UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

eXascale Infolab

UNIVERSITY OF FRIBOURG

SOCIAL MEDIA ANALYTICS

Project Report: Swiss Train Network

Author:
Mattias Dürrmeier, Christian Galley, Olivia Lecomte

Supervisor:
Dr. Mourad Khayati

Co-Supervisor:
Manuel Mondal

May 28, 2024

eXascale Infolab
Department of Informatics

Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland
phone +41 (26) 300 84 65 • diuf-secr@unifr.ch • www3.unifr.ch/inf

Contents

List of Figures

Chapter 1

Introduction

Information is more accessible than ever thanks to increased data collection and the reduced cost of data storage. However, with this abundance of data comes challenges related to analysis and interpretation. Due to their effectiveness at illustrating relationships, information is often stored as networks. Graph analysis, which can include network exploration, community detection, and graph visualization, helps in understanding the underlying patterns in the data.

Networks can be constructed from diverse datasets to solve complicated problems. We analyze the Swiss train network including all trains operated by Swiss companies in Switzerland and its neighboring countries. Through this analysis, we aim to evaluate the network's underlying structure and to understand how stations are clustered. This project also measures the performance of various community detection algorithms. We implemented our own version of the Louvain algorithm, and compared its results against other techniques, such as Girvan-Newman and Leiden. Since our communities do not have a ground truth, we anticipated that the networks would likely be distinguished based on a combination of geographic proximity, language region, urbanization, and other factors. We use the silhouette index to measure the quality of clustering. The source code for this project is publicly available.[1]

[1] https://github.com/mattiasduerrmeier/sbb-analytics.git

Chapter 2

Background

Community detection algorithms are employed to identify underlying groups present in social graphs. In this chapter, we briefly describe Leiden, a recently published technique that improves on Louvain. We outline the silhouette index used to assess the clustering quality of community detection algorithms.

2.1 Leiden Algorithm

Leiden was developed as an extension of the Louvain algorithm: it guarantees well-connected communities [1]. It is typically faster than Louvain. In the worst case, they share the same computational complexity ($\mathcal{O}(n \log n)$). The authors of the Leiden algorithm use the Constant Potts Model (CPM) as their quality function but specify that any quality function can be used with Leiden. For this project, we use the modularity function across algorithms.

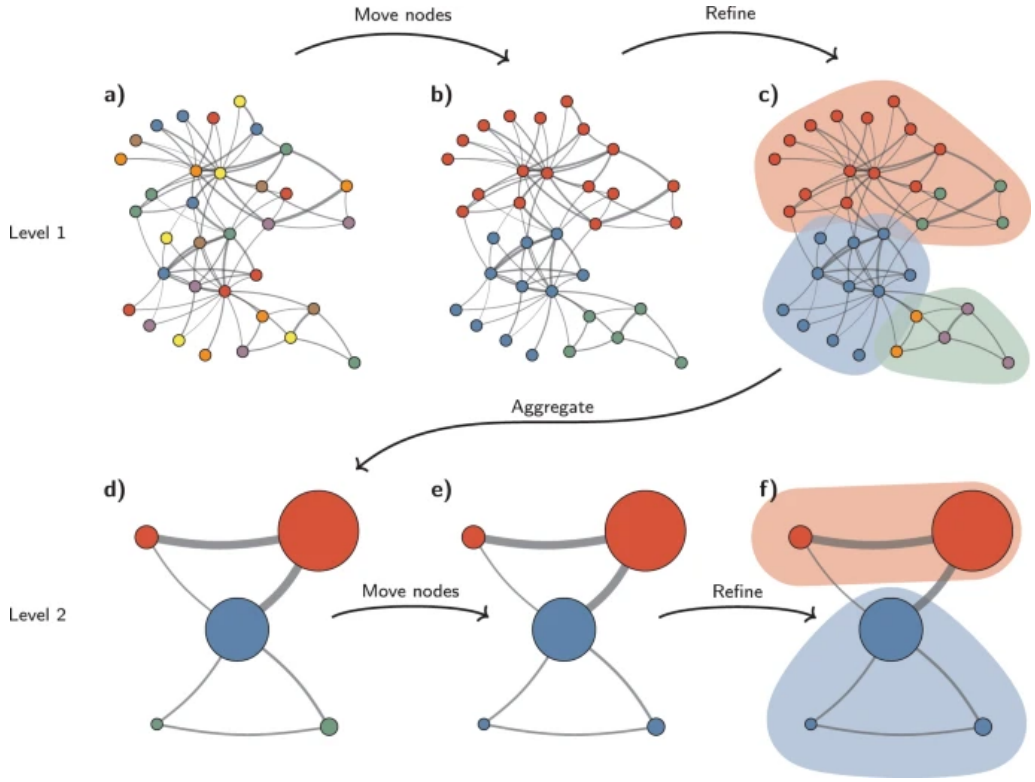

FIGURE 2.1: Visualization of the Leiden algorithm [1]

