HyperMap: a visual analytics approach for analyzing large collections of unstructured text

Sam Yu-Te Lee*
University of California, Davis

Aryaman Bahukhandi†
University of California, Davis

Kwan-Liu Ma‡
University of California, Davis

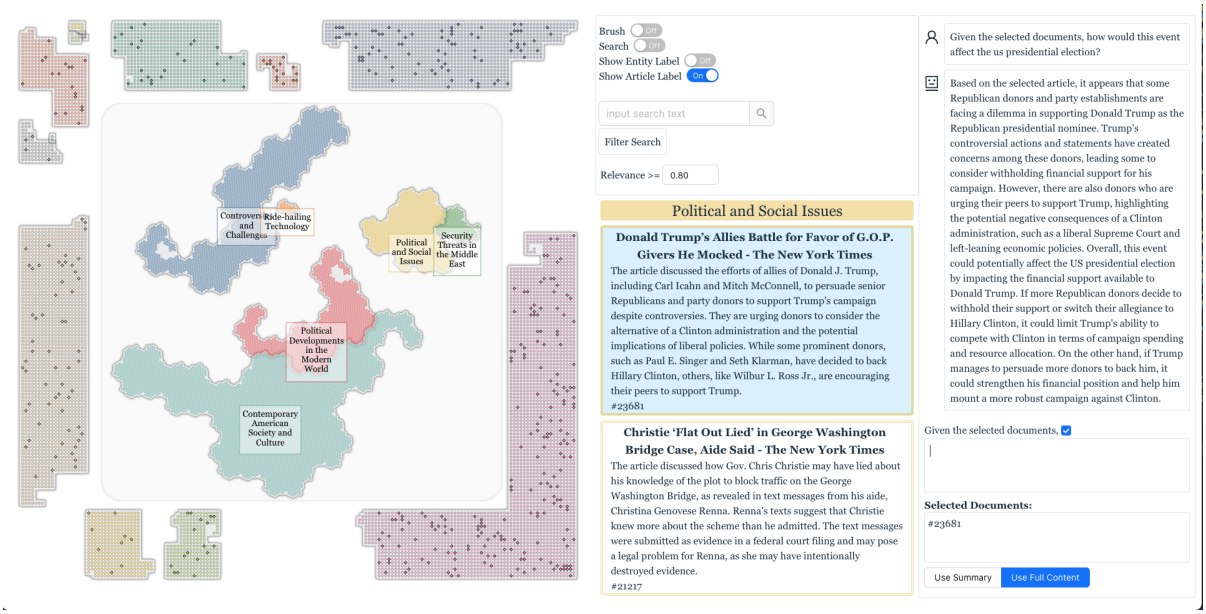


Figure 1: The hypergraph visualization generated by combining generalized Hilbert curves and Gosper curves.

Abstract

Sensemaking on large collections of unstructured text (corpus) is a challenging task that analysts have to perform. Previous works approach this problem either from a topic- or entity-based perspective, but they lack interpretability and trust. In this paper, we propose a visual analytics approach that combines topic- and entity-based techniques seamlessly by modeling the corpus as a hypergraph. The hypergraph is then hierarchically clustered with an agglomerative clustering algorithm by combining semantic and connectivity similarity. We visualize the clustering result to allow analysts to explore and reorganize a corpus for their analysis. The system is designed to foster interpretability and trust by providing rich semantic context for the visualization and by supporting curating interactions. Case studies and a task-based evaluation are used to demonstrate the effectiveness and trustworthiness of the system.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Text data is ubiquitous in the modern world. From news articles and social media posts to scientific publications, the tremendous amount of text data that is produced poses a great challenge to anyone who needs to analyze them. Visual analytics (VA) mitigates this challenge by utilizing visualizations to reduce the cognitive load, but previous approaches have been struggling to deal with the unstructured nature of text data and user preferences. On the one hand, VA systems often need to create structures from unstructured text data before visualizing them. The most common of which is to perform a modeling process on the words of n-grams, such as topic modeling. However, the visualization of these modeling results has been criticized for their lack of interpretability and trustworthiness [4, 10]. On the other hand, users have been reported to prefer reading through the text themselves over depending on their visual forms [10], especially when the task at hand is at high risks [18] The inability to produce an interpretable visualization and user preference over text reading limits the effectiveness of VA systems in analyzing large collections of unstructured text data.

Recent advances in large language models (LLM) present a promising solution to this problem. LLMs have been reported to excel at understanding user intents and generating human-like responses [11]. Instead of relying on abstruse and unfathomable probabilistic models, LLMs can be utilized to directly process the text data at a higher level. For example, topic models utilize statistical tools to generate *topics* from text data, where a *topic* is characterized as a probabilistic distribution spanning a given vocabulary [20]. This transformation from *topics*, a high-level concept that the user seeks to understand, to a *probabilistic distribution*, a low-level concept that mathematical models can operate on, is unnecessary when LLMs are used. Instead, a user can directly ask a LLM: *'What is the topic of this article?'*, and the LLM would give a human-like response, such as *'This article is about the COVID-19 pandemic.'*.

However, LLMs are not without their limitations, especially when it comes to analyzing a specific corpus. First, the token limit of

*e-mail: ytlee@ucdavis.edu
†e-mail: abahukhandi@ucdavis.edu
‡e-mail: klma@ucdavis.edu

LLMs prevents them from processing long or large amounts of documents. As of 2023, most common LLMs are limited to around 4,000 tokens, with certain LLMs reaching up to 32,000 tokens. This limit can be easily exceeded by a single document, let alone a collection of documents. Second, LLMs are reported to make up information that is factually incorrect under certain scenarios, known as *hallucination*. Similarly, LLMs sometimes do not adhere to the provided information, and use parametric knowledge learned during training to answer user questions. The measurement of the ability to adhere to user-provided information is known as the *faithfulness* of LLMs. Finally, a recent study by Sultanum et.al. [19] has shown that prompting could be found too intimidating or unfamiliar to certain users. The use of LLMs in VA systems should mitigate cognitive loads, not the other way around.

