Network community detection on Wikipedia

PGM 2024/2025

Melvin Gode[1], Antoine Sicard[1], and Andrej Perković[1]

[1]Ecole Normale Supérieure, Paris-Saclay

Abstract

Wikipedia is one of the most visited websites in the world and serves as a vast knowledge repository organized accross millions of interconnected articles. In this work, we explored and implemented a variety of hierarchical clustering algorithms on a graph representation of Wikipedia. Our study highlights the difference in behavior of the selected methods and most interestingly, presents highly interpretable results in the partitioning of the studied dataset into different topics. Additionally, we discuss practical considerations which limit the implementation of our approaches for clustering on large-scale graphs.

1 Introduction

Wikipedia is an online multilingual collaborative encyclopedia created in 2001. It is one of the most visited websites in the world, with more than 700 millions visits in 2022. Wikipedia is highly valuable for education, research of free documentation (Head and Eisenberg 2010; Xiao and Askin 2012). In English language, Wikipedia hosts over 7 million articles on various topics including cultures, art, geography, society and sciences (*Wikipedia* 2024). In this sense, Wikipedia represents one of the largest and most recognizable reference resources of current times.

Articles on Wikipedia are interconnected through hyperlinks, forming a huge graph structure where nodes represent articles, and edges represent hyperlinks. Developments in clustering algorithms offer the possibility to analyze and better understand this complexity enlightening the relationships and connections within this dense network.
Our report is based on the paper of Schaeffer 2007 which provides a comprehensive overview of clustering methods in graphs. She presents important concepts such as metrics, used algorithms, and classical applications. Specifically, we were interested in this paper because it covers a huge number of graph-clustering methods within various approaches. This catalogue includes spectral methods based on manipulations of the graph Laplacian, distance based methods in cases where we have quantitative information on the graph's vertices (which here is not the case), and a number of hierarchical methods, whether top-down or bottom-up.
It is this last category of methods -hierarchical ones- we chose to apply to our project. We were particularly interested in this class of methods because they provide highly interpretable and multi-level structured cluster formation, enabling a deep understanding of the relationships and organization of the articles. We thus decided to implement 3 different types of clustering methods for graphs: an agglomerative global clustering method, based on bottom-up node set merging, and two different divisive global clustering methods (also called top-down cluster division), based on edge betweenness and Markov chains, respectively.

Contributions: Literature review : MG, AS; report writing and poster creation: AS, MG; search for available database: MG, AS; implementation of algorithms: MG AS, AP; result analysis: AS, MG. Most of the codes to do the clustering was hand-made, we tried to use as few as possible existing libraries.

2 Selected approaches

2.1 Dataset and quality measure

In order to study the structure of Wikipedia articles, we downloaded a dataset published by Stanford (`https://snap.stanford.edu/data/wiki-topcats.html`). This dataset is composed of 2 files: "wiki-topcats.txt" which contains all the links between the different Wikipedia articles and "wiki-topcats-page-names" which contains all the names of wikipedia articles. We imported a graph with nodes representing wikipedia articles and edges representing *mutual* hyperlinks between them. We first analyzed the basic properties of this graph. We found that the graph contains 1,791,489 articles with 25,447,873 hyperlinks. We found 3,666 self-connected articles. Importantly, this dataset only includes the largest connected component of the full graph, which means that there exists at least one path to connect one article to another. In addition, this graph is by construction undirected.

A well known quality measure for graph clustering is **modularity**. Modularity is defined in a confusingly high number of different ways but they all share the same idea : maximizing the number of edges interior to clusters while minimizing the number of edges between different clusters. The formula we chose to use is the one from Newman 2011 which is :

$$\mathcal{M} = \sum_{c=1}^{K} \left[\frac{L_c}{m} - \left(\frac{k_c}{2m} \right)^2 \right] \tag{2.1}$$

Where L_c is the number of edges within cluster c and k_c is the sum of degrees of nodes belonging to cluster c, (m is the total number of edges).

2.2 Edge betweenness

We first used a global divisive graph clustering method based on edge betweenness. It consists first, in estimating the betweenness of each edge in the graph.
The **betweenness of an edge** e is an adaptation of the definition of the betweenness of a node and is defined as the proportion of shortest paths passing through that edge between all pairs of nodes in the graph.

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t \mid e)}{\sigma(s,t)} = \sum_{s,t \in V} \delta(s,t \mid e) \tag{2.2}$$

where V is the set of nodes, $\sigma(s,t)$ is the number of shortest paths between s and t and $\sigma(s,t \mid e)$ is the number of those paths with contain e. The ratio $\frac{\sigma(s,t|e)}{\sigma(s,t)}$ denoted $\delta(s,t \mid e)$ represents the **pairwise dependencies** of s and t on an intermediary edge e.

