, tyler, peter, amy}, {eliza, max, diana, tyler, joyce, amy}, {eliza, max, diana, tyler, joyce, martha}, {eliza, max, diana, tyler, joyce, larry}, {eliza, max, joyce, hannah}, {eliza, max, joyce, rono, martha}, {eliza, max, joyce, rono, larry}, {eliza, judith, hannah, scott}, {eliza, judith, hannah, joyce}, {eliza, alan, martha, scott}, {eliza, alan, martha, rono}, {eliza, yousef, joan, amy}, {eliza, yousef, joan, rono}, {eliza, yousef, eric, rono, larry}, {eliza, yousef, joyce, hannah, jose}, {eliza, yousef, joyce, rono, martha}, {eliza, yousef, joyce, rono, jose}, {eliza, yousef, joyce, rono, larry}, {eliza, yousef, joyce, diana, amy}, {eliza, yousef, joyce, diana, martha}, {eliza, yousef, joyce, diana, jose}, {eliza, yousef, joyce, diana, larry}, {eliza, joan, tyler, amy, peter}, {eliza, eric, tyler, larry}, {eliza, jose, tyler, diana, joyce}, {siam, tyler, max, megan, martha}, {siam, tyler, max, amy}, {siam, tyler, max, larry}, {siam, tyler, martha, megan, justin}, {siam, tyler, nathan, jason, joan}, {siam, tyler, nathan, justin, larry}, {siam, tyler, nathan, jose},

{siam, tyler, megan, jason}, {siam, tyler, megan, jose}, {siam, tyler, amy, jason, joan}, {siam, kelly, jason, rajat, amy}, {siam, kelly, jason, ethan, amy}, {siam, kelly, larry, max, scott}, {siam, kelly, larry, max, rono}, {siam, kelly, larry, ethan}, {siam, kelly, lina, chris, megan, scott, max}, {siam, kelly, lina, chris, ethan}, {siam, kelly, lina, raju, megan}, {siam, kelly, lina, martha, rajat}, {siam, kelly, lina, martha, max, scott, megan}, {siam, kelly, lina, martha, max, rono}, {siam, kelly, lina, martha, ethan}, {siam, nathan, rajat, jason, joan}, {siam, nathan, rajat, jose}, {siam, nathan, rono, joan}, {siam, nathan, rono, jose}, {siam, nathan, rono, larry}, {siam, nathan, ethan, jason}, {siam, nathan, ethan, jose}, {siam, nathan, ethan, larry}, {siam, joan, rajat, jason, amy}, {siam, justin, larry, scott}, {amy, ruth, kelly, ayon, rajat}, {amy, ruth, kelly, ayon, saiful}, {amy, ruth, kelly, rajat, deba}, {amy, ruth, kelly, max, diana}, {amy, ruth, kelly, max, saiful}, {amy, ruth, samuel, rajat, ayon}, {amy, ruth, samuel, rajat, deba}, {amy, ruth, tyler, max, diana}, {amy, ruth, tyler, max, saiful}, {amy, yousef, kelly, ayon, saiful, ethan}, {amy, yousef, kelly, diana}, {amy, yousef, kelly, deba}, {amy, yousef, joan, ayon}, {amy, yousef, joyce, deba}, {amy, yousef, joyce, ethan}, {amy, keith, samuel, joan}, {amy, keith, diana}, {amy, keith, ethan, saiful}, {amy, ayon, jason, saiful, kelly, ethan}, {amy, ayon, jason, rajat, kelly}, {amy, ayon, jason, rajat, samuel, joan}, {amy, ayon, jason, rajat, samuel, ayuba}, {amy, ayon, laura, rajat, samuel}, {amy, joyce, jason, diana, tyler}, {amy, joyce, jason, ayuba, samuel}, {amy, joyce, jason, ayuba, tyler}, {amy, joyce, roger, deba}, {amy, joyce, deba, samuel, ayuba}, {amy, peter, max, kelly, diana}, {amy, peter, max, kelly, saiful}, {amy, peter, max, saiful, tyler}, {amy, peter, jason, kelly, rajat}, {amy, peter, jason, kelly, diana}, {amy, peter, jason, kelly, saiful, ethan}, {amy, peter, jason, joan, rajat, samuel}, {amy, peter, jason, joan, tyler}, {amy, peter, jason, tyler, diana}, {amy, peter, jason, tyler, saiful}, {amy, peter, deba, rajat, kelly, roger}, {amy, peter, deba, rajat, laura, samuel}, {amy, peter, roger, rajat, joan}, {amy, ayuba, jason, tyler, saiful}, {amy, ayuba, deba, rajat, samuel}, {eric, samuel, chris, lina}, {eric, samuel, keith, justin}, {eric, samuel, ayon, jason, ayuba}, {eric, samuel, ayon, lina}, {eric, samuel, ayon, larry}, {eric, samuel, justin, larry}, {eric, samuel, justin, ayuba}, {eric, chris, kayle}, {eric, chris, ethan, yousef}, {eric, chris, ethan, lina}, {eric, kayle, roger, justin}, {eric, kayle, rono, larry}, {eric, kayle, tyler, larry, justin}, {eric, kayle, tyler, ayuba, jason}, {eric, kayle, tyler, ayuba, justin}, {eric, larry, yousef, ayon, ethan}, {eric, larry, yousef, justin}, {eric, justin, keith, rezart}, {eric, justin, rezart, yousef}, {eric, saiful, keith, ethan}, {eric, saiful, rono, rezart, yousef}, {eric, saiful, rono, lina}, {eric, saiful, ayon, rezart, jason}, {eric, saiful, ayon, rezart, yousef}, {eric, saiful, ayon, ethan, yousef}, {eric, saiful, ayon, ethan, lina}, {eric, saiful, tyler, jason, ayuba},

{justin, samuel, ruth, nathan}, {justin, samuel, keith, scott}, {justin, samuel, laura, andrea, scott}, {justin, samuel, laura, andrea, nathan}, {justin, samuel, laura, peter}, {justin, samuel, laura, larry, scott}, {justin, samuel, laura, larry, nathan}, {justin, diana, yousef, raju}, {justin, diana, yousef, martha}, {justin, diana, yousef, rezart}, {justin, diana, yousef, larry}, {justin, diana, tyler, ruth}, {justin, diana, tyler, kayle, larry}, {justin, diana, tyler, peter}, {justin, diana, tyler, martha}, {justin, diana, keith, raju}, {justin, diana, raju, kayle}, {justin, andrea, raju, laura}, {justin, andrea, raju, kayle}, {justin, andrea, raju, yousef}, {justin, andrea, kayle, nathan}, {justin, jack, ruth}, {justin, jack, kayle, roger}, {justin, nathan, kayle, larry, tyler}, {justin, nathan, tyler, ruth}, {justin, scott, keith, megan, martha}, {justin, scott, keith, rezart}, {justin, megan, ruth, tyler}, {justin, megan, kayle, raju}, {justin, megan, kayle, tyler}, {justin, megan, raju, keith}, {justin, megan, peter, laura}, {justin, megan, peter, tyler}, {justin, roger, peter}, {justin, ayuba, tyler, martha}, {jason, samuel, andrea, joan, ayon}, {jason, samuel, andrea, joan, nathan}, {jason, samuel, andrea, ryan}, {jason, samuel, rajat, judith, ayon}, {jason, samuel, rajat, nathan, joan}, {jason, samuel, joyce, judith}, {jason, samuel, joyce, ryan}, {jason, diana, kelly, kayle, raju}, {jason, diana, kelly, rezart}, {jason, diana, joyce, rezart}, {jason, diana, tyler, kayle}, {jason, raju, andrea, kayle}, {jason, raju, andrea, hannah}, {jason, raju, andrea, saiful, ayon}, {jason, raju, andrea, saiful, ryan}, {jason, raju, kelly, judith, ayon}, {jason, raju, kelly, judith, kayle}, {jason, raju, kelly, judith, hannah}, {jason, raju, kelly, alan, kayle}, {jason, raju, kelly, megan, kayle}, {jason, raju, kelly, megan, saiful}, {jason, raju, kelly, saiful, ayon}, {jason, raju, ryan, megan, saiful}, {jason, raju, ryan, alan}, {jason, tyler, megan, kayle}, {jason, tyler, megan, peter, saiful}, {jason, tyler, megan, joyce}, {jason, tyler, nathan, joan, kayle}, {jason, tyler, nathan, saiful}, {jason, jack, alan, kayle}, {jason, jack, alan, ethan}, {jason, jack, alan, ryan}, {jason, jack, kayle, joan}, {jason, jack, ayon, joan, rajat}, {jason, jack, ayon, ethan, saiful}, {jason, kelly, rajat, judith, ayon}, {jason, kelly, peter, megan, saiful}, {jason, kelly, peter, ethan, hannah}, {jason, kelly, rezart, ayon, saiful}, {jason, kelly, ethan, judith, ayon}, {jason, kelly, ethan, judith, hannah}, {jason, kelly, ethan, alan}, {jason, kayle, joan, andrea, nathan}, {jason, kayle, ayuba, alan}, {andrea, rono, nathan, jose}, {andrea, rono, nathan, samuel, joan}, {andrea, rono, nathan, samuel, laura}, {andrea, rono, nathan, kayle, joan}, {andrea, rono, nathan, saiful}, {andrea, rono, ryan, lina, samuel}, {andrea, rono, ryan, lina, saiful}, {andrea, rono, yousef, joan}, {andrea, rono, yousef, jose}, {andrea, rono, yousef, saiful}, {andrea, rono, lina, laura, samuel}, {andrea, polash, samuel, joan}, {andrea, polash, samuel, ryan}, {andrea, polash, jose}, {andrea, polash, saiful, ryan}, {andrea, chris, kayle}, {andrea, chris, yousef, ethan}, {andrea, chris, lina, scott, samuel}, {andrea, chris, lina, ethan}, {andrea, yousef, ayon, joan}, {andrea, yousef, ayon, saiful, raju}, {andrea, yousef, ayon, saiful, ethan}, {andrea, yousef, hannah, raju},

