Knowledge Graph Analysis For Cornell University Course Catalog

1st Anastasia Sorokina
Connective Media
Cornell Tech
New York, United States
as2764@cornell.edu

2nd Eesha Khanna
Urban Tech
Cornell Tech
New York, United States
ek542@cornell.edu

***Abstract*—In our project, we create and analyze a graphical representation of courses offered at Cornell University. We use web-scraping techniques to obtain course titles, corresponding text descriptions and other attributes. Further, we calculate token embeddings and pairwise similarity scores, and create a network based on the derived similarity measure. Finally, we apply various methods like community detection, topic modeling, and clustering to understand the graph structure of the course catalog and existing relationships between courses. The final output is an interactive graph that acts as a reccommender system for identifying similar courses as well as a graph-based course catalog tool that allows users to select courses using multiple filters.**

I. Introduction

Inspired by the NLP and network embedding topics covered in class, we decided to apply these concepts to a dataset of courses offered at Cornell University, with the goal of constructing a knowledge graph using course titles and descriptions as well as additional attributes such as the number of credits. We use the result of our analysis to build a graph-based visualization of courses that are similar to each other. This can provide an overview of the course catalog and can be used by students as a roster-in-built tool to find courses similar to other courses that they liked.

A literature review of the problem space shows that there have been attempts to work with similar data in a graph context, but most of the applications have been different. For instance, Yang et al. proposed a knowledge graph-based approach in order to learn prerequisite relations among courses and build a universal concept graph which can inform the order in which concepts should be studied or learned [1]. Another interesting application in this domain has been by Aliyu et al in the paper "Development of Knowledge Graph for University Courses Management," which proposes an automated approach based on a knowledge graph to address the problem of manual course allocation [2]. While these two applications are very different from what we are proposing, we found one paper that has a similar goal to ours, but adapts a very different approach. Xu et al. in their paper "Personalized Course Recommendation System Fusing with Knowledge Graph and Collaborative Filtering" propose a recommendation algorithm that combines knowledge graph and collaborative filtering to overcome the shortcomings of regular recommendation systems that overlook semantic relationships between courses [3]. They use the TransE algorithm to calculate embeddings and find relationships using semantic similarity and fusing similarity measures. Our approach is different (as described in the section below), especially for the calculation of embeddings as we plan to use the BERT model to produce them. Moreover, we use cosine similarity as it is known to work well for NLP-related tasks.

II. Problem Formulation

In our project, we aim to build and analyze a graphical representation of courses offered at Cornell University. We start with the assumption that textual information can be represented numerically and a cosine similarity score of two given courses can be calculated using their text attributes such as titles and full course descriptions. By representing courses as nodes, we can draw edges between them based on the similarity of their text descriptions. Using pairwise similarity scores, we can determine a threshold value when we assume that there is a relation (edge) between courses. The resulting edges can further be weighted by the corresponding similarity scores and nodes can be grouped using various techniques like community detection, topic modeling, and clustering. The final graph representation can be used for:

1) Identifying similar courses: we can use various algorithms and network methods to create a graph-based recommender system for Cornell classes.
2) Understanding the structure of the course catalog: we can apply spectral clustering, communities detection algorithms, or alternative methods for detecting groups in the network.

III. Proposed Approach

Our proposed approach consists of four main steps: (i) data collection and cleaning, (ii) knowledge graph generation, (iii) graph analysis, and (iv) visualization. The sections below outline the methodology for each of these steps.

A. Data Collection and Cleaning

We scraped course data including course titles and descriptions from the https://courses.cornell.edu/ Apart from the course titles, the data was fairly unstructured as information

about the course - term, credits, professor, etc. - was all embedded in the description section. Therefore, we cleaned and structured the course descriptions by extracting the course term, credits, and professor, removing frequently occurring phrases and words.

We scraped data for 11,500 Cornell courses from the course catalog. After removing the ones that are not currently offered, had a description and/or other essential fields missing; we were left with 6858 courses. Lastly, we combined the title and the description fields, and removed the course number as well as department code from the titles in order to prepare the data ready for the construction of the graph.