After computing betweenness for all edges, the edge with the highest value is removed. These two step are repeated until all egdes have been removed or a stopping criterion is met (Newman and Girvan 2002; Girvan and Newman 2002). At each iteration, clusters are defined by the different connected components in the graph. The principle of this method relies on the fact that we should observe denser connectivity within clusters than between them. Following this hypothesis, removing high betweenness "bridge" edges should not disconnect nodes within clusters while separating those lying in different groups.

To implement this method, we first wrote a first "naive" algorithm from scratch to compute the edge betweenness of a graph (see attached notebook) based on the general definition of betweenness from Schaeffer 2007. Briefly, we first search for all shortest paths between any two nodes of the dataset using an adaptation of a breadth-first search algorithm and save the list of used edges for each of them. At the same time, we also count the number of parallel shortest paths connecting any two nodes with the same number of steps (multiple geodesic paths). Then, the betweenness of each edge was computed recursively, by adding 1 each time the edge appears in each identified shortest path, or $1/number_{parallel\_paths}$ in case of several geodesics. This allowed to give an equal weight of $1/number_{parallel\_paths}$ to all parallel shortest paths between a pair of nodes, as suggested by Schaeffer 2007.

Brandes 2001 suggests an optimized version of the betweenness computation algorithm based on the fact that pairwise dependencies can be aggregated without computing all of them explicitely. After a bread-first search, the algorithm visits all nodes in reverse order of their discovery, to accumulate dependencies using a recursive relation (see Brandes 2001 for further details, for space reason, we decided not to add all the explicit formulas in the report). In brief, this recursive relation asserts that the dependency of a node s on some node v can be compiled from dependencies on nodes one edge father away. Compared to previous existing algorithms which suffer from a $\mathcal{O}(n^3)$ (n = number of nodes) time complexity in a sense that they compute explicitly all of the cubic number of pair-dependencies (see for example Freeman 1977), this new version of the algorithm has a time complexity reduced to $\mathcal{O}(nm)$ (m = number of edges). We then wrote an algorithm to perform the divisive clustering adapting Brandes algorithm to edge betweenness, noting that during the back-propagation of dependencies, the value propagated through an edge corresponds to the betweenness of this edge. However, as the betweenness has to be computed again at each iteration before each edge removal, the whole algorithm for clustering based on betweenness has a time complexity of $\mathcal{O}(nm^2)$ in the worst case which was still too large to be run on the whole Wikipedia dataset, especially on our personal computers. To improve runtime, we decided to implement two additional hyperparameters in our clustering algorithm:

- $\underline{k_{sources}}$. When $k_{sources}$ is specified, instead of using all nodes, only $k_{sources}$ randomly selected nodes are used to compute the betweenness values for the entire graph.
- $\underline{m_{remove}}$ which represents the number of edges to be removed at each iteration of the algorithm, instead of deleting only the highest betweenness edge, we delete the top m_{remove} ones.

These two hyperparameters allows us to gain a huge amount of time by drastically reducing the number of computations. They are not ideal and they can induce notable approximations. In addition, we decided to work on only a subset of the entire graph but will come back to this later. After several attempts, we chose to work with $k_{sources} = none$ (all nodes are considered) but with $m_{remove} = 10$.

2.3 Markov Chains and Random Walks

Another global divisive method is considering Markov chains and random walks on the graph. The core idea is that a random walk passing through a vertex is more likely to visit other nodes in the same clusters before leaving it.

There have been several theoretical and practical approaches developed for simulating the random walks. Spielman and Teng 2004 propose an approximate graph partitioning algorithm based on estimating random walk distributions over the input graph. Meila and Shi 2000; Meilă and Shi 2001 demonstrate that the normalized cut of a graph, a measure of graph conductance, can be represented using the transition probabilities and stationary distribution of a random walk. This links the mathematical framework of random walks to cut-based clustering techniques. Orponen, Schaeffer, and Gaytán 2008 further relate absorption times in random walks to the eigenvectors of the graph's Laplacian. By approximating the Fiedler vector in Orponen and Schaeffer 2005, they compute absorption times, connecting the concept of random walks to spectral clustering, which is itself closely tied to cut-based methods. We decided to implement the algorithm developed by Van Dongen 2000. He develops the clusters using Markov chains. Effectively, with the help of a sequence of algebraic matrix operations, the algorithm diminishes the intercluster interactions while accentuating intraclusters connections through the simulation of stochastic flow.