{andrea, yousef, hannah, jose, ethan}, {andrea, lina, scott, samuel, laura}, {andrea, lina, scott, samuel, ryan}, {andrea, lina, ayon, raju, laura}, {andrea, lina, ayon, raju, saiful}, {andrea, lina, ayon, samuel, laura}, {andrea, lina, ayon, ethan, saiful}, {andrea, lina, ryan, raju, saiful}, {andrea, scott, samuel, joan}, {andrea, scott, hannah, laura}, {andrea, laura, nathan, hannah}, {andrea, laura, hannah, raju}, {nathan, joe, deba, laura}, {nathan, joe, laura, larry}, {nathan, joe, ethan, larry}, {nathan, joe, ethan, saiful}, {nathan, ruth, hannah, jose}, {nathan, ruth, hannah, deba}, {nathan, ruth, rajat, samuel, deba}, {nathan, ruth, rajat, jose}, {nathan, ruth, rono, samuel, deba}, {nathan, ruth, rono, jose}, {nathan, ruth, rono, saiful}, {nathan, ruth, tyler, jose}, {nathan, ruth, tyler, saiful}, {nathan, rajat, samuel, laura, deba}, {nathan, deba, hannah, laura}, {nathan, deba, rono, laura, samuel}, {nathan, deba, rono, kayle}, {nathan, larry, rono, laura, samuel}, {nathan, larry, rono, kayle}, {rezart, scott, kelly}, {rezart, polash, joyce, deba}, {rezart, yousef, kelly, ayon, saiful}, {rezart, yousef, kelly, rono, deba}, {rezart, yousef, kelly, rono, saiful}, {rezart, yousef, kelly, diana}, {rezart, yousef, joyce, rono, deba}, {rezart, yousef, joyce, diana}, {polash, rajat, judith, samuel}, {polash, rajat, judith, roger}, {polash, rajat, peter, joan, samuel}, {polash, rajat, peter, joan, roger}, {polash, rajat, peter, deba, samuel}, {polash, rajat, peter, deba, roger}, {polash, rajat, roger, jack, joan}, {polash, rajat, jose}, {polash, jack, roger, ryan}, {polash, joyce, samuel, judith}, {polash, joyce, samuel, deba}, {polash, joyce, samuel, ryan}, {polash, joyce, ryan, megan}, {polash, joyce, ryan, roger}, {polash, joyce, tyler, megan, jose}, {polash, joyce, tyler, megan, martha}, {polash, joyce, deba, roger}, {polash, joyce, deba, martha, megan}, {polash, joan, tyler, peter}, {polash, peter, megan, saiful, tyler}, {polash, peter, megan, deba}, {polash, saiful, megan, ryan}, {polash, saiful, megan, martha, tyler}, {martha, tyler, max, megan, joyce}, {martha, tyler, max, megan, saiful}, {martha, tyler, ayuba, joyce}, {martha, tyler, ayuba, saiful}, {martha, jack, alan, ethan}, {martha, kelly, max, diana}, {martha, kelly, max, lina, saiful, megan}, {martha, kelly, max, lina, saiful, rono}, {martha, kelly, alan, joe, scott}, {martha, kelly, alan, joe, ethan}, {martha, kelly, alan, rono}, {martha, kelly, rajat, deba}, {martha, kelly, yousef, saiful, megan}, {martha, kelly, yousef, saiful, joe, ethan}, {martha, kelly, yousef, saiful, rono}, {martha, kelly, yousef, diana}, {martha, kelly, yousef, deba, megan}, {martha, kelly, yousef, deba, joe}, {martha, kelly, yousef, deba, rono}, {martha, kelly, lina, ethan, saiful}, {martha, joyce, yousef, ethan, joe}, {martha, joyce, yousef, deba, megan}, {martha, joyce, yousef, deba, joe}, {martha, joyce, yousef, deba, rono}, {martha, joyce, ayuba, deba}, {martha, laura, joe, scott}, {martha, laura, joe, deba}, {martha, laura, lina, max, megan, scott}, {martha, laura, deba, megan}, {martha, laura, deba, rono}, {rono, ruth, deba, kelly}, {rono, ruth, lina, kelly, max, saiful}, {rono, ruth, lina, samuel}, {rono, deba, samuel, joyce},

{rono, deba, kayle, kelly}, {rono, kelly, kayle, alan}, {rono, kelly, kayle, larry, max}, {rono, kelly, larry, yousef}, {rono, ryan, alan}, {rono, ryan, samuel, joyce}, {rono, ryan, max, joyce}, {rono, ryan, max, saiful, lina}, {kelly, joe, judith, scott}, {kelly, joe, judith, ayon, ethan}, {kelly, joe, ayon, yousef, ethan, larry}, {kelly, joe, ayon, yousef, ethan, saiful}, {kelly, joe, larry, scott}, {kelly, max, megan, kayle, chris}, {kelly, max, megan, lina, ruth, saiful}, {kelly, max, megan, lina, peter, saiful}, {kelly, max, hannah, scott}, {kelly, max, hannah, ruth}, {kelly, max, hannah, peter}, {kelly, max, diana, kayle, larry}, {kelly, ruth, ayon, lina, rajat}, {kelly, ruth, ayon, lina, saiful}, {kelly, ruth, deba, megan}, {kelly, ruth, deba, hannah}, {kelly, chris, judith, scott}, {kelly, chris, judith, kayle}, {kelly, chris, judith, ethan}, {kelly, chris, alan, scott}, {kelly, chris, alan, kayle}, {kelly, chris, alan, ethan}, {kelly, chris, yousef, megan}, {kelly, chris, yousef, ethan}, {kelly, yousef, larry, diana}, {kelly, yousef, raju, ayon, saiful}, {kelly, yousef, raju, saiful, megan}, {kelly, yousef, raju, diana}, {kelly, yousef, raju, deba, megan}, {kelly, yousef, raju, deba, hannah}, {kelly, yousef, ethan, hannah}, {kelly, lina, raju, saiful, megan}, {kelly, lina, raju, saiful, ayon}, {kelly, lina, ayon, ethan, saiful}, {kelly, lina, peter, rajat}, {kelly, lina, peter, ethan, saiful}, {kelly, deba, raju, kayle, megan}, {kelly, deba, kayle, roger}, {kelly, deba, peter, megan}, {kelly, deba, peter, hannah}, {kelly, scott, hannah, judith}, {kelly, roger, judith, rajat}, {kelly, roger, judith, kayle}, {kelly, roger, alan, kayle}, {saiful, ruth, lina, jack, ayon}, {saiful, ruth, tyler, megan, max}, {saiful, ayon, ethan, jack, lina}, {saiful, ryan, lina, megan, raju}, {saiful, ryan, lina, megan, max}, {saiful, raju, keith, megan}, {saiful, peter, tyler, max, megan}

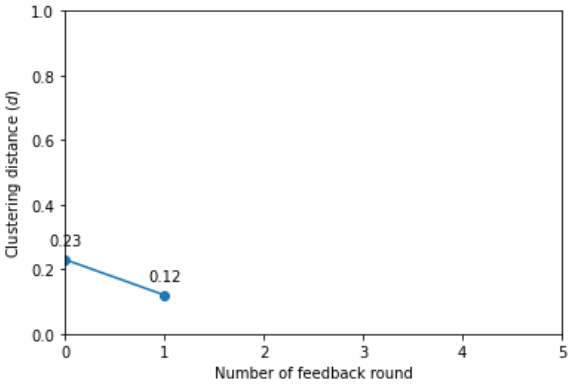


Fig. C.17 Clustering distances based on f^{co} in Figure C.15.

Feedback for removing a person from a cluster

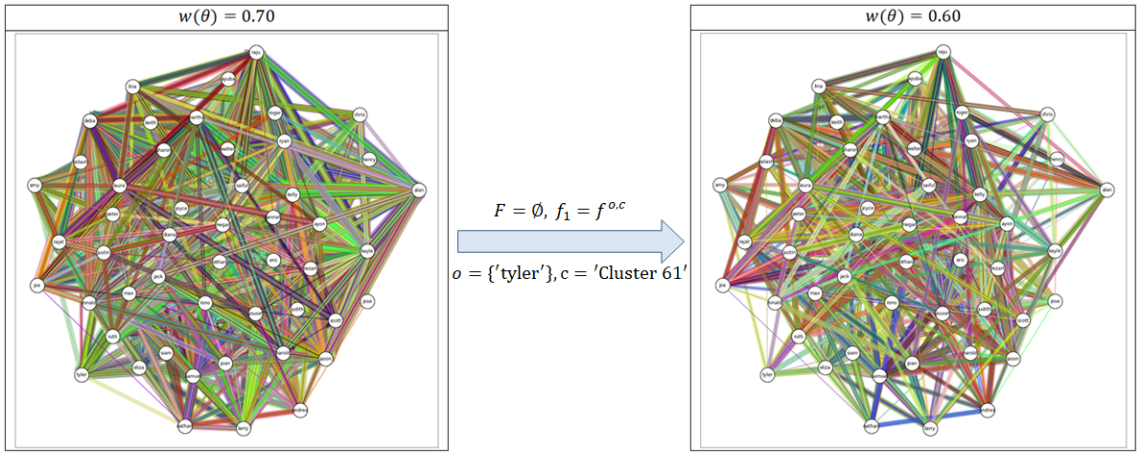


Fig. C.18 Clustering based on $f^{o,c}$ on a fifty-people network in *conjunction*.