B. Knowledge Graph Generation

We used the course titles and cleaned description fields to calculate embeddings using the BERT (Bidirectional Encoder Representations from Transformers) language model, which is specifically trained to distinguish between sentences semantically [4]. Next, we calculated the cosine similarity measure between each pair of nodes (Eq. 1).

$$\text{cos similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

We adopted an empirical approach to determine the cut-off threshold for two nodes to be considered similar enough that they are connected by an edge. A threshold of zero means that the graph is fully connected, while a threshold of one means that the graph only consists of self-loop edges. We built an interactive representation (using Plotly library) that allowed us to observe the graph's connectivity based on different threshold values. After experimenting with different cut-off values, we selected 0.75 as the threshold as it resulted in desired connectedness (Figure 1). The resulting graph is a thresholded similarity graph, i.e. a graph where nodes are connected by an edge if and only if the similarity between the two nodes is higher than a given threshold.

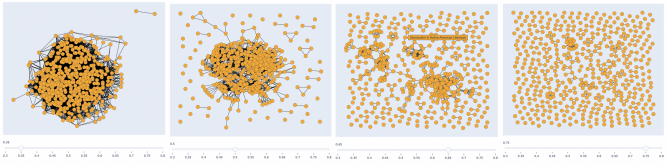

Fig. 1. Graph visualization for determining similarity threshold.

C. Graph Analysis

In order to gain a high level understanding of the structure of the graph, we started by analyzing some basic graph properties like transitivity, clustering coefficient, and degree assortativity. Next, in order to understand the connected components of the graph, we calculated k-cores, specifically 5-cores and 15-cores.

Next, we experimented with three different techniques for grouping similar courses as described below.

1) Community Detection: In order to better understand the catalog structure and find the groups of nodes in the network, we applied two community detection methods – the Clauset-Newman-Moore greedy modularity maximization algorithm as well as the Girvan–Newman algorithm. We chose these algorithms specifically because their logics are somewhat opposite of each other, i.e. the former is additive while the latter is subtractive.

The Clauset-Newman-Moore method starts with communities of size one (one node per community), and repeatedly merges the community pairs that lead to the largest modularity until the marginal increase in modularity becomes insignificant such that the maximum is reached [5]. On the other hand, the Girvan–Newman algorithm detects communities by progressively removing edges from the original network [6]. At every step the algorithm removes the "most valuable" edge, i.e. the edge with the greatest betweenness centrality. We compare and contrast the results obtained from these approaches in the next section.

2) Clustering: We conducted spectral clustering using the K-means algorithm and used the elbow method to find the optimal number of clusters. Given the size of our graph and the potential for similarities getting lost in higher dimensions, spectral clustering is an appropriate technique as it uses the eigenvalues to perform dimensionality reduction before clustering the nodes.

3) Topic Modeling: Finally, we also experiment with topic modeling since it is better suited for NLP problems. BERTopic leverages transformer models and c-TF-IDF to create dense clusters allowing for easily interpretable topics while keeping important words in the topic descriptions [7]. The algorithm roughly consists of three steps: embeddings calculation, dimensionality reduction, and clustering. In the first modeling step vector representations of text documents were obtained through the use of language models (specifically BERT). Semantically similar words have similar values in their vectors. In the second stage, dimensionality reduction was performed using the UMAP technique. Since BERT embeddings are very large, 768 dimensions, they need to be reduced to ensure optimal calculations and high-quality results on the next step. Finally, the reduced vectors were clustered using the HDBSCAN model. HDBSCAN allows for soft cluster assignmens and outliers. Since not all the classes are easily categorized into specific topics, HDBSCAN was a logical model choice.

D. Visualization

Since the intent of our project is to build a recommendation system for courses, we decided to visualize and publish our final graph using an interactive visualization tool called Retina. Retina is a free, open source web application for sharing network visualizations online. We exported the resulting graph, including all node attributes (like term, grading, professor), as well as the groupings created by the various techniques and set up an interactive visualization. This visualization not only allows users to pick a course and find all similar ("neighboring") courses, but also to color the graph using a

particular grouping scheme and apply various filters (term, grading system, department) to view filtered subsets.

IV. Analysis and Numerical Results

A. Knowledge Graph Generation

The thresholded similarity graph for Cornell courses using the 0.75 threshold cut-off, i.e. the Cornell courses knowledge graph is shown below (Figure 2). Visual inspection shows that the majority of the courses are concentrated in the center. However, there are also several courses near the edges that have less than two neighbors.