The input parameters for the algorithm are the adjacency matrix A and the inflation parameter r_0. The latter determines the degree of coarseness of the cluster. This is one of the advantages of the approach, since it does not require the number of clusters to be determined in advance, but rather to arise naturally through stochastic flow on the graph. At the start of the algorithm, the adjacency matrix is column-normalized in order to make it stochastic and represent the transition probabilities. If the graph is undirected and edges do not have weight, the transition probabilities will be uniformly distributed across neighbors. The main part of the approach is alternation between

the expansion step and inflation step. More precisely, for $T_1 = M$ the column-normalized adjacency matrix and for $i = 1, 2, \ldots$:

$$T_{2i} = \mathrm{Exp}_{e(i)}(T_{2i-1}) \tag{2.3}$$

$$T_{2i+1} = \Gamma_{r(i)}(T_{2i}), \quad \text{for } i = 1, 2, \ldots \tag{2.4}$$

The $\mathrm{Exp}_{e(i)}$ represents the exponentiation of the matrix. Most of the implementations calculate the square, i.e. $e(i) = 2$. This step mimics the diffusion of flow by raising the stochastic matrix to a power, simulating random walks of increasing length. Operator $\Gamma_{r(i)}$ is the element-wise exponentiation, where $r(i) > 1$ always.

The stopping condition of the algorithm is the convergence of the matrix to a idempotent (stable) one. There are several ways to track the convergence. We have implemented the Frobenious norm of the difference of matrices from consecutive iterations $\|M^{(k+1)} - M^{(k)}\|_F$ with epsilon $\epsilon = 10^{-6}$.

Finally, the extraction of clusters is based on the theoretical guarantees of the properties of the final matrix. After alternating expansion and inflation operations, the stochastic matrix M converges to a state where it becomes idempotent, meaning $M \cdot M = M$. The idempotent matrix M can be interpreted as defining attractors, which are nodes or groups of nodes that the random walks consistently return to. These attractors form the cores of the clusters. For a given node i , its cluster is determined by identifying nodes j with which i shares significant flow (high values in M_{ij}).

We implemented the main elements of the algorithm in Python and compared the results with the official C library[1] from the paper authors. Although there are strong theoretical guaranties for the algorithm and simplicity at its core, practically it might have poor performance due to numerical instability. We implemented several strategies that slightly improved the performance like pruning out the edges with very small transition value and adding self-loops to nodes for stabilization.

2.4 Agglomerative Clustering

Still in the hierarchical clustering category, another attempted approach was "bottom-up" rather than the other "top-down" methods we present. This time instead of starting from one cluster including all nodes, we begin with n clusters, each containing only one node. Based on a criterion we will define later, we then find which pair of cluster matches best and merge them together. Similar to other proposed methods, we can keep iterating until we build a specified number of clusters or stop the algorithm when the criterion becomes too low for our liking.

The criterion we chose to explore here is neighborhood similarity, defined as :

$$NS(c_i, c_j) = \frac{\langle \Gamma(c_i); \Gamma(c_j) \rangle}{|c_i| \times |c_j|} \tag{2.5}$$

Where

$$\Gamma(c) = \bigcup_{x \in c} \{y | e_{x,y} \in E\} \tag{2.6}$$

The cluster neighborhood $\Gamma(c)$ can be chosen to include all neighbors of nodes within the cluster like in 2.6 or restricted to neighboring nodes *outside* of c (2.7).

$$\Gamma_{out}(c) = \bigcup_{x \in c} \{y | e_{x,y} \in E, y \notin c\} \tag{2.7}$$

These two criteria achieve different goals. While the "full neighborhood" one tends to produce densely connected clusters, the "outside neighborhood" can also produce clusters in which few nodes are linked but a lot of them are linked to a same "central node". We can observe this phenomenon in 1C with the articles on roads in Missouri which are almost all linked to the same article *"List of Missouri Highways"*, while almost no link appears between them. We can see with this example that making a cluster out of this group of nodes can semantically make sense but whether we decide to use this definition or the dense connection definition is up to us depending on the data and the types of clusters we are interested in.

We implemented from scratch methods using both versions of the criterion but decided to keep the outside neighborhood version as it seemed to us that the produced results were more coherent.
One important thing to note is that this outside neighborhood similarity method is not ideal for optimizing modularity as it does not explicitly focus on the edges directly linking pairs of clusters to merge them. As a consequence, it is expected that this algorithm will score lower in modularity.