Leiden is an agglomerative community detection algorithm where each node is initialized as its own community. To make passages faster, the Leiden algorithm

uses a more efficient node-movement function. On the first iteration, Leiden visits every node in the network. On further iterations, however, Leiden visits only the neighbors of nodes that have changed communities. Once moving the nodes between communities is done, Leiden starts a refinement phase. The goal of this step is to ensure internal connectivity inside a community. During the refinement phase, each node is randomly assigned to a sub-community if the two communities are well connected. Once the refinement ends, the resulting communities are aggregated into hyper-nodes. The passages are repeated on the hyper-graph until the modularity gain can not be increased further.

Leiden brings two improvements over Louvain. First, the efficient node-movement function increases the speed of each passage. Second, Leiden can produce higher-quality communities thanks to the refinement step. While Louvain often returns internally disconnected communities, the Leiden algorithm remediates this during the refinement phase. These partitions are more meaningful and stable than the communities found by Louvain.

2.2 Silhouette Index

The clustering quality is an important factor to consider when choosing a community detection algorithm. However, the evaluation of communities for graphs without ground truth can be challenging. We therefore measure the quality of clustering using the silhouette index [2], a technique used to evaluate clustering algorithms.

This score function considers both the intra-community cohesion and inter-community separation. The cohesion of a cluster is computed as the average distance between each node and the centroid of its cluster. The separation is the average distance between each node and the centroid of the nearest neighboring community. The function for the silhouette index then computes and averages the cohesion and separation metrics for all nodes in the graph. The resulting silhouette index is bound between $[-1, 1]$ with higher values linked to better quality clustering. Negative silhouette scores indicate that the node are not well clustered.

Chapter 3

Swiss Train Network

This chapter covers the steps to construct the Swiss train network from the timetable of public transport in General Transit Feed Specification (GTFS) format. Additionally, it presents some of the results of the network exploration.

3.1 Data Source

The SBB does not offer a complete network of trains operated by Swiss companies. As a result, we constructed a network using other data sources.

The Open Data Platform Mobility Switzerland provides the timetable of all public transport in Switzerland. This large dataset is freely available in the GTFS format, an international standard for route planning systems. It includes static information of all forms of public transportation in Switzerland.

The GTFS Static format is organized as a relational database, with each table containing information related to one aspect of the travel such as the station names and their geographic location. The files and their interactions must adhere to the standard depicted by the GTFS relational database. This approach ensures that transportation network data is compatible with other international transit systems.

3.2 Network Construction

Of all the available files, five are crucial for the construction of the train network: agencies, routes, trips, stop times, and stops. We load this selection of the Swiss transport data into Neo4j, a graph database management system. The decision to use such a system allows for simplified queries, with the added advantage of graph persistence. The dataset is clean from noise, outliers and missing values, due to the GTFS format. We access the data via two Cypher queries.

The first query is designed to construct a table of station names and their corresponding coordinates. First, it matches the files according to the relationships detailed by the GTFS format. It then filters these results to only include train data. These trains, though all operated by Swiss companies, service stations within Switzerland and its neighboring countries. At this stage, the results contain duplicate station names with slight variations in coordinates. To remediate this, only the first instance of each station is included in our dataset. The resulting data is exported as a GeoJSON file for further processing. The coordinates are attributed to each node which allows for enriched graph visualization and community detection benchmarking.