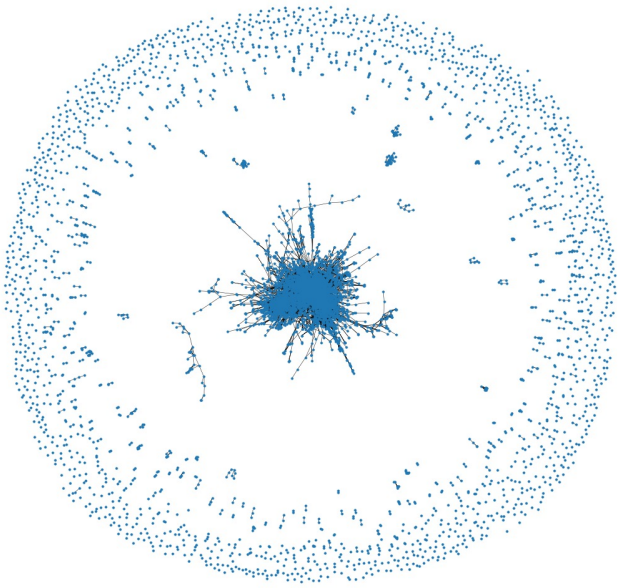

Fig. 2. Cornell courses knowledge graph.

B. Graph Analysis

Our graph has a transitivity score of 0.37 which means that the fraction of all possible triangles present in the graph is 37%. The average clustering coefficient we obtained is 0.33, which means that 33% of the nodes are connected on average. The degree assortativity of the graph, i.e. the similarity of connections in the graph, is 0.44. In analyzing the k-cores for this graph, we found that 40% of the nodes appeared in a 5-core of the graph, and another 12% appeared in a 15-core of the graph. The edges are visualized in Figure 3, with the 5-core edges shown in green and the 15-core edges shown in red.

The results for the various grouping techniques are described below.

C. Community Detection

The Clauset-Newman-Moore method found 1790 communities (Figure 4), while the Garvin-Newman method found 1722 communities (Figure 5).

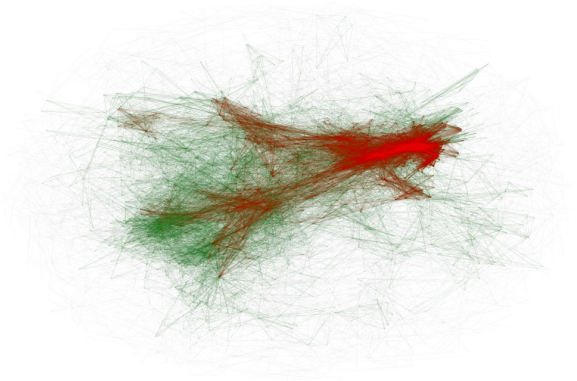

Fig. 3. Cornell courses knowledge graph: 5-core and 15-core edges.

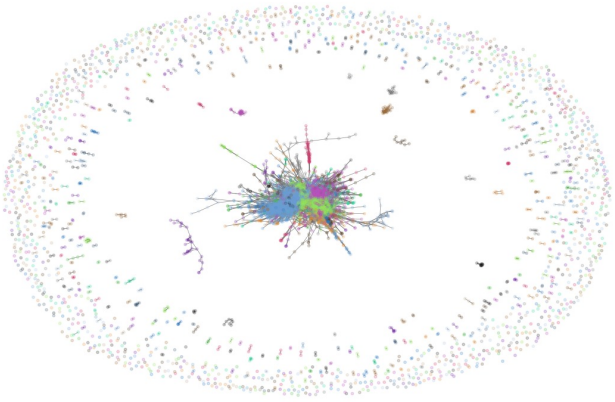

Fig. 4. Communities formed using Clauset-Newman-Moore method.