At each iteration we have to update the criterion for the newly merged cluster only. This comes down to computing the product of this cluster's neighborhood with the matrix of all cluster's neighborhoods which is an

[1] https://github.com/micans/mcl.git

$\mathcal{O}(m)$-sparse matrix. Thus the time complexity of one iteration is $\mathcal{O}(m)^2$. At each iteration we reduce the number of clusters by one, starting from n, therefore, for a fixed number of clusters, the number of iteration is $\mathcal{O}(n)$. This gives us a total time complexity of $\mathcal{O}(n \cdot m)$.

Another important point is that at each iteration, the pair of clusters maximizing our criterion might not be unique. Therefore, we select one uniformly at random to break the tie and not be biased by the argmax function selecting the smallest valid indices. This introduces slight randomness to our implementation.

We also tried a third agglomerative approach based on an intersection over union criterion which we decided on not including since it appeared to yield very poor results, with cluster distributions being extremely top-heavy.

3 Results

3.1 Edge betweenness

We tested both our "naive" and "Brandes" clustering algorithm methods. As the first one takes a long time to run, we decided to focus on the more effective "Brandes" implementation. In order to know when to stop partitioning, we looked at the evolution of the modularity a each step, as well as the evolution of the number of clusters and the betweenness of removed edges.

Interestingly, we observe that the modularity first exhibits a stepwise increase which correlates with high betweenness values for the edges removed in these iterations (Figure 1A,B). This pattern arises from the fact that progressively removing edges with the highest betweenness often has no immediate big effect until a high-betweenness critical edge is removed, causing a significant cluster to split. This disconnection significantly impacts the modularity. Additionally, we identified a maximum modularity around the removal of 1250 edges. After this peak, modularity starts to decrease and the edge betweenness of removed edges remains at a relative low level. This modularity peak can be interpreted as the optimal clustering paradigm, and we decided to save the clustering result from this step, giving us a 5-big-cluster partitioning of the graph (the other ones have fewer than 5 nodes, Figure 1C). For each cluster, we then generated a wordcloud, giving a visual representation of the most common words in the names of incorporated articles (Figure S1) and we looked directly at several names and corresponding positions.

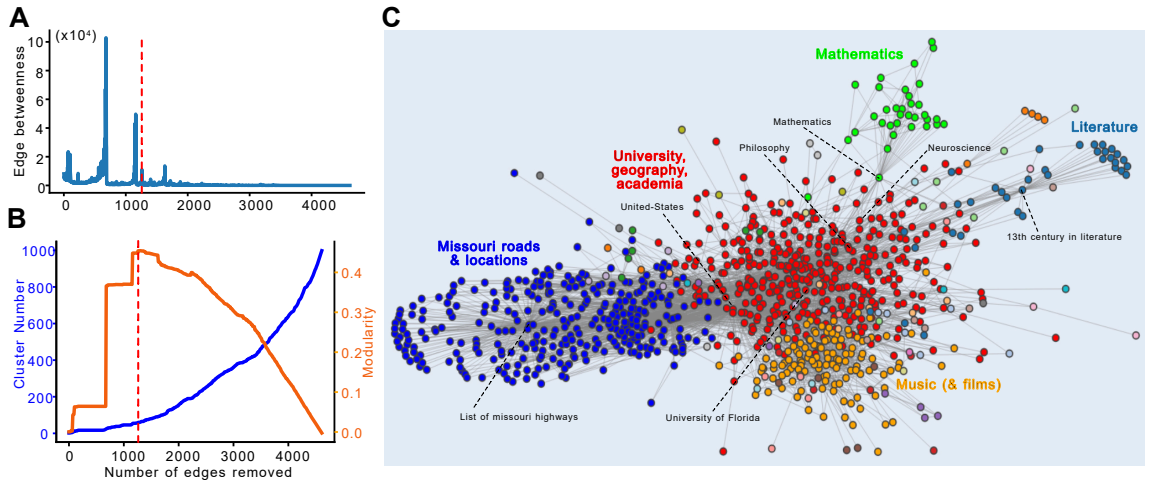

Figure 1: **Clustering with the betweenness approach**, using our implementation of the "Brandes" version. Only the first 1,000 articles of the Wikipedia dataset were used. **(A)** Evolution of the betweenness of the removed edges at each step. **(B)** Evolution of the cluster number (blue) and modularity (orange) at each step. The red dotted line shows the step when we stopped the clustering. **(C)** Representation of obtained clusters when 1250 edges were removed (corresponding to the dotted line in A and B). Main cluster topics as well as some important nodes are indicated (see text and Figure S1).