Weights based on influence

Feedback for constructing a new cluster

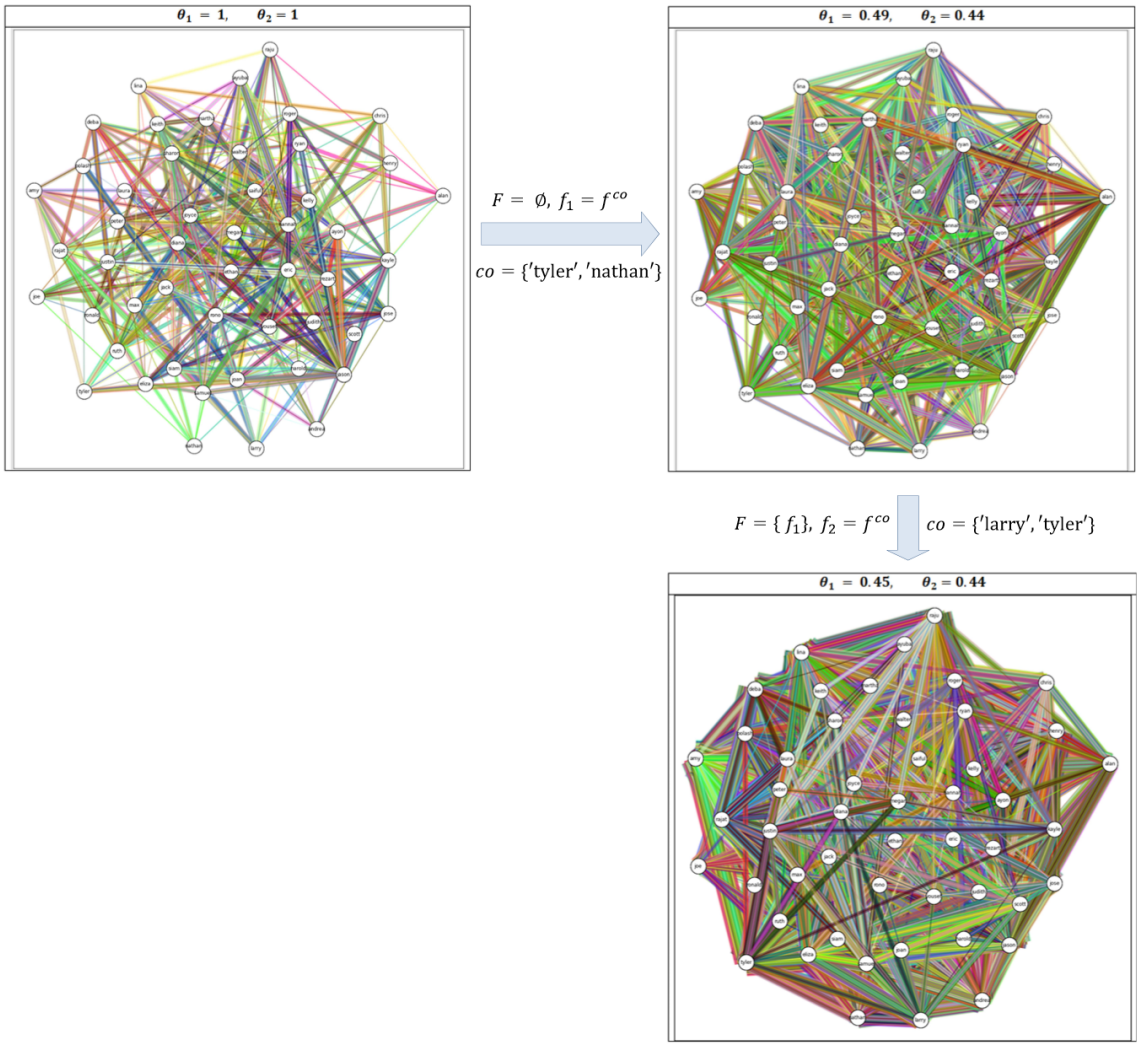


Fig. C.19 Clustering based on f^{co} on a fifty-people network in *conjunction*.

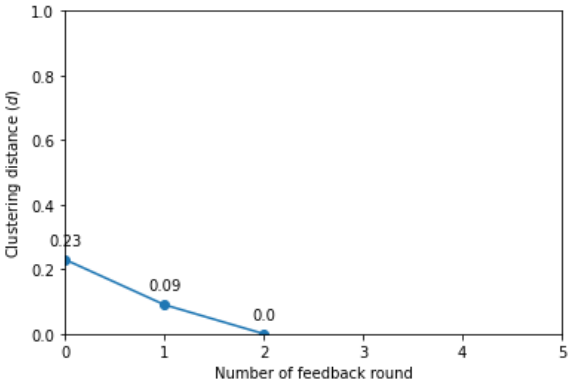


Fig. C.20 Clustering distances based on f^{co} in Figure C.19.

Feedback for removing a person from a cluster

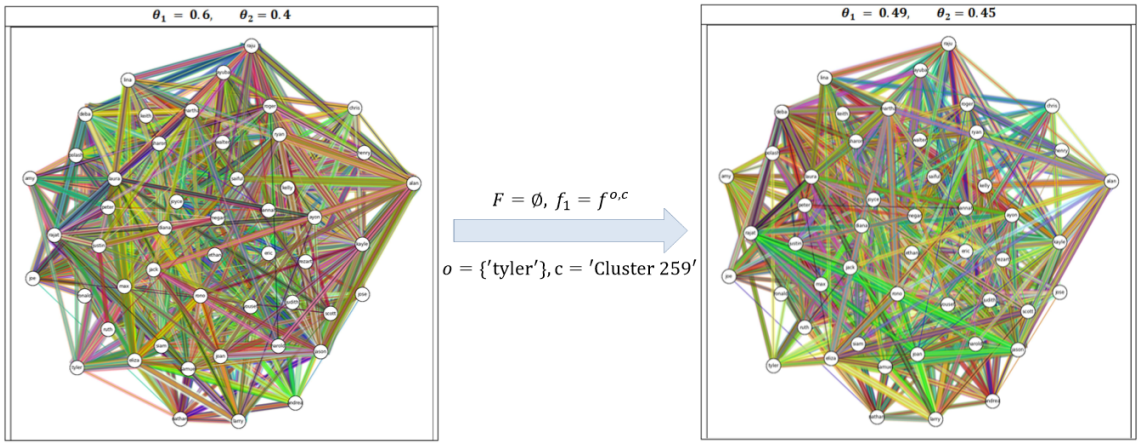
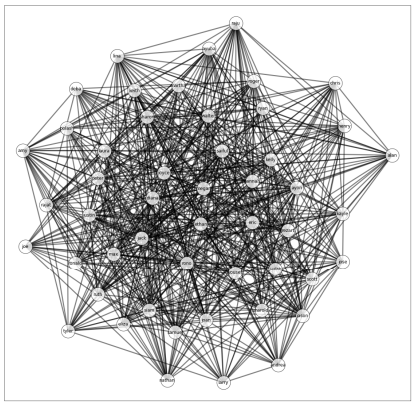


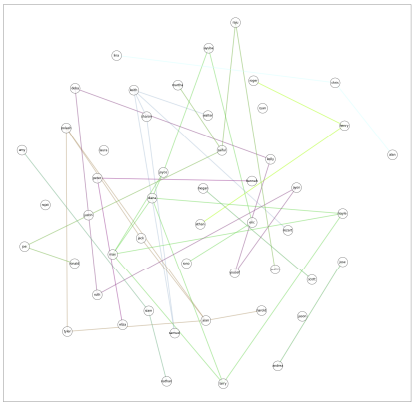
Fig. C.21 Clustering based on $f^{o,c}$ on a fifty-people network in *conjunction*.

C.2.2 CQQL query with disjunction (∨)

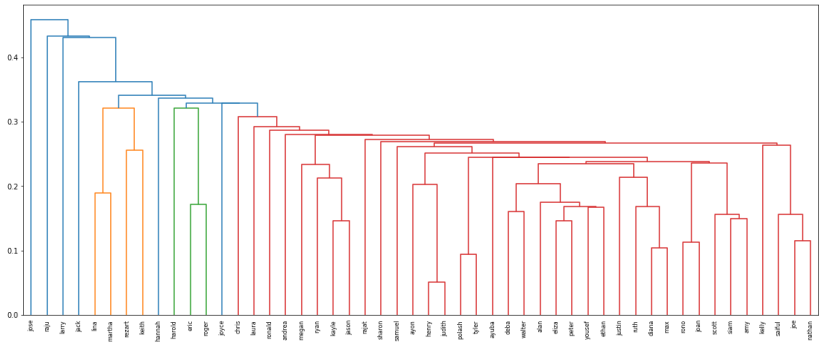
Clustering approaches



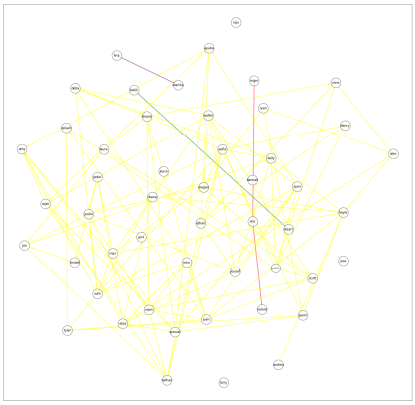
(a) Unweighted



(b) k-medoids ($k = 10$) clustering



(c) Dendogram of Figure C.22d



(d) Hierarchical (single-linkage $k = 10$) clustering

Fig. C.22 Clustering on a fifty-people network in *disjunction*.

Table C.6 Clusters after clustering on a fifty-people network in *disjunction*

Methods	**Clusters**
k-medoids (k=10)	*{rezart, justin, sharon, samuel, walter, keith}, {scott, jose, megan, andrea}, {polash, ryan, jack, tyler, joan, harold}, {henry, ethan, roger}, {diana, ayuba, kayle, rono, max, eric, larry, joyce}, {ayon, deba, yousef, rajat, ruth, kelly}, {joe, raju, saiful, ronald, judith, martha}, {alan, lina, chris, jason}, {siam, amy, nathan}, {eliza, peter, laura, hannah}*
Hierarchical (Single-linkage k=10)	*{eric, harold, roger}, {rezart, keith}, {siam, ayon, alan, joe, eliza, diana, ayuba, deba, kayle, yousef, polash, rono, saiful, rajat, max, peter, chris, ryan, jason, ronald, amy, justin, sharon, scott, laura, samuel, ruth, tyler, henry, nathan, kelly, ethan, joan, walter, megan, judith, andrea}, {joyce}, {lina, martha}, {raju}, {hannah}, {larry}, {jose}, {jack}*

User interaction

Complimented form of a weight

The logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. Therefore, the experimental procedure and results will be equal for the same parameter settings as described in C.2.1.

Weights based on influence

Feedback for constructing a new cluster

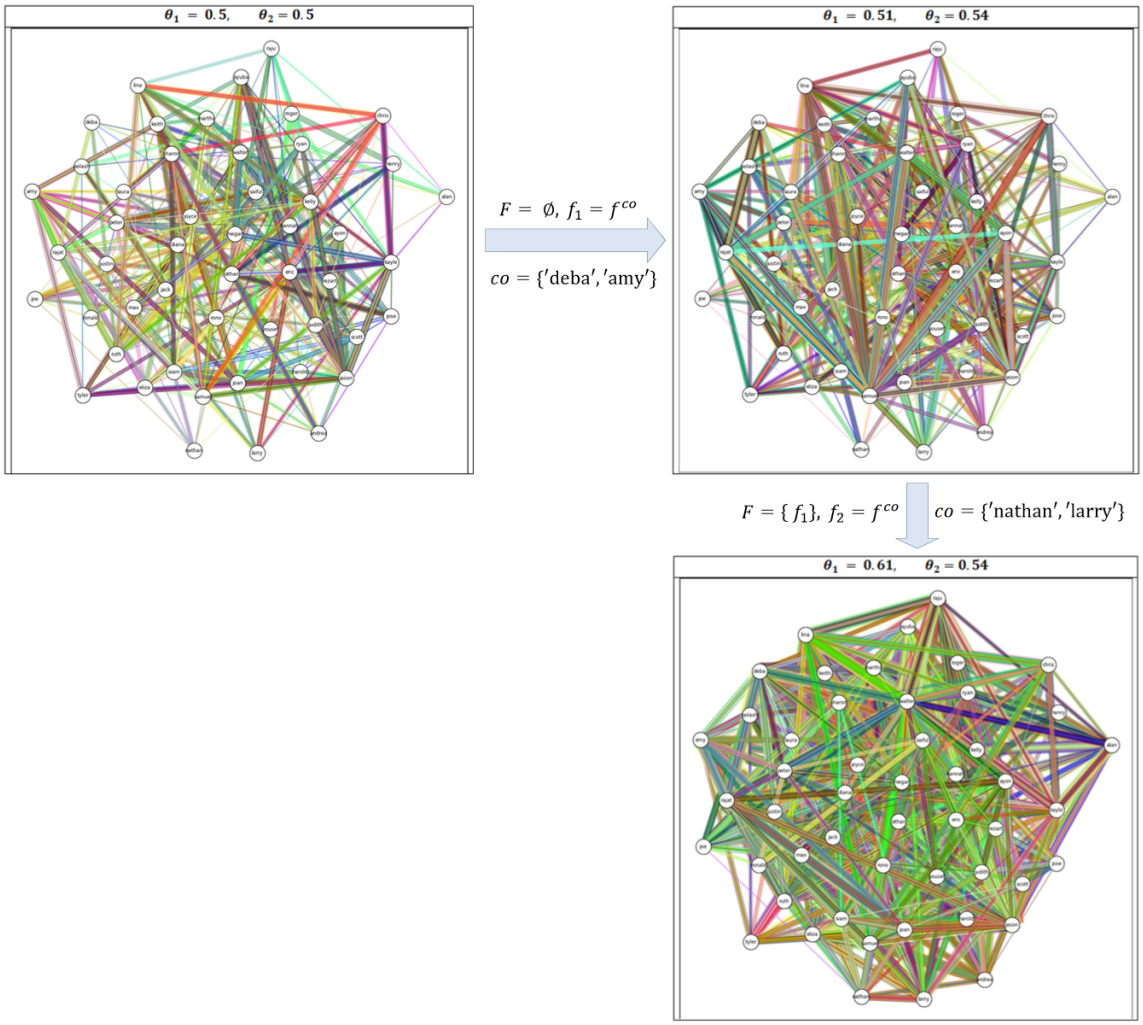


Fig. C.23 Clustering based on f^{co} on a fifty-people network in *disjunction*.

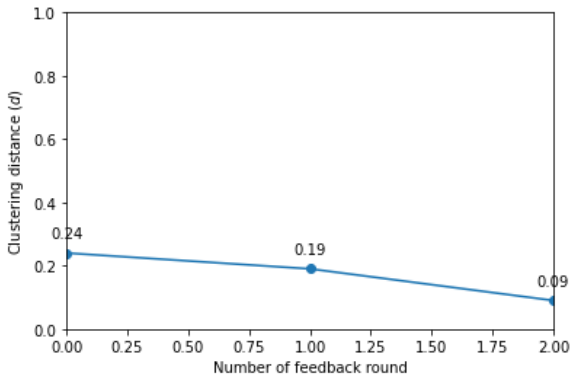


Fig. C.24 Clustering distances based on f^{co} in Figure C.23.

Feedback for removing a person from a cluster

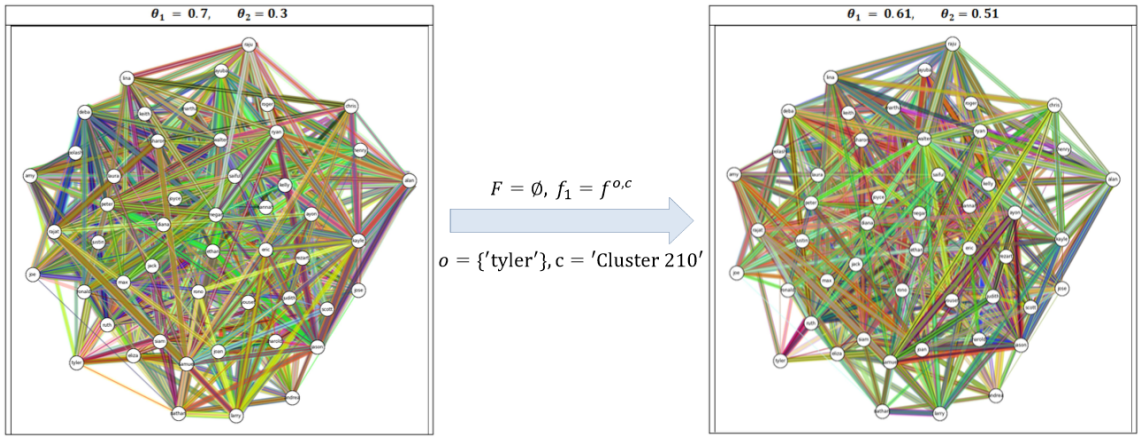
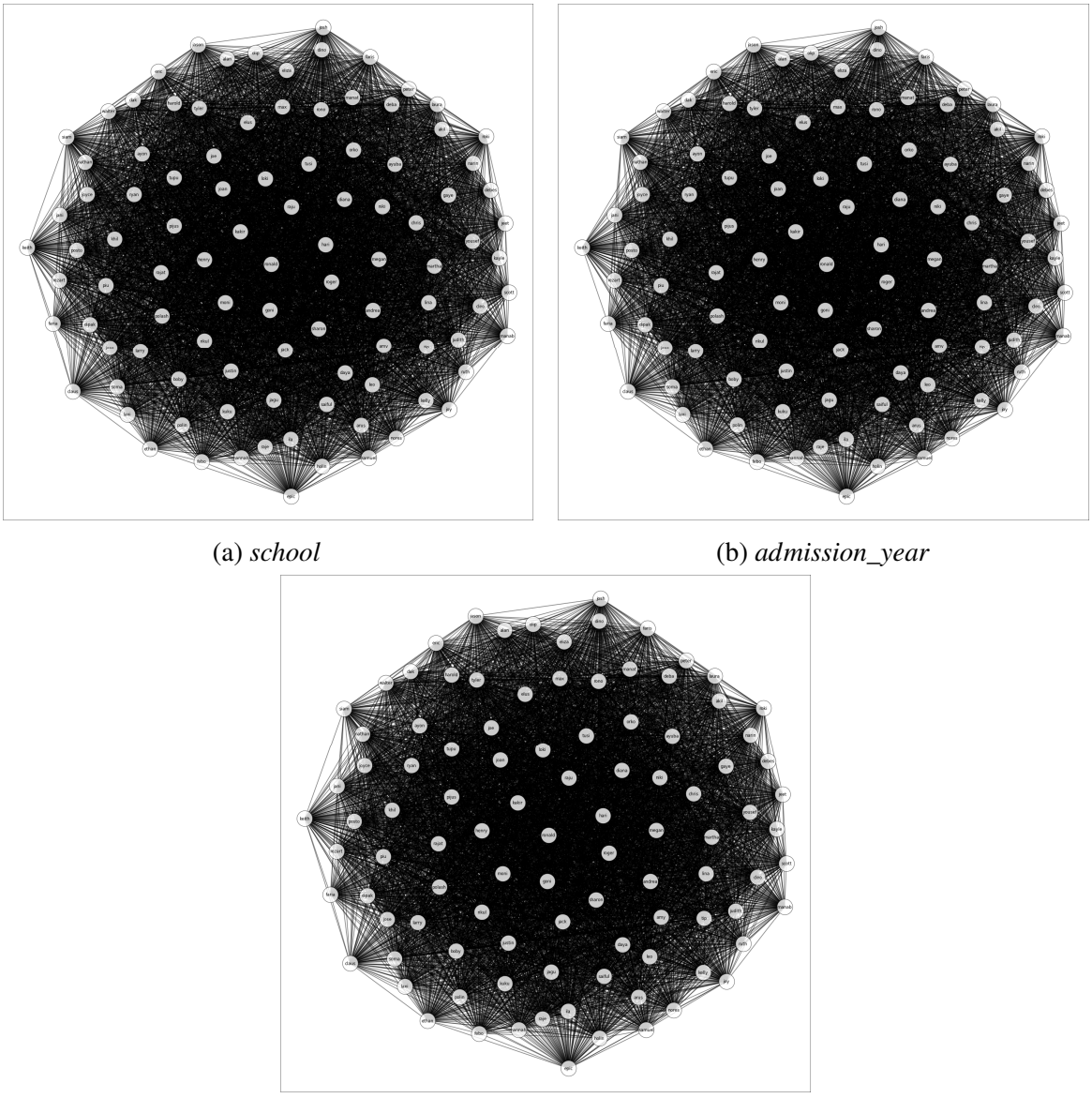


Fig. C.25 Clustering based on $f^{o,c}$ on a fifty-people network in *disjunction*.

C.3 Clustering of a social network (100 people)



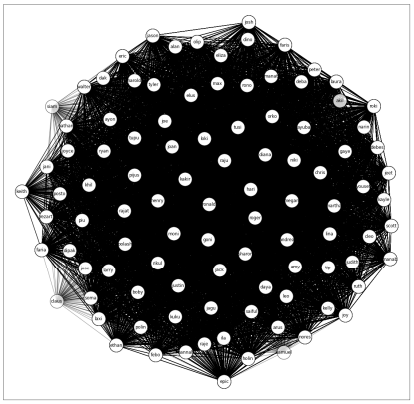
(a) *school* (b) *admission_year*

(c) *graduation_year*

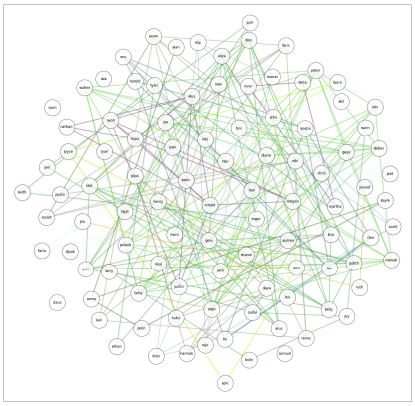
Fig. C.26 Structure of a 100-people network.

C.3.1 CQQL query with conjunction (∧)

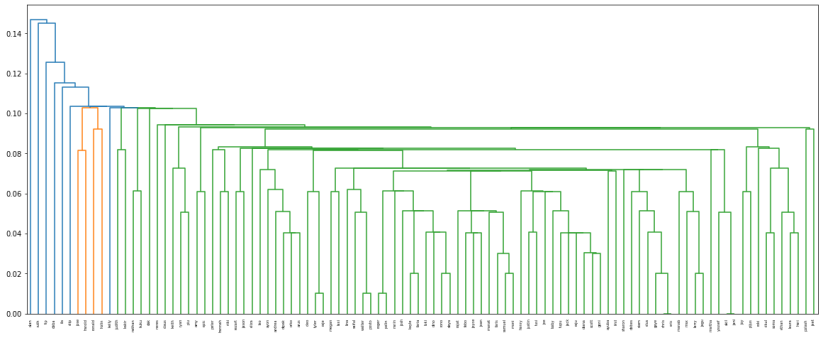
Clustering approaches



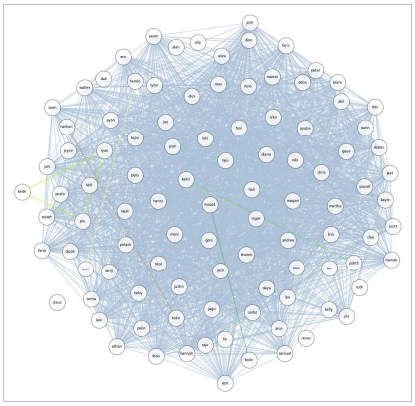
(a) Unweighted



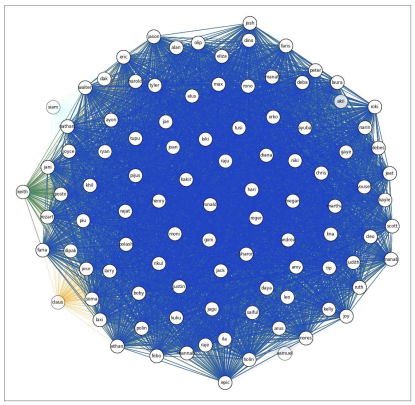
(b) k-medoids ($k = 10$) clustering



(c) Dendogram of Figure C.27d



(d) Hierarchical (single-linkage $k = 10$) clustering



(e) Non-TI

Fig. C.27 Clustering on a 100-people network in *conjunction*.

Table C.7 Clusters after clustering on a 100-people network in *conjunction*

Methods	**Clusters**
k-medoids (k=16)	*{siam, ayon, joe, eliza, diana, lina, ayuba, kayle, raju, rezart, yousef, polash, rono, saiful, rajat, max, peter, chris, jason, eric, larry, amy, justin, sharon, scott, laura, samuel, jack, tyler, joyce, henry, ethan, joan, walter, megan, andrea, roger, hannah, martha, leo, joy, orko, hari, cleo, elus, manat, raje, faris, pijus, posto, debes, faria, narin, khil, febo, jagu, daya, dino, goni, roki, boby, dipak, gaye, rikul, josh, arus, niki, loki, soma, jeet, akil, laxi, tusi, jani, epic, polin, moni, manab, tupu}, {ronald, holin}, {nathan, kuku}, {ryan, keith, piu}, {judith, kakir}, {olip}, {ila}, {tip}, {kelly}, {deba}, {jose, harold}, {ruth}, {dak}, {nores}, {claus}, {alan}*
Hierarchical (Single-linkage k=16)	*{scott, manat, raje, febo, olip, tusi}, {eliza, lina, jose, henry, kelly, judith, khil, dino, goni, arus}, {sharon, andrea, hannah, pijus, daya, kuku}, {amy, laura, joyce, orko, rikul, epic}, {joe, diana, polash, peter, jack, walter, debes, jagu, roki, boby, gaye, manab}, {ayon, ronald, justin, nathan, megan, elus, tupu}, {rajat, josh, jani, moni}, {chris, faria, loki, jeet}, {saiful, larry, joy, narin, holin}, {deba, rono, martha}, {siam, raju, ryan, samuel, joan, leo, claus, dak, dipak, akil}, {rezart, max, jason, faris, nores, posto, kakir, niki, laxi}, {alan, kayle, soma, polin}, {ayuba, eric, harold, tip, cleo, ila}, {tyler, ethan, keith, roger, hari}, {yousef, ruth, piu}*

Non-TI	*{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, Samuel},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, dak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, joan},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, raju},*

{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, leo},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, dipak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, saiful, ryan},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, siam, dipak},

{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, dak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, joan},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, raju},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, leo},

{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, dipak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, keith, ryan},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, akil},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, claus},

{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, dak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, joan},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, raju},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, leo},

	{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, dipak},{ayon, posto, chris, ethan, rezart, elus, walter, kelly, jason, manab, megan, joe, sharon, polash, rajat, josh, henry, khil, piu, roki, tip, gaye, pijus, justin, orko, faria, peter, alan, febo, diana, harold, ruth, manat, andrea, kayle, jani, max, jose, debes, nathan, soma, rikul, goni, joyce, daya, judith, scott, holin, laura, cleo, eric, tusi, arus, raje, epic, tupu, jeet, tyler, lina, boby, loki, ayuba, yousef, faris, martha, moni, hannah, rono, eliza, jagu, nores, narin, kakir, roger, deba, laxi, hari, larry, niki, ila, jack, amy, polin, ronald, joy, kuku, olip, dino, ryan}

User interaction

Complimented form of a weight

Feedback for constructing a new cluster

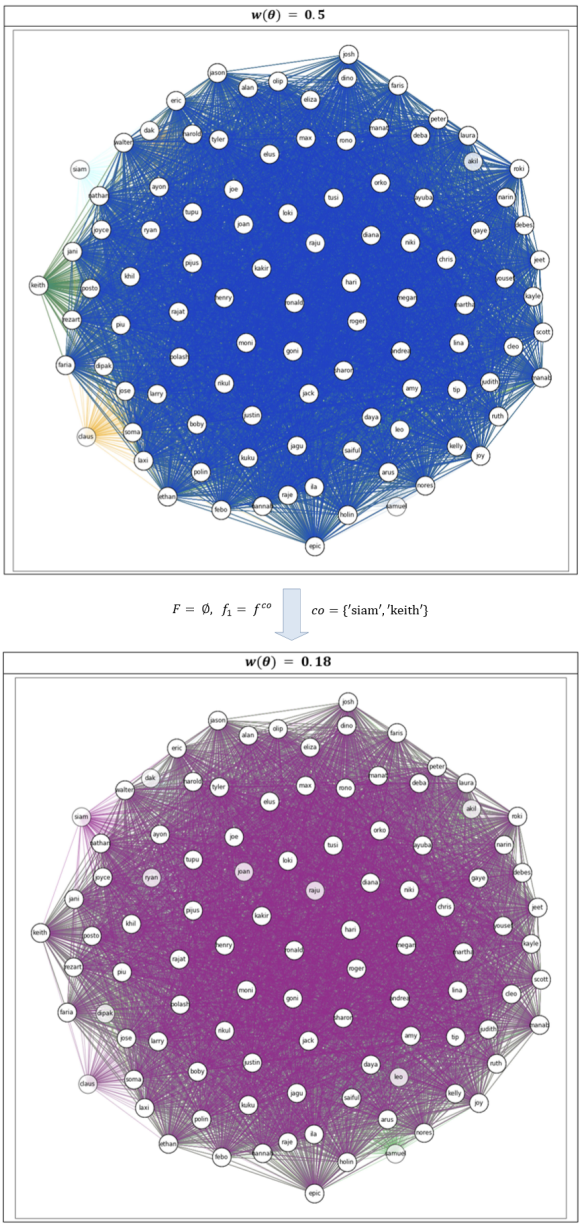


Fig. C.28 Clustering based on f^{co} on a 100-people network in *conjunction*.

Feedback for removing a person from a cluster

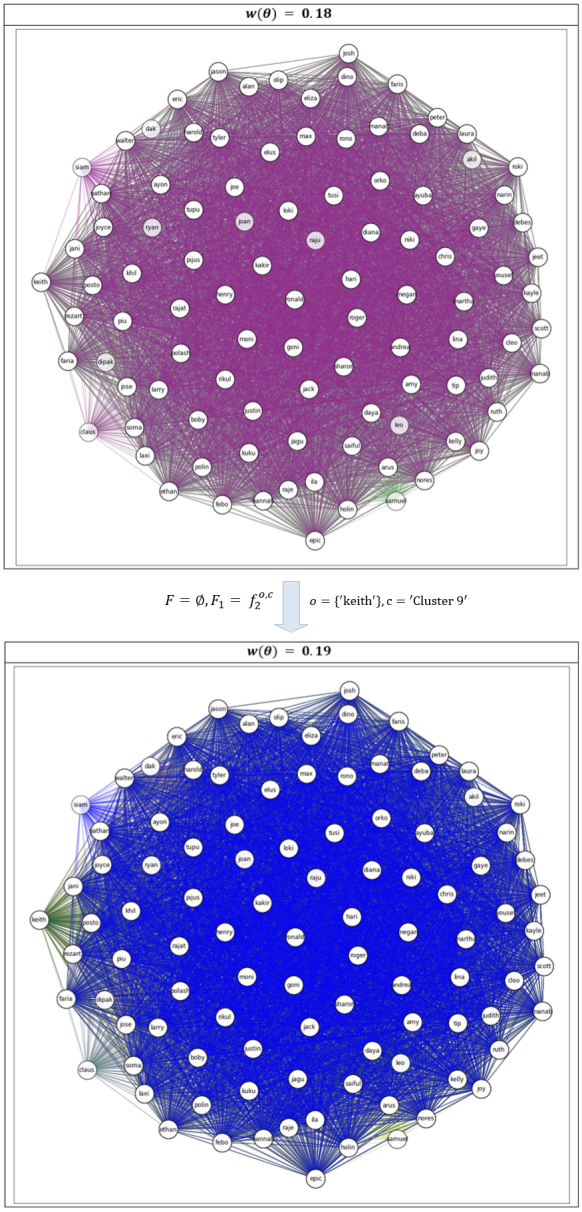


Fig. C.29 Clustering based on $f^{o,c}$ on a 100-people network in *conjunction*.

Weights based on influence

Feedback for constructing a new cluster

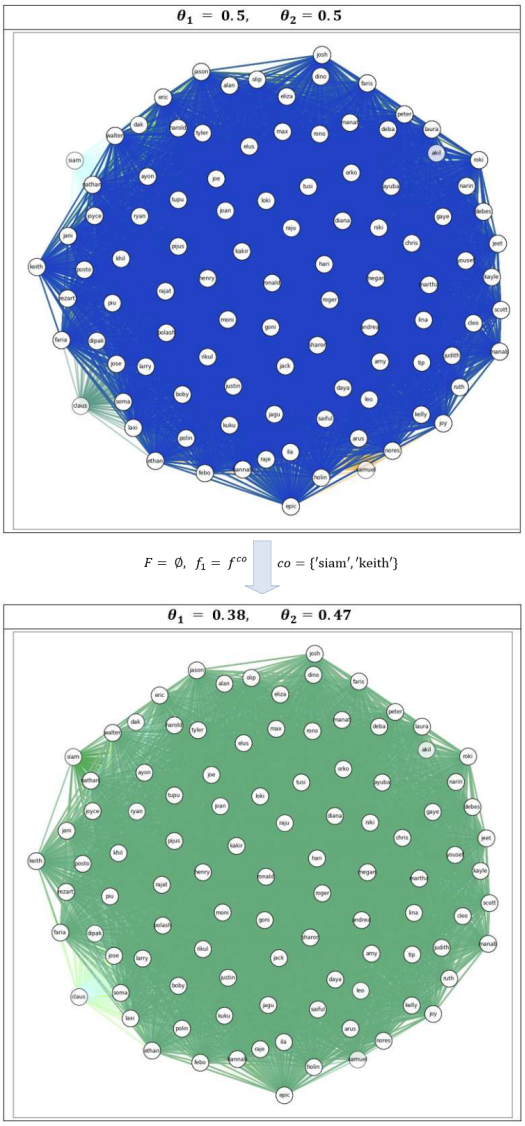


Fig. C.30 Clustering based on f^{co} on a 100-people network in *conjunction*.

Feedback for removing a person from a cluster

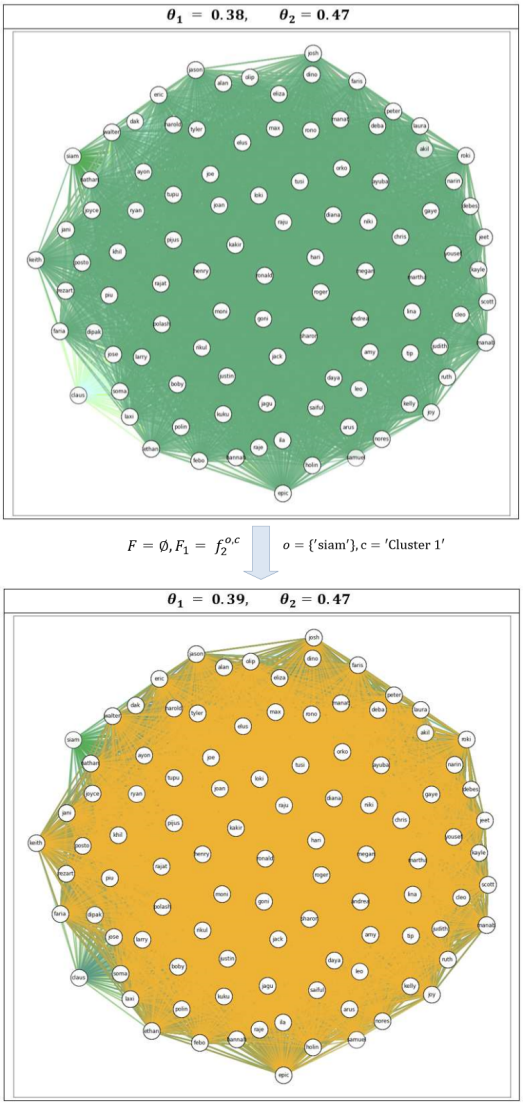
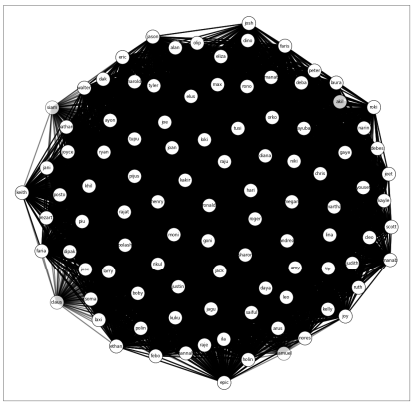


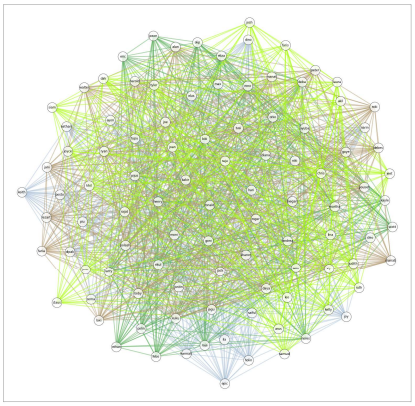
Fig. C.31 Clustering based on $f^{o,c}$ on a 100-people network in *conjunction*.

C.3.2 CQQL query with disjunction (∨)

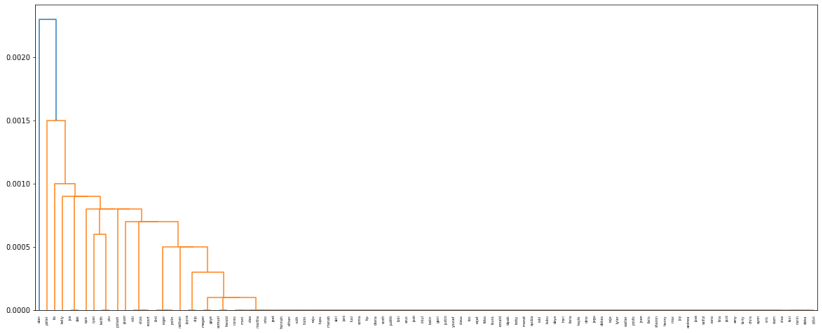
Clustering approaches



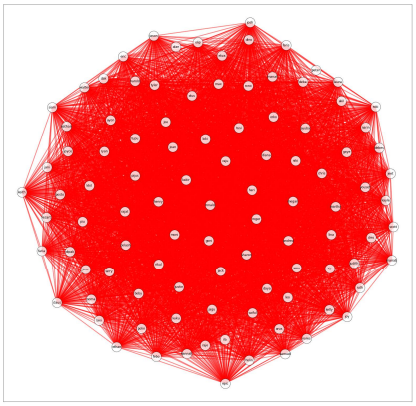
(a) Unweighted



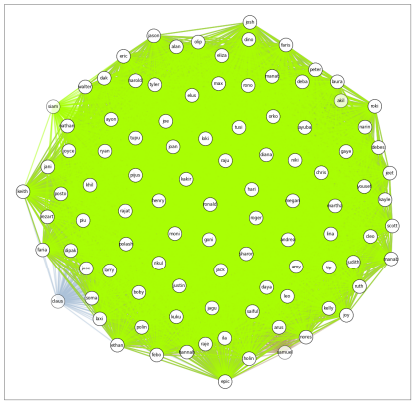
(b) k-medoids ($k = 4$) clustering



(c) Dendogram of Figure C.32d



(d) Hierarchical ($k = 4$) clustering



(e) Non-TI

Fig. C.32 Clustering on a 100-people network in *disjunction*.

Table C.8 Clusters after clustering on a 100-people network in *disjunction*

Methods	**Clusters**
k-medoids (k=4)	*{ayon, diana, saiful, max, ronald, justin, sharon, nathan, megan, keith, andrea, hannah, joy, hari, cleo, elus, piu, posto, ila, narin, dino, dipak, niki, soma, holin, epic, moni, tupu}, {eliza, ayuba, deba, kayle, rono, jason, eric, larry, amy, scott, henry, ethan, martha, orko, raje, pijus, nores, febo, olip, goni, rikul, loki, polin}, {alan, joe, rezart, yousef, polash, rajat, peter, ryan, jack, walter, harold, roger, tip, manat, debes, faria, jagu, daya, kuku, roki, boby, gaye, kakir, laxi, tusi, jani, manab}, {siam, lina, raju, chris, laura, samuel, ruth, tyler, jose, joyce, kelly, joan, judith, leo, claus, faris, khil, dak, josh, arus, jeet, akil}*
Hierarchical (Single-linkage k=4)	*{siam, ayon, joe, eliza, diana, lina, ayuba, deba, kayle, raju, rezart, yousef, polash, rono, saiful, rajat, max, chris, ryan, jason, ronald, eric, larry, amy, justin, sharon, scott, laura, samuel, ruth, jack, tyler, jose, joyce, henry, nathan, kelly, ethan, joan, walter, megan, judith, harold, keith, andrea, roger, hannah, martha, leo, joy, orko, tip, hari, cleo, elus, piu, manat, raje, claus, faris, pijus, nores, posto, debes, faria, narin, khil, dak, febo, jagu, olip, daya, dino, goni, kuku, roki, boby, dipak, gaye, kakir, rikul, josh, arus, niki, loki, soma, jeet, akil, holin, laxi, tusi, jani, epic, polin, moni, manab, tupu}, {ila}, {peter}, {alan}*
Non-TI	*{piu, hari, joyce, nores, yousef, joy, andrea, tupu, kelly, laura, elus, harold, deba, loki, holin, kuku, amy, rikul, jason, megan, goni, dak, justin, ethan, josh, tip, peter, manat, polin, gaye, febo, ruth, dino, debes, martha, jeet, daya, henry, polash, narin, ayon, max, walter, faris, jani, scott, diana, jagu, moni, alan, khil, eric, ayuba, laxi, raju, lina, manab, saiful, joe, roki, chris, hannah, olip, ronald, rezart, keith, jose, boby, rono, sharon, rajat, kayle, niki, kakir, raje, larry, orko, joan, cleo, dipak, soma, judith, nathan, epic, arus, ila, eliza, leo, faria, jack, posto, pijus, roger, tyler, ryan, tusi, claus},*

	{piu, hari, joyce, nores, yousef, joy, andrea, tupu, kelly, laura, elus, harold, deba, loki, holin, kuku, amy, rikul, jason, megan, goni, dak, justin, ethan, josh, tip, peter, manat, polin, gaye, febo, ruth, dino, debes, martha, jeet, daya, henry, polash, narin, ayon, max, walter, faris, jani, scott, diana, jagu, moni, alan, khil, eric, ayuba, laxi, raju, lina, manab, saiful, joe, roki, chris, hannah, olip, ronald, rezart, keith, jose, boby, rono, sharon, rajat, kayle, niki, kakir, raje, larry, orko, joan, cleo, dipak, soma, judith, nathan, epic, arus, ila, eliza, leo, faria, jack, posto, pijus, roger, tyler, ryan, tusi, akil},{piu, hari, joyce, nores, yousef, joy, andrea, tupu, kelly, laura, elus, harold, deba, loki, holin, kuku, amy, rikul, jason, megan, goni, dak, justin, ethan, josh, tip, peter, manat, polin, gaye, febo, ruth, dino, debes, martha, jeet, daya, henry, polash, narin, ayon, max, walter, faris, jani, scott, diana, jagu, moni, alan, khil, eric, ayuba, laxi, raju, lina, manab, saiful, joe, roki, chris, hannah, olip, ronald, rezart, keith, jose, boby, rono, sharon, rajat, kayle, niki, kakir, raje, larry, orko, joan, cleo, dipak, soma, judith, nathan, epic, arus, ila, eliza, leo, faria, jack, posto, pijus, roger, tyler, ryan, tusi, samuel}, {piu, hari, joyce, nores, yousef, joy, andrea, tupu, kelly, laura, elus, harold, deba, loki, holin, kuku, amy, rikul, jason, megan, goni, dak, justin, ethan, josh, tip, peter, manat, polin, gaye, febo, ruth, dino, debes, martha, jeet, daya, henry, polash, narin, ayon, max, walter, faris, jani, scott, diana, jagu, moni, alan, khil, eric, ayuba, laxi, raju, lina, manab, saiful, joe, roki, chris, hannah, olip, ronald, rezart, keith, jose, boby, rono, sharon, rajat, kayle, niki, kakir, raje, larry, orko, joan, cleo, dipak, soma, judith, nathan, epic, arus, ila, eliza, leo, faria, jack, posto, pijus, roger, tyler, ryan, tusi, siam}

User interaction

Complimented form of a weight

The logical transformation in Section 4.3.2 shows that connected weights in a conjunction equals exactly connected weights in a disjunction. Therefore, the experimental procedure and results will be equal for the same parameter settings as described in C.3.1.

Weights based on influence

Feedback for constructing a new cluster

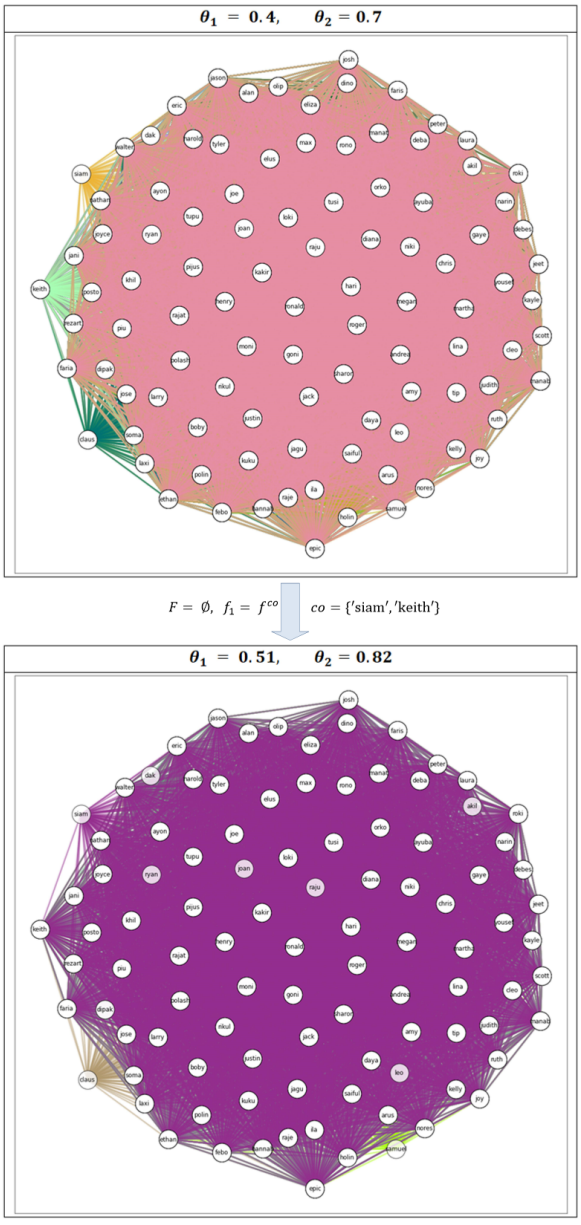


Fig. C.33 Clustering based on f^{co} on a 100-people network in *disjunction*.

Feedback for removing a person from a cluster

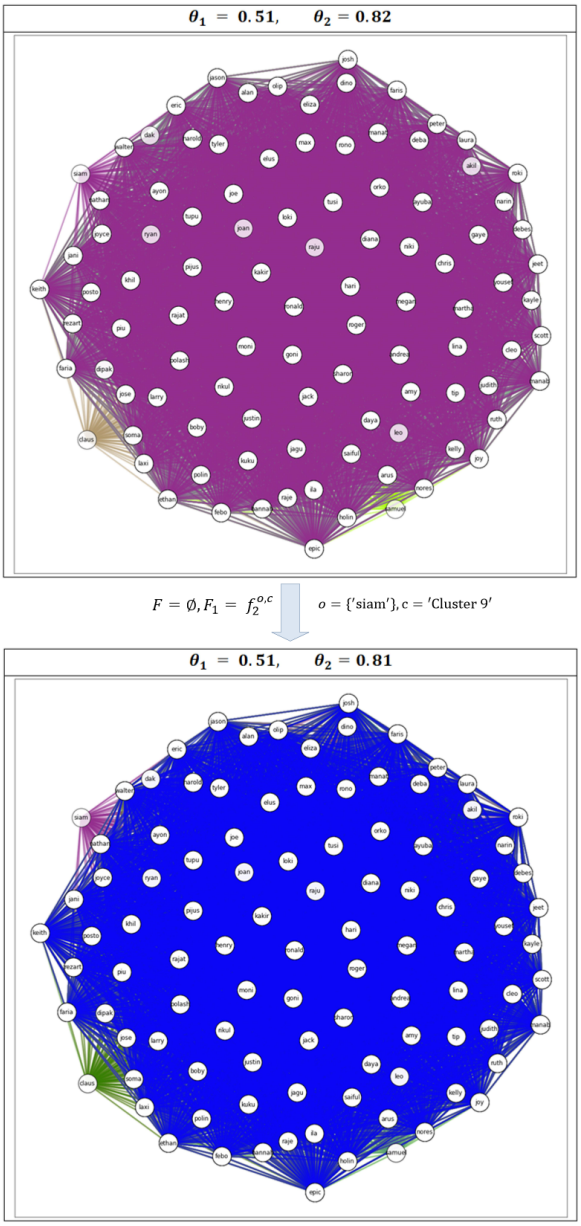


Fig. C.34 Clustering based on $f^{o,c}$ on a 100-people network in *disjunction*.

Appendix D

List of Publications

This section presents a chronological arrangement of selected publications relevant to the dissertation.

1. Sanjit Kumar Saha, Ingo Schmitt: **A Relevance Feedback-Based Approach for Non-TI Clustering**. In: *International Conference on Advanced Data Mining and Applications (ADMA), Lecture Notes in Computer Science - Springer*, 13088, pp. 381–393, Sydney, Australia, 2022.

2. Sanjit Kumar Saha, Ingo Schmitt: **Quantitative Weighting Approach for Non-TI Clustering**. In: *The International Workshop on Web Search and Data Mining (WSDM), Procedia Computer Science*, 184, pp. 966–971, Warsaw, Poland, 2021.

3. Sanjit Kumar Saha, Ingo Schmitt: **Non-TI Clustering in the Context of Social Networks**. In: *The International Workshop on Web Search and Data Mining (WSDM), Procedia Computer Science*, 170, pp. 1186–1191, Warsaw, Poland, 2020.

Bibliography

[1] Ahn, Y.-Y., Bagrow, J. P., and Jørgensen, S. L. (2010). Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764.

[2] Amelio, A. and Pizzuti, C. (2014). Overlapping community discovery methods: A survey. In Şule Gündüz-Öğüdücü and Şima Etaner-Uyar, A., editors, *Social Networks: Analysis and Case Studies*, volume 25, page 105–125. Lecture Notes in Social Networks, Springer.

[3] Ash, R. B. (2008). *Basic Probability Theory*. Dover Publications, New York.

[4] Attar, R. and Fraenkel, A. S. (1977). Local feedback in full-text retrieval systems. *Journal of the ACM*, 24(3):397–417.

[5] Bader, D. A., Meyerhenke, H., Sanders, P., Schulz, C., Kappes, A., and Wagner, D. (2014). Benchmarking for graph clustering and partitioning. In *Encyclopedia of Social Network Analysis and Mining*, pages 73–82. Springer New York, NY.

[6] Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search*. Harlow : Pearson Addison-Wesley [u.a.], second edition.

[7] Beltrametti, E. G. and Fraassen, B. C. V. (1981). *Current Issues in Quantum Logic*. Springer.

[8] Birkhoff, G. and Neumann, J. V. (1936). The logic of quantum mechanics. *Annals of Mathematics*, 37:823–843.

[9] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[10] Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. Elsevier, New York.

[11] Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

[12] Chen, J. and Yuan, B. (2006). Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 22:2283–2290.

[13] Danon, L., Díaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008.

[14] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26:297–302.

[15] Dieudonné, J. A. (1960). *Foundations of Modern Analysis*. Academic Press, first edition.

[16] Dirac, P. (1958). *The Principles of Quantum Mechanics*. Oxford University Press.

[17] Dominich, S. (2008). *The Modern Algebra of Information Retrieval*. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg.

[18] Eppstein, D., Löffler, M., and Strash, D. (2013). Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18.

[19] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231.

[20] Everitt, B. S. (1980). *Cluster Analysis*. Heineman Educational Books Ltd, London, second edition.

[21] Fagin, R. and Wimmers, E. L. (2000). A formula for incorporating weights into scoring rules. *Theoretical Computer Science*, 239(2):309–338.

[22] Falkowski, T., Barth, A., and Spiliopoulou, M. (2007). Dengraph: A density-based community detection algorithm. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 112–115.

[23] Fortunato, S. and Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports*, 659:1–44.

[24] Fowlkes, E. B. and Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569.

[25] Fred, A. L. N. and Jain, A. K. (2003). Robust data clustering. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II–II.

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

[27] Goyal, S., van der Leij, M. J., and Moraga-Gonzalez, J. L. (2006). Economics: An emerging small world. *Journal of political economy*, 114(10):403–412.

[28] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145.

[29] Hao, F., Park, D.-S., Min, G., Jeong, Y.-S., and Park, J.-H. (2016). K-cliques mining in dynamic social networks based on triadic formal concept analysis. *Neurocomputing*, 209(C):57–66.

[30] Hearst, M. A. (2009). *Search User Interfaces*. Cambridge University Press.

[31] Himmel, A.-S., Molter, H., Niedermeier, R., and Sorge, M. (2017). Adapting the bron–kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):1–16.

[32] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

[33] Jain, A. and Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ.

[34] Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons.

[35] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

[36] Kloster, K. and Gleich, D. F. (2014). Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1386–1395, New York, NY, USA. Association for Computing Machinery.

[37] Kraft, D. (1988). A software package for sequential quadratic programming. Technical report, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

[38] Lee, J. H. (1994). Properties of extended boolean models in information retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 182–190, Berlin, Heidelberg. Springer-Verlag.

[39] Lee, J. H., Kim, M. H., and Lee, Y. J. (1994). Ranking documents in thesaurus-based boolean retrieval systems. *Information Processing & Management*, 30(1):79–91.

[40] Leskovec, J. and Mcauley, J. (2012). Learning to discover social circles in ego networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25, page 548–556. Curran Associates, Inc.

[41] Liu, Y., Zhang, D., Lu, G., and Ma, W. Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282.

[42] Lock, P. F. and Hardegree, G. M. (1985a). Connections among quantum logics, part 1: Quantum propositional logics. *International Journal of Theoretical Physics*, 24:43–53.

[43] Lock, P. F. and Hardegree, G. M. (1985b). "connections among quantum logics, part 2: Quantum event logics. *International Journal of Theoretical Physics*, 24:55–61.

[44] Meilă, M. (2003). Comparing clusterings by the variation of information. In *COLT*.

[45] Meilă, M. and Heckerman, D. (2001). An experimental comparison of model-based clustering methods. *Machine Learning*, 42:9–29.

[46] Mendenhall, W., Beaver, R. J., and Beaver, B. M. (2020). *Introduction to Probability and Statistics*. Cengage Learning, fifteenth edition.

[47] Miller, I., Miller, M., and Beaver, B. M. (1999). *John E. Freund's Mathematical Statistics*. Prentice Hall, Upper Saddle River, New Jersey 07458, sixth edition.

[48] Mirghorbani, M. and Krokhmal, P. (2013). On finding k-cliques in k-partite graphs. *Optimization Letters*, 7(6):1155–1165.

[49] Mirkin, B. (2001). Eleven ways to look at the chi-squared coefficient for contingency tables. *The American Statistician*, 55(2):111–120.

[50] Neumann, J. V. (1932). Grundlagen der quantenmechanik. *New York : Springer Verlag*.

[51] Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113.

[52] Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

[53] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

[54] Rocchio, J. (1971). Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, pages 313–323.

[55] Rui, Y., Huang, T. S., Ortega, M., and Mehrotra, S. (1998). Relevance feedback: a power tool for interactive content-based image retrievals. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655.

[56] Ruthven, I. (2001). Abduction, explanation and relevance feedback. *Unpublished doctoral dissertation, University of Glasgow, Glasgow, UK*.

[57] Saha, S. K. and Schmitt, I. (2020). Non-TI clustering in the context of social networks. *Procedia Computer Science*, 170:1186–1191.

[58] Saha, S. K. and Schmitt, I. (2021). Quantitative weighting approach for Non-TI clustering. *Procedia Computer Science*, 184:966–971.

[59] Saha, S. K. and Schmitt, I. (2022). A relevance feedback-based approach for non-ti clustering. In *Advanced Data Mining and Applications*, pages 381–393, Cham. Springer International Publishing.

[60] Schmitt, I. (2008). QQL: A DB&IR query language. *VLDB J.*, 17(1):39–56.

[61] Schmitt, I. (2019). *Incorporating Weights into a Quantum-Logic-Based Query Language*, pages 129–143. Springer International Publishing.

[62] Schmitt, I., Zellhofer, D., and Nurnberger, A. (2008). Towards quantum logic based multimedia retrieval. In *NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 1–6.

[63] Schöning, U. (1989). *Logik für Informatiker*. Bibliographisches Institut, Mannheim.

[64] Siefkes, D. (1990). *Formalisieren und Beweisen: Logik für Informatiker*. Braunschweig: Vieweg-Verlag.

[65] Strehl, A. and Ghosh, J. (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617.

[66] Theodoridis, S. and Koutroubas, K. (1999). *Pattern recognition*. Academic Press, New York.

[67] Thurstone, L. L. (1927). A law of comparative judgement. *Psychology Review*, 34:278–286.

[68] Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42.

[69] van Rijsbergen, C. J. (2004). *The Geometry of Information Retrieval*. Cambridge University Press.

[70] Wittgenstein, L. (2004). *Tractatus logicophilosophicus: Logisch-philosophische Abhandlung*. Frankfurt am Main : Suhrkamp.

[71] Xie, J., Kelley, S., and Szymanski, B. K. (2013). Overlapping community detection in networks. *ACM Computing Surveys*, 45(4):1–35.

[72] Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 4–11.

[73] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

[74] Zhu, X. S. and Huang, T. S. (2003). Relevance feedback in image retrieval: a comprehensive review. *Multimedia System*, 8(6):536–544.

[75] Ziegler, M. (2005). Quantum logic: Order structures in quantum mechanics. Technical report, University Paderborn, Germany.