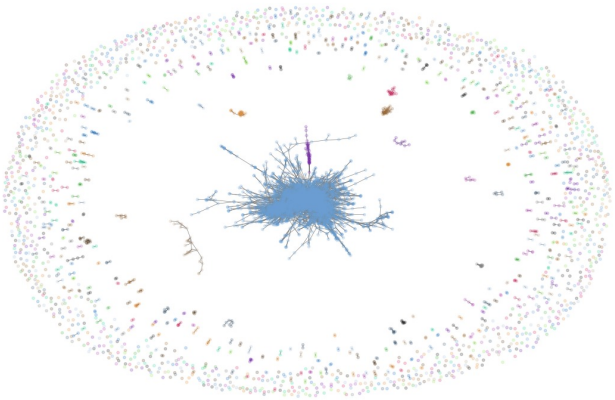

Fig. 5. Communities formed using Garvin-Newman method.

Figure 6 shows an example of a community using the Clauset-Newman-Moore algorithm.

```
frozenset({'CS 5435 - Security and Privacy Concepts in the Wild ',
           'CS 5436 - Privacy in the Digital Age ',
           'CS 5830 - Cryptography ',
           'CS 5831 - Security Protocols and Privacy ',
           'INFO 1200 - Information Ethics, Law, and Policy ',
           'INFO 5365 - Responsible AI ',
           'INFO 7060 - Digital Life Research Seminar ',
           'LAW 6205 - Cyber Enforcement, Regulation and Policy Analysis ',
           'LAW 6473 - Free Speech Law on the Internet ',
           'LAW 6536 - Internet Transactions ',
           'LAW 6568 - Internet Law, Privacy and Security ',
           'LAW 6643 - Law of Autonomous Vehicles ',
           'LAW 6762 - The Prosecution of Cyber-Crime ',
           'LEGAL 5117 - Privacy Law, Regulation, and Business ',
           'TECH 5010 - Values at Play in Digital Technologies ',
           'TECH 5270 - Practical Introduction to Computer Security and Policy '}),
```

Fig. 6. Example of a community formed using the Clauset-Newman-Moore algorithm.

The Girvan–Newman algorithm fails to classify the central graph component and results in mainly single-node communities. The Clauset-Newman-Moore method works significantly better but also results in a very large number of communities, perhaps because of its additive approach. For that reason, we decided to try alternative methods for finding groups.

D. Clustering

For the spectral clustering based groupings, we used the elbow curve shown in Figure 7 to decide the optimal number of clusters. Since the curve has the sharpest edge at k = 300 and the improvements beyond that are relatively marginal, we decided to use 300 as the number of clusters.

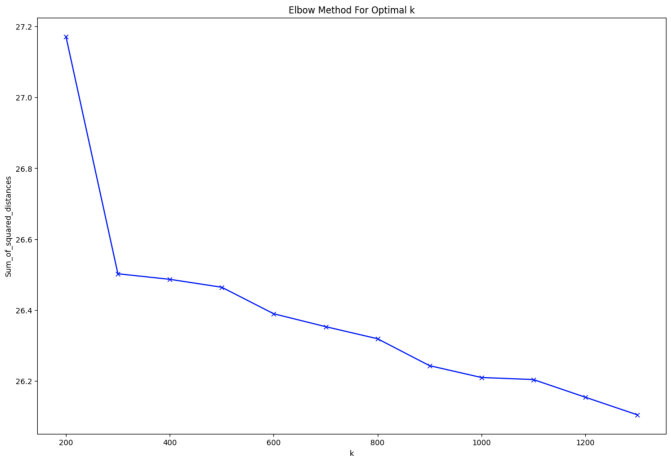

Fig. 7. Elbow curve for determining optimal number of clusters.

The clustered result (Figure 8) was similar to the one obtained using the Girvan–Newman algorithm where the central graph component was not segmented into multiple clusters. Given these findings from the community detection and clustering methods, we decided to experiment with topic modeling.

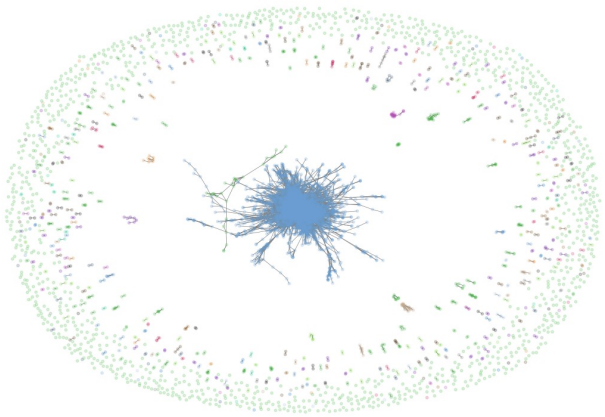

Fig. 8. Clusters formed using spectral clustering method.