When looking at the clusters obtained using the "Brandes" implementation of the betweenness computation (Figure 1C and S1), we noticed a dark blue cluster on the left which is gathering articles about Missouri highways and cities, a red cluster in the center which is incoporating articles about university topics, geography and America, a yellow cluster about music and films, a green cluster about Mathematical topics and a light blue one about literature (mostly medieval one). Several general observations derive from the results:

- We can notice that some clusters are poorly connected to each other. For example the Mathematics cluster is not directly connected to Missouri and to Music clusters. This is expected because these topics have little in common.

[2]Though we did not implement this, finding the argmax of the criterion matrix can be done in $\mathcal{O}(n)$ operations instead of n^2 by keeping the argmax outside of the merged cluster in memory. Therefore we addressed the complexity of this more clever approach instead of our exact code.

- It is very interesting to notice a form of "transitivity" within clusters. For example, Missouri roads are linked with towns in Missouri, which themselves are linked with towns in neighboring states and finally with articles about other countries. Another example is the fact that we observe many articles about American football clustered with academia and universities. This is because of the college sports teams system of the USA which makes football and universities relevant together. Therefore we can see how geographic or even conceptual and cultural distances translate into the structure this graph clustering.

- In each cluster, some articles are densely connected to all the other members of the same cluster. This is the case of the article titled "Mathematics" for the green cluster, "List of Missouri roads" for the dark blue cluster or "13th century in Literature" for the light blue cluster. Interestingly, these "key" articles are making bridges between many articles present in each cluster and articles outside the cluster. This is illustrating the hierarchical organization of Wikipedia: one general article is referencing to more specific articles and out-of-field articles mainly refer to this general article.

- The articles at the borders are interesting. The article titled *"United States"* has many connections to the other ones and is classified in the red cluster but has many connections has well with the blue cluster as Missouri is in the United States. In the big red cluster the articles dealing with academic topics (e.g. neuroscience, history etc.) generally share more edges with the "Mathematics" cluster.

Each cluster therefore has a topic unity which is different from other ones showing that our algorithm was able to separate meaningful clusters and that links between Wikipedia articles are not randomly distributed but reflect thematic and topical relationships.

3.2 Markov Clustering Algorithm

Simple and powerful in theory, the MCL algorithm can be numerically unstable and sensitive to floating point precision challenges. Our Python implementation could not exactly match the performance of the official library, but was close. In Figure 2, you can see the results of the test. Convergence is rather fast with less than 30 iterations necessary. Additionally, the algorithm is the fastest of the three covered in this report, taking only 1.6s on average for the given graph. Our implementation gives 21 clusters (5 of which are significant) for the case of the inflation being set to 1.5, which is the one resulting in the highest modularity. The official library outputs 11 graphs.

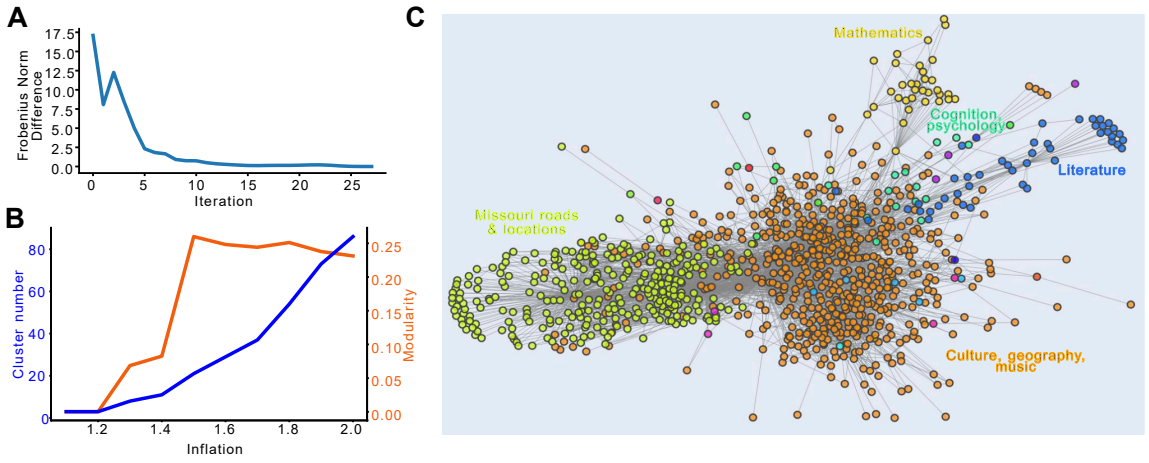

Figure 2: **Clustering with the stochastic flow simulation**, using our implementation of the MCL algorithm. Only a 1,000 articles of the Wikipedia dataset were used. **(A)** Evolution of the convergence of the stochastic Markov matrix at each iteration. **(B)** Number of clusters and global modularity as a function of the initial inflation parameter. **(C)** Representation of the obtained result using the official MCL library.