The second query aims to create an edge list of the train network. As with the first query, the second joins the five transit data files according to their relationships and is filtered to include only train data. It then returns unique trip paths in the train routes with the station names and their sequence in the path. The final edge list is

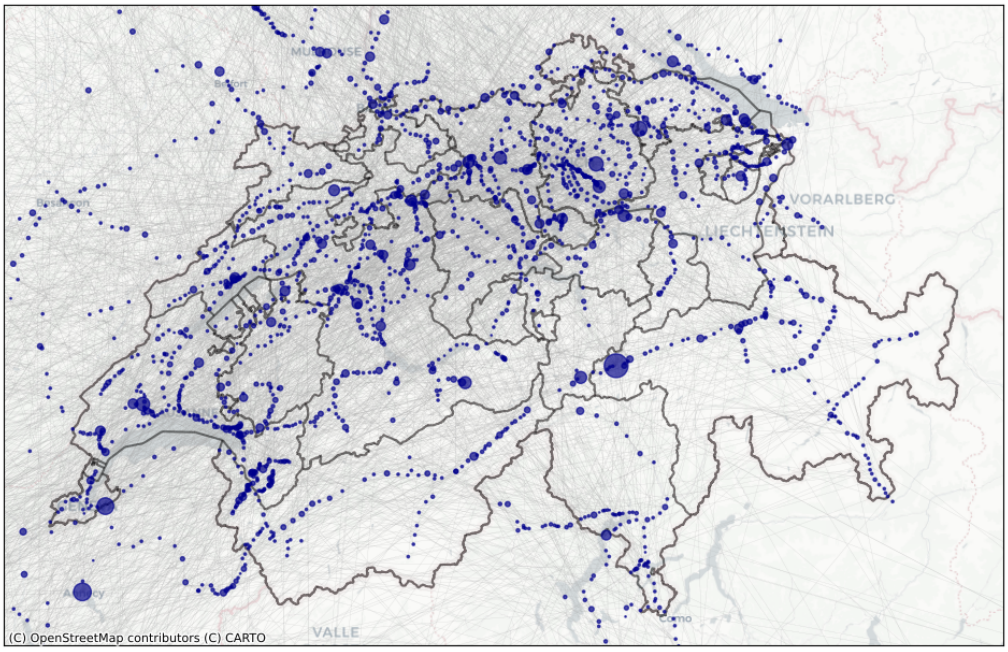

FIGURE 3.1: Swiss Train Network

processed in Python by extracting the unique edges for each train route. The result contains the train stations as nodes and the railways as edges.

3.3 Network Exploration

The final undirected and unweighted graph consists of 3152 nodes and 4678 edges. The network is plotted on top of the Swiss map in figure 3.1 where the node size is dependent on the station's degrees. We perform some network exploration to understand the characteristics of the network.

TABLE 3.1: Top 10 Stations with the Highest Degrees

Station	Degree
Lyon Part Dieu (FR)	40
Zürich HB (CH)	36
Strasbourg (FR)	27
Bern (CH)	27
Olten (CH)	25
Paris Gare de Lyon (FR)	24
Dijon (FR)	22
Luzern (CH)	21
Nancy (FR)	20
Winterthur (CH)	19

First, we look at the node degrees in the graph. Table 3.1 shows the ten stations with the highest degrees. These stations are equally distributed across cities in Switzerland and France. The stations with the highest degrees are Lyon Part Dieu in France and Zürich HB in Switzerland. Interestingly, the five cities with the highest

degrees are located in the German-speaking regions of Switzerland. There are a total of 123 stations with a degree of one, representing the ends of train lines.

Station	Betweenness
Zürich HB (CH)	0.40
Basel SBB CH)	0.35
Mulhouse (FR)	0.32
Paris Gare de Lyon (FR)	0.19
Bern (CH)	0.17
...	...
Schorndorf (DE)	0.0
Neuffen (DE)	0.0
Nürtingen (DE)	0.0

Station	Closeness
Zürich HB (CH)	0.15
Basel SBB (CH)	0.15
Olten (CH)	0.14
Mulhouse (FR)	0.14
Zürich Altstetten (CH)	0.14
...	...
Nürtingen (DE)	0.0
Pied du barrage (CH)	0.0
Les Montuires (CH)	0.0

TABLE 3.2: Betweenness and Closeness Centrality ordered by score.

Table 3.2 summarizes the centrality measures of the train network. The stations with the highest betweenness centrality measures are in major cities on either side of Switzerland's borders: these stations serve as the gateways in and out of Switzerland. This contrasts with the 554 stations with a betweenness centrality score of zero.