E. Topic Modeling

Topic modeling is an unsupervised learning approach, which uncovers latent topics in the documents without specifying the optimal number of such topics. Using this approach, we found 163 groups (or topics) associated with course descriptions (Figure 9). We then assigned names to the groups through the combination of the most relevant keywords specific to the corresponding topics. Figure 10 shows some examples of topics, value counts and topic names.

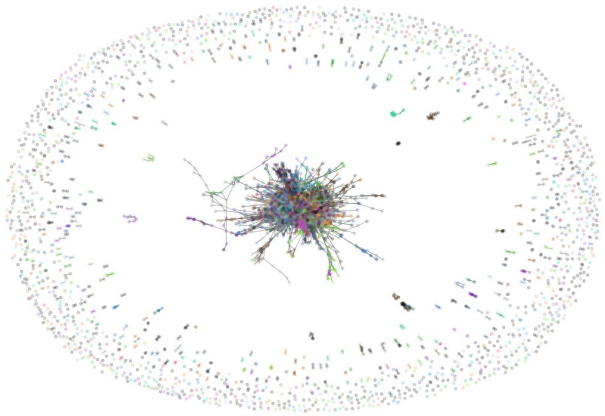

Fig. 9. Topic groupings formed using topic modeling approach.

1	0	162	0_law_clinic_legal_criminal
2	1	86	1_clinical_veterinary_animal_medicine
3	2	84	2_language_hindi_reading_burmese
4	3	77	3_calculus_equations_algebra_differential

Fig. 10. Example of topic groupings formed using topic modeling approach.

Note that courses that do not have enough overlap with other courses are tagged as outliers. About 1300 data points fell in the outlier category under this technique as they did not have substantial overlap with other courses. The smallest topics had 10 courses associated with them. While the community detection and clustering methods allow for single-class communities, the topic modeling technique does not. Instead, it includes those nodes in the outlier category. This is perhaps the reason that the topic modeling method provides better groupings for our use case.

F. Recommendations

Based on the graph we also implemented a basic recommender algorithm that returns neighbors of a given node (nodes are represented as titles) sorted by their weight. The example below shows all the neighbors for the course "AEM 1200 - Introduction to Business Management."

```
recommend_course(list(G_large.nodes())[7])
```

```
INITIAL:  AEM 1200 - Introduction to Business Management
RECS:
('AEM 2200 - Business Management and Organization ', 0.8596717)
```

```
recommend_course(list(G_large.nodes())[68])
```

```
INITIAL:  AEM 1200 - Introduction to Business Management
RECS:
('NBA 5111 - Foundations of Financial Modeling ', 0.82074857)
('ILRHR 6910 - Business Strategy and Finance for Human Resources ', 0.7893483)
('NBAT 6050 - Advanced Topics in Accounting ', 0.77384055)
('NCCY 5000 - Financial Accounting ', 0.7632629)
```

Fig. 11. Example showing neighbors (similar courses) for a business course.

In order to validate these recommendations, we did a qualitative check on the course descriptions for the neighbors of a few courses and found that the recommendations are reasonable, i.e. the courses are similar to each other. These are the exact relationships that we aimed to uncover to make the enrollment period easier for students.

G. Visualization

Our ultimate goal was to represent the graph in such a way that it is easy to understand and navigate through. In our Retina visualization, we used the following attributes as filters:

1) Term (fall/spring/summer/winter)
2) Grading system (letter grades only, S/U etc.)
3) Professor
4) Department
5) Grouping technique (community/cluster/topic)

Users can filter for a particular topic and then add more constraints to narrow down to a subset of classes. Figure 12 shows an example where we filtered for statistics-related courses offered during the fall with letter grades only.

Users can also look at various topics that are represented in the graph. Figure 13 shows an example where we highlighted a few topics to demonstrate the representation on the graph.