Following the above discussion, we propose a novel VA system that allows user to explore, reorganize and analyze large collections of unstructured text data. The system is built upon an LLM-based information extraction pipeline, which is capable of extracting topics and salient entities (or concepts) from a given corpus. The result is then modeled as a hypergraph and hierarchically clustered. The clustering result is visualized as an interactive bipartite graph using space-filling curve layouts, with rich interaction supporting expansion, deletion, and searching. Finally, users can directly use the reorganized corpus to query LLM for detailed analysis.

The contributions of this paper are as follows:

- We propose a novel LLM-based information extraction pipeline that is capable of extracting topics and salient entities from a given corpus.
- We propose a novel bipartite space-filling curve layout that is capable of visualizing clusters in large hypergraphs.
- We propose a novel VA system that allows users to explore, reorganize and analyze large collections of unstructured text data.

2 Related Works

2.1 Analyzing large collections of text

Topic models, entity-based summarization (VA approaches) Literature review tools Document retrieval tools [4, 10] Interpretability and trust of text analysis [16] although a dynamic document categorization that is informed by user query, the representation of categorization and answers is still a sequence of words that lacks Interpretability. Can only answer simple factual questions, experts do not know what they are missing

2.2 LLM for Information Extraction

[11] performance on different tasks: standard IE (poor) vs.Open IE (good). Therefore we use ReFined for EL and chatgpt for others Explainability: Ask the model to explain its prediction: a possible solution to provide explainability for topic assignment and entity linking Calibration: over-confidence Faithfulness: verified to be good

3 Design Rationale

What DRs are needed for analysts to explore and reorganize a corpus for their analytical tasks and why?

3.1 Design Considerations

- **DC1: An overview of the topic structure**
- **DC2: Support for curation through user interaction**
- **DC3: Detailed analysis of the curated result**

4 Methodology

4.1 Modeling

4.1.1 Hypergraph Construction

A hypergraph is a generalization of a graph in which an edge can connect any number of nodes [7]. A hyperedge thus represents a multi-way relationship between nodes. In this paper, we model two types of hypergraphs: article hypergraph and participant hypergraph, where articles and participants are the nodes, respectively. *Participants* are the core components that the article's content discuss [**?**]. For example, in a news article, the participants can be named entities such as people, organizations, or locations. In a research article, the participants can be the concepts or techniques used in the article.

Conducting analysis on article hypergraph and participant hypergraph correspond to topic-based and entity-based analysis, respectively. Following the definition of a hypergraph node, a hyperedge can be used to represent two types of multi-way relationships: (1) A hyperedge between *articles* can be constructed if the articles all mention the same participant. In this case, the hyperedge represents the co-mention of a participant, i.e. a named entity or a concept; (2) A hyperedge between *participants* can be constructed if the participants are mentioned together in the same article. In this case, the hyperedge represents a co-occurrence relationship between participants. Once the two hypergraphs are constructed, they are hierarchically clustered separately. Clusters in the article hypergraph represent topics that are discussed in the dataset. Clusters in the participant hypergraph represent participants (entities or concepts) that frequently co-occurred in an article. For better interpretability of the clustering result, we further assign *tags* for each cluster, which is further explained in Sect. 4.2.4.

Although these two types of hyperedges are constructed differently, we utilize the *dual* of a hypergraph to simplify the construction process. The dual of a hypergraph is simply another hypergraph, where the hyperedges are now nodes and the nodes are now hyperedges. (Add formulas here to explain). Therefore, we first model the articles as nodes and participants as hyperedges to construct the article hypergraph H_A. The participant hypergraph H_P can then be easily constructed by taking the dual of H_A. This construction process also allows us to use the same clustering algorithm on both hypergraphs, which is further explained in Sect. 4.1.2.

4.1.2 Hierarchical Clustering

Common clustering algorithms for graphs consider only graph connectivity. However, for the best interpretability of the clustering result, the node embeddings must be also used in the clustering process. The necessity of incorporating node embeddings is further explained in Sect. 4.2.4. Therefore, this limits our choice of clustering algorithms to attributed node clustering algorithms.

Although there are existing approaches that can cluster attributed nodes on graphs such as EVA [5] and iLouvain [6], they are not designed for hypergraphs. In general, hypergraphs can be clustered in two different ways: (1) Directly operate on the hyperedges by generalizing the graph clustering algorithms. For example, Kamiński et.al. [3] generalizes the modularity metric for graphs to hypergraphs; (2) First transform the hypergraph into a graph and then apply normal graph clustering algorithms [9]. Although the first approach is more intuitive, it is less scalable and hard to incorporate node attributes. Thus, we decided to design our clustering algorithm following the second approach.

Considering all the above, we implemented our hierarchical clustering algorithm by first transforming the hypergraph into a graph following the edge re-weighting process proposed by Kumar et.al. [9], then an agglomerative clustering algorithm [17] is applied on the re-weighted graph. In agglomerative clustering, the key is to define the similarity between nodes and similarity between clusters. We can easily incorporate node attributes into the clustering process by

defining the similarity between nodes and clusters as a combination of attribute similarity S_s and connectivity similarity S_