The closeness centrality measure similarly demonstrates that stations that are on average closest to all other stations are found in major cities along Switzerland's borders. We found 65 stations with a closeness centrality near zero. Three out of the top five stations with the highest closeness centrality are also present in the top five stations with the highest betweenness centrality. While the stations with the lowest scores for both centrality measures highlight areas that are infrequently serviced by Swiss trains, the scores are not necessarily representative of their centrality in the European train network.

Chapter 4

Network Analysis

We analyze the performance of the NetworkX implementations for Girvan-Newman and Louvain algorithms and compare them with our implementation of Louvain. We also measure the reference implementation of the Leiden algorithm. The performance of each community detection technique is assessed based on runtime and the quality of clustering at each iteration. We perform all techniques on the network's largest connected component. This adjustment effectively discards the few rural train lines disconnected from the main network. By evaluating the largest connected component, we achieve a decrease in runtime and denser clustering compared with the full graph.

4.1 Runtime Performance

To accurately evaluate the performance of each algorithm, we run the algorithms 10 times each and average the results. Unfortunately, this was not feasible for the Girvan-Newman algorithm as it did not converge within a reasonable amount of time. The performance results of the Girvan-Newman algorithm are therefore only reported for the iterations that were completed within the first 8 hours of running the algorithm.

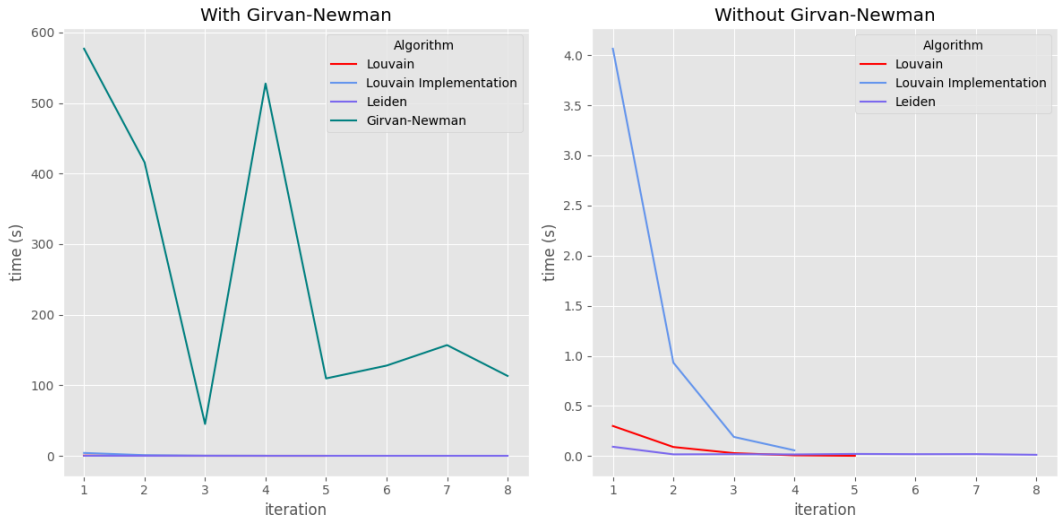

FIGURE 4.1: Execution time per iteration

Figure 4.1 shows the runtime for each iteration. Girvan-Newman is significantly slower than the other algorithms, requiring nearly 10 minutes for the first iteration. The overall runtimes for our implementation of Louvain are slower than both the

reference Louvain and Leiden algorithms. A difference in runtime between our implementation and the reference algorithms is especially pronounced for the first iteration after which the runtimes steeply decline. This is partly because our implementation aims to mimic the results in communities of the NetworkX implementation. Further speed optimizations are possible. Leiden exhibits the best runtime performance per iteration but occasionally requires more iterations than Louvain. Whereas Louvain usually converges in four to five iterations for our network, the number of iterations for the Leiden algorithm typically varies between three and eight. The slight difference in runtime between Leiden and Louvain algorithms is attributed to using a more efficient node-moving function, more prominent during the first few iterations. Unlike both Louvain implementations, which are coded entirely in Python, the reference Leiden algorithm is optimized for performance by using C++ for computationally expensive steps.

4.2 Clustering Quality

Since there is no ground truth for the clusters in our network, the quality of clustering was determined by the silhouette index. For our analysis, we compute the silhouette scores with two different distance functions. First, we create a matrix with the shortest paths between each pair of nodes and use the number of hops as the distance metric. Secondly, we compute the Euclidean distance between the geographical coordinates of the stations. In both scenarios, the labels refer to the list of communities found by the algorithms at each iteration.