Users can also search for specific classes in the upper right corner using the title or the code of a given course. They may

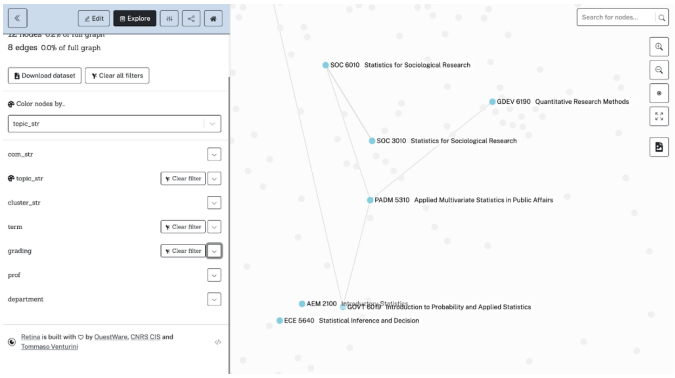

Fig. 12. Retina example - Neighboring/similar courses (statistics).

Fig. 13. Retina example - Highlighted topics.

also highlight a specific node and see what its neighbors are given the active filters (Figure 14).

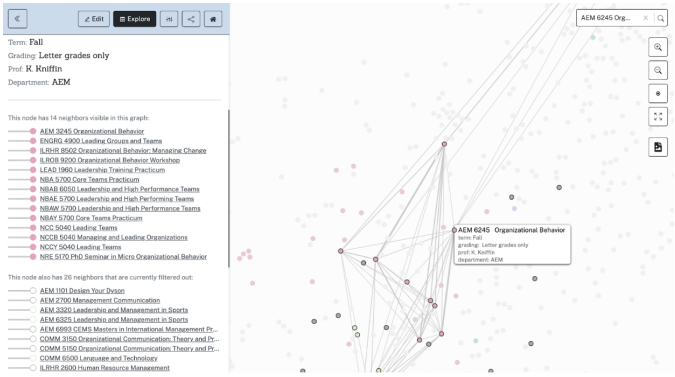

Fig. 14. Retina example - Search function.

V. Conclusion

We were able to develop a robust, replicable workflow for scraping, cleaning, and analyzing the Cornell course catalog as well as for developing an interactive visualization for the generated knowledge graph. The final representation online

acts as a course search and a recommendation tool with filtering and selection capabilities. Students can select a course and find similar courses to that course or just get a top-down view of the course catalog under the various groupings and filters. We found that the topic modeling method was most appropriate for this use case due to its robustness to outliers. However, we decided to keep all three grouping techniques in our final visualization in case users want to play around with them.

If we were to update this visualization as more courses are added to the roster in the future, we can directly apply this approach apart from perhaps tweaking the data cleaning part to incorporate any new idiosyncrasies in the descriptions.

The next steps would include experimenting with other similarity threshold values and ways to determine them. In addition, one could consider different metrics for measuring similarity. We would also like to incorporate the campus location information as another graph attribute. However, it is currently not presented on the Cornell University registrar's website. Another potential improvement would be incorporating pairwise similarity values as an edge attribute in the interactive implementation.

VI. Project Links

- Retina Graph Visualization: Retina
- Code: Scraping, Data Cleaning, Analysis
- Slides: Slides

References

[1] Yang, Y., Liu, H., Carbonell, J., Ma, W. (2015). Concept graph learning from educational data. Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. https://doi.org/10.1145/2684822.2685292

[2] Aliyu, I., A. F. D., K., Aliyu, S. (2020). Development of knowledge graph for university courses management. International Journal of Education and Management Engineering, 10(2), 1–10. https://doi.org/10.5815/ijeme.2020.02.01

[3] Xu, G., Jia, G., Shi, L., Zhang, Z. (2021). Personalized course recommendation system fusing with knowledge graph and collaborative filtering. Computational Intelligence and Neuroscience, 2021, 1–8. https://doi.org/10.1155/2021/9590502

[4] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Google AI Language. https://doi.org/10.48550/arXiv.1810.04805

[5] Clauset, A., Newman, M. E., Moore, C. (2004). Finding community structure in very large networks. Physical Review E, 70(6). https://doi.org/10.1103/physreve.70.066111

[6] Girvan, M. Newman, M. E. J. (2002) Community structure in social and biological networks, Proc. Natl. Acad. Sci. https://doi.org/10.1073/pnas.122653799

[7] Grootendorst, Maarten. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. https://doi.org/10.48550/arXiv.2203.05794