The summary of the 5 most important clusters that arise with our implementation and sorted by decreasing number of nodes of this method are - Diverse cultural references combined with geography; Missouri transportation and local geography; medieval literature and philosophy; mathematics and mathematical theories and psychology and cognition. The respective themes are displayed on Figure 2C. We can see overlap with the clustering result of the betweenness algorithm, with a few notable differences. Unlike the previous algorithm, it fails to separate music and film nodes from the geography nodes, but rather combines them in one big cluster. Other than that, it similarly distinguishes the Missouri roads and geography separately from mathematics and literature. Additionally, it clusters psychology pages.

3.3 Agglomerative clustering

As we can see on Figure 3C, the clustering neighborhood similarity approach presents approximately the same results as edge betweenness with the same semantic themes appearing. However, borders between clusters are visually less

clearly defined, an intuition which is confirmed when diving deeper in the analysis. Indeed, different clusters overlap on very similar topics and many clusters also spread across different concepts. We can give the example of the red cluster including both Missouri highways and articles about music. This is all the more frustrating since other clusters, the yellow and deep blue ones, are very under-utilized and could have been used to fill one of these purposes. We also observe an extremely sharp exponential decay in the merging criterion, which interestingly this time, doesn't appear to correlate to modularity (even in log-scale).
A feature of this algorithm that we can observe here is that highly central nodes in clusters, often appear to be matched with a different cluster than their neighbor's. We can explain this with the example of Missouri highways. All highway articles are linked to the central *"List of Missouri highways"* article, and thus are very similar in terms of neighborhood. However the individual highway articles are not linked to each other and thus have a very different neighborhood to the central article itself. Once again, whether we want to have central nodes belong to the same cluster as their neighbors or be highlighted in a different group can be a preference choice.

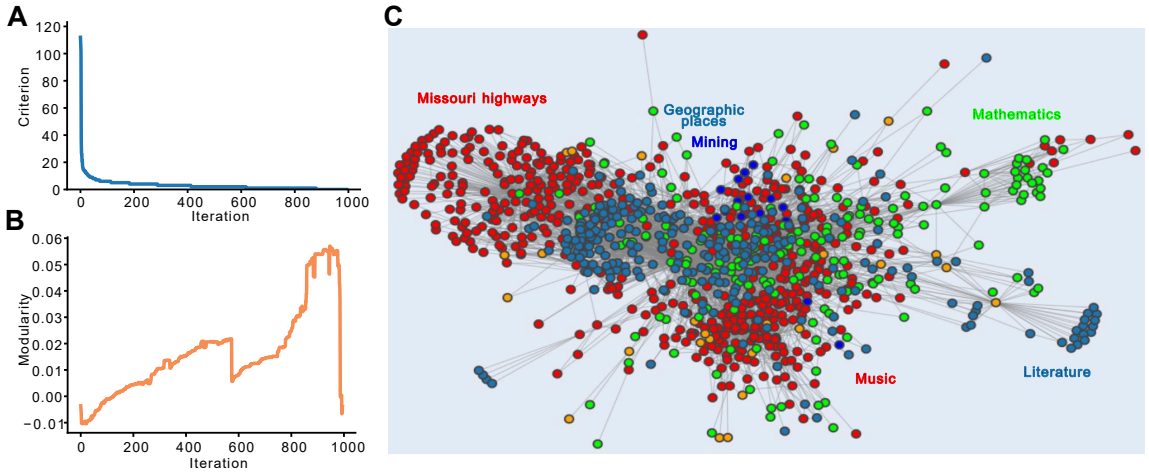

Figure 3: **Clustering with the agglomerative approach**. Only the first 1,000 articles of the wikipedia dataset were used. **(A)** Evolution of the criterion at each step. **(B)** Evolution of the modularity at each step. **(C)** Representation of obtained clusters. We kept the number of clusters to 5 to stay coherent with our other results.

4 Discussion and open pathways

An important drawback we faced when trying to scale up our algorithms to larger portions of the Wikipedia graph was being met with the impracticality of their complexities. Indeed, all of the approaches discussed in this project have a time complexity of $\geq \mathcal{O}(n \cdot m)$ and worse, run $\geq \mathcal{O}(n)$ iterations, a part which therefore cannot be parallelized. This is a significant downside of hierarchical clustering methods which are inherently iterative. Thus, important runtimes in addition to wanting to stay familiar with the support dataset (mainly for result interpretation) meant that we restrained ourselves to a small subset of the huge Wikipedia graph. The biggest gap left unexplored is scaling up our experiments to get a more macroscopic picture of the whole encyclopedia.

As mentioned in the introduction 1, an interesting upside to hierarchical clustering methods is the multi-level results they offer. We would have been very interested in diving deeper into this aspect and analysing results on different scales to see if our cluster encapsulation brought valuable semantic insight. We did as mentioned in 3 observe smaller but coherent topics grouped together into larger clusters and it would have been interesting to make sure this translated into smaller clusters at a reduced scale. For the MCL algorithm, comparing and overlaying different clustering results for different inflation parameter might be an insightful experiment. However, due to the imposed length of this report, as well as time constraints we will have to keep this aspect for another time.
Another path we would have liked to explore is the one of finite mixture models and stochastic block models which are a very interesting way of bringing graphical models to the topic of graph clustering.

We also tested out algorithms on other datasets such as different theme-specific subsets of Wikipedia, urban road networks and the Cora dataset (McCallum 2017) which links researchers together based on paper collaboration.

5 Conclusion

To conclude, we were able to develop three complementary approaches with different pros and cons (see Figure S2 for further details). We succeeded in getting meaningful clusters which reflected well-defined topics for most of them. Our study paves the way to future research on larger Wikipedia datasets.

6 Data availability

All our codes are available in the attached zip file. We developed one notebook per approach (betweenness, stochastic simulation and aggregation).

References

Brandes, Ulrik (June 2001). "A faster algorithm for betweenness centrality". en. In: *The Journal of Mathematical Sociology* 25.2, pp. 163–177. ISSN: 0022-250X, 1545-5874. DOI: 10.1080/0022250X.2001.9990249. URL: http://www.tandfonline.com/doi/abs/10.1080/0022250X.2001.9990249 (visited on 12/07/2024).

Freeman, Linton C. (Mar. 1977). "A Set of Measures of Centrality Based on Betweenness". en. In: *Sociometry* 40.1, p. 35. ISSN: 00380431. DOI: 10.2307/3033543. URL: https://www.jstor.org/stable/3033543?origin=crossref (visited on 12/07/2024).

Girvan, M. and M. E. J. Newman (June 2002). "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12. Publisher: Proceedings of the National Academy of Sciences, pp. 7821–7826. DOI: 10.1073/pnas.122653799. URL: https://www.pnas.org/doi/10.1073/pnas.122653799 (visited on 12/11/2024).

Head, Alison J. and Michael B. Eisenberg (Feb. 2010). "How today's college students use Wikipedia for course-related research". en. In: *First Monday*. ISSN: 1396-0466. DOI: 10.5210/fm.v15i3.2830. URL: https://firstmonday.org/ojs/index.php/fm/article/view/2830 (visited on 12/09/2024).

McCallum, Andrew (July 2017). *Cora Dataset*. en. DOI: 10.3886/E100859V1. URL: https://www.openicpsr.org/openicpsr/project/100859/version/V1/view.

Meila, Marina and Jianbo Shi (2000). "Learning Segmentation by Random Walks". In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press. URL: https://papers.nips.cc/paper_files/paper/2000/hash/069654d5ce089c13f642d19f09a3d1c Abstract.html (visited on 12/17/2024).

Meilă, Marina and Jianbo Shi (Jan. 2001). "A Random Walks View of Spectral Segmentation". en. In: *International Workshop on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, pp. 203–208. URL: https://proceedings.mlr.press/r3/meila01a.html (visited on 12/17/2024).

Newman, M. E. J. (2011). "Networks: An Introduction". In: Oxford University Press, p. 224.

Newman, M. E. J. and M. Girvan (2002). "Mixing patterns and community structure in networks". en. In: vol. 625. arXiv:cond-mat/0210146, pp. 66–87. DOI: 10.1007/978-3-540-44943-0_5. URL: http://arxiv.org/abs/cond-mat/0210146 (visited on 12/07/2024).

Orponen, Pekka and Satu Elisa Schaeffer (2005). "Local Clustering of Large Graphs by Approximate Fiedler Vectors". en. In: *Experimental and Efficient Algorithms*. Ed. by Sotiris E. Nikoletseas. Berlin, Heidelberg: Springer, pp. 524–533. ISBN: 978-3-540-32078-4. DOI: 10.1007/11427186_45.

Orponen, Pekka, Satu Elisa Schaeffer, and Vanesa Avalos Gaytán (Oct. 2008). *Locally computable approximations for spectral clustering and absorption times of random walks*. arXiv:0810.4061 [cs]. DOI: 10.48550/arXiv.0810.4061. URL: http://arxiv.org/abs/0810.4061 (visited on 12/17/2024).

Schaeffer, Satu Elisa (Aug. 2007). "Graph clustering". en. In: *Computer Science Review* 1.1, pp. 27–64. ISSN: 15740137. DOI: 10.1016/j.cosrev.2007.05.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1574013707000020 (visited on 11/10/2024).

Spielman, Daniel A. and Shang-Hua Teng (2004). "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. STOC '04. New York, NY, USA: Association for Computing Machinery, pp. 81–90. ISBN: 978-1-58113-852-8. DOI: 10.1145/1007352.1007372. URL: https://doi.org/10.1145/1007352.1007372 (visited on 12/17/2024).

Van Dongen, Stijn (2000). "Graph clustering by flow simulation". PhD thesis. PhD thesis, University of Utrecht.

Wikipedia (Dec. 2024). *Wikipedia:Size of Wikipedia*. en. Page Version ID: 1261075323. URL: https://en.wikipedia.org/w/index.php?title=Wikipedia:Size_of_Wikipedia&oldid=1261075323 (visited on 12/09/2024).

Xiao, Lu and Nicole Askin (Jan. 2012). "Deliberation in Wikipedia: Rationales in article deletion discussions". en. In: *Proceedings of the American Society for Information Science and Technology* 49.1, pp. 1–4. ISSN: 0044-7870, 1550-8390. DOI: 10.1002/meet.14504901234. URL: https://asistdl.onlinelibrary.wiley.com/doi/10.1002/meet.14504901234 (visited on 12/09/2024).

Additional Figures

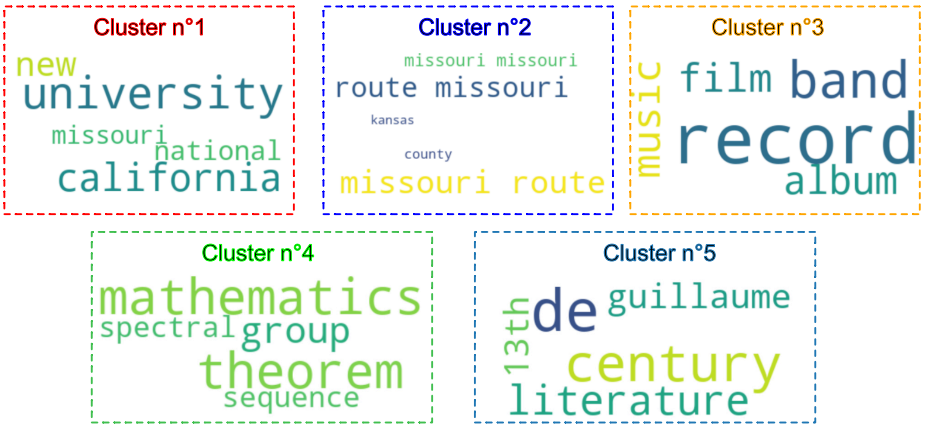

Figure S1: **Most frequent words in the names of nodes for the main five clusters obtained using the betweenness-based approach.** These clusters were identified using our implementation of the "Brandes" algorithm (**corresponding to Figure 1C**). The word clouds were generated with the Python 'wordcloud' library, providing insights into the thematic coherence of each cluster. To validate these results, we manually reviewed the names of many nodes to ensure consistency. In particular, for cluster 1, additional analysis of node names was conducted to refine and more precisely interpret the cluster's thematic significance, as node names were diverse in the cluster. n(cluster 1) = 388; n(cluster 2) = 329; n(cluster 3) = 123; n(cluster 4) = 34; n(cluster 5) = 32; all other clusters have fewer than 5 nodes.

	Top-down cluster division		Bottom-up cluster aggregation
	Betweenness	Stochastic simulation	
Criterion used	Betweenness of removed edges	Idempotence of stochastic matrix	Outside neighborhood
Modularity	0.43	0.26	0.06
Meaningful clusters?	+++	+++	++
Run time	~11 minutes	~1.6 s	~5 s
Complexity	$O(nm^2)$	$O(I \cdot k \cdot n^2)$	$O(nm)$

Figure S2: **Method comparison summary**