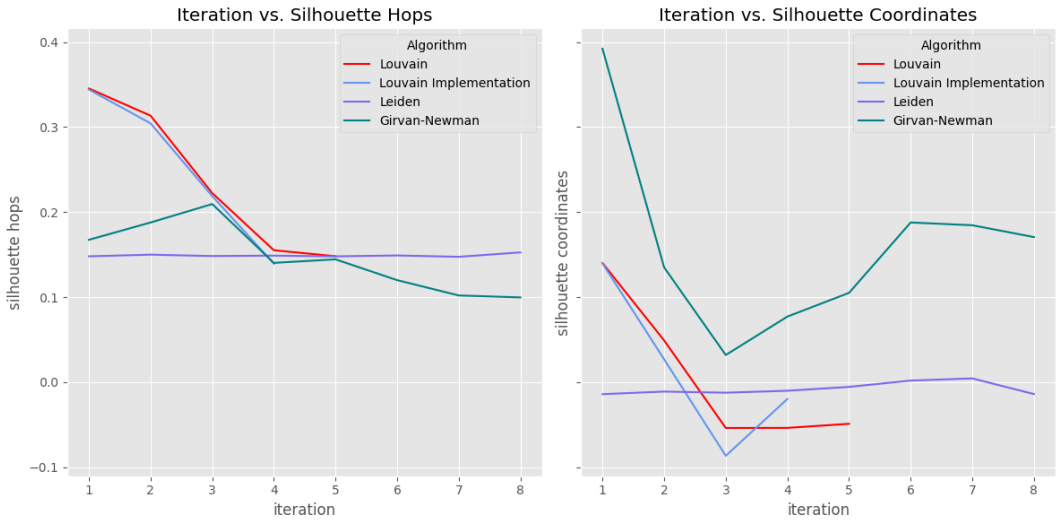

FIGURE 4.2: Silhouette index per iteration

Our implementation of Louvain performs similarly to the NetworkX version for both distance metrics. When comparing the silhouette scores derived from the shortest paths, both the Louvain algorithms exhibit a decrease in clustering quality at each iteration. The reference Louvain converges with a silhouette score of 0.15 after five iterations. Our implementation obtains a score of 0.14 in four iterations, a negligible difference. Interestingly, the silhouette score of the NetworkX Louvain algorithm plateaus after four iterations, suggesting that the communities found at the end are only marginally better. While the Girvan-Newman and Leiden algorithms

score lower silhouette indexes after the first iteration, their scores remain consistent across iterations, reaching a score similar to that of the Louvain algorithms.

The second distance metric, which examined the physical distance between stations, led to poorer performance for most algorithms. Again, the Louvain scores show a downward trend and the Leiden scores are consistently close to 0. Both fluctuated around zero and indicate that the found communities are unrelated to the topology of Switzerland. Though the Girvan-Newman algorithm appears to capture aspects of the network's topology, the plot only displays a fraction of the iterations computed by the algorithm and is therefore not representative of the evolution of the clustering quality. A plot of the first 900 iterations of Girvan-Newman iterations is available in the appendices 6.1.

While mostly positive, the low silhouette scores for both functions indicate poor clustering quality across iterations and techniques. The best silhouette scores occur when stations are clustered in a single community or individually. This suggests that maximizing the modularity or minimizing the edge betweenness does not lead to the best clustering.

Chapter 5

Conclusion

5.1 Summary

In this report, we demonstrate the construction process of the Swiss train network from a GTFS timetable and detail the results of our network exploration in chapter 3. Interestingly, the nodes with the highest degrees and centrality measures are mostly located in the German-speaking region of Switzerland and in France.

Standard community detection algorithms could not produce high-quality clusters within the Swiss train network. Our implementation of the Louvain algorithm performs similarly to the NetworkX implementation in terms of runtime and clustering quality.

5.2 Limitations and Future Work

Future work could investigate all forms of public transport in Switzerland, or expand the network to include train companies of neighboring countries. Scaling the network would require adding the desired GTFS data to Neo4j and modifying the database queries. The rest of the pipeline could remain unchanged.

Our implementations of the Louvain have not been optimized for performance and could therefore be significantly improved to match the performance of the reference algorithms. The reference Leiden algorithm is a prime example of how a performance-aware implementation significantly reduces the runtime of such algorithms.

The evaluated algorithms did not consider the features of the transportation network which may have resulted in poor clustering quality. Expanding the graph to include additional features such as scheduling and topology, as well as employing more advanced techniques such as graph neural networks which can leverage this added information, could yield better communities. To obtain better clustering results, we recommend creating solutions tailored for this problem.

Chapter 6

Appendix

6.1 Girvan-Newman Plots

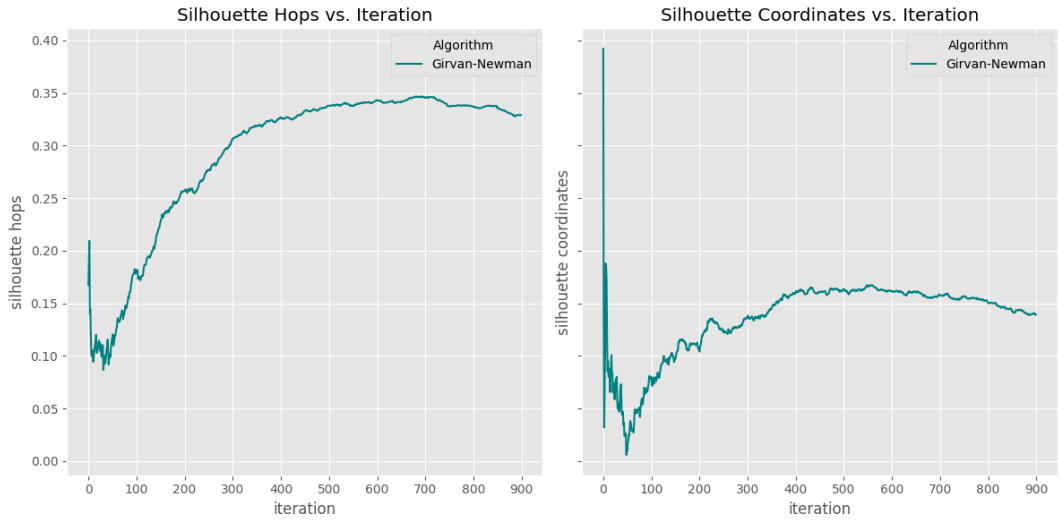

FIGURE 6.1: Silhouette scores for Girvan-Newman

6.2 AED Project

Initially, our focus was on mapping Automated External Defibrillators (AED) in Zürich to determine the most strategic locations for new AED placements. After researching the topic further, we realized that the data we had access to combined with the randomness of cardiac events and the limited coverage of each AED meant that AEDs could not be strategically placed based on network analysis. Although we did not test it, we hypothesized that the best solution involved placing AEDs at every intersection within a 300m radius of one another. Though network analysis could have been used to determine the shortest paths between a randomly initialized cardiac event and the nearest AED, this problem did not meet the project requirements and was therefore discarded.

While working on this idea, we sourced the locations of AEDs and all pedestrian walkways in Zürich from Open Street Map. To further enrich our data, we were in contact with SWISSRECA (Swiss Registry of Cardiac Arrests) who agreed to give us access to their database. We also performed basic network exploration and created graph visualizations where we mapped the locations of active AEDs over a map of all pedestrian walkways.

6.3 Task Distribution

TABLE 6.1: Task Distribution

Task	Contributor(s)
SBB Network Sourcing	Mattias Dürrmeier, Olivia Lecomte
Neo4j	Mattias Dürrmeier, Olivia Lecomte
Network Exploration	Olivia Lecomte, Mattias Dürrmeier
Community Detection	Mattias Dürrmeier, Christian Galley, Olivia Lecomte
Community Detection Evaluation	Olivia Lecomte, Mattias Dürrmeier
SBB Graph Visualization	Mattias Dürrmeier, Olivia Lecomte
Louvain Implementation	Mattias Dürrmeier, Christian Galley, Olivia Lecomte
Leiden Implementation (unfinished)	Christian Galley, Mattias Dürrmeier, Olivia Lecomte

Bibliography

[1] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific reports* 9.1 (2019), p. 5233.

[2] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social Media Mining: An Introduction*. New York, NY, USA: Cambridge University Press, 2014. ISBN: 1107018854, 9781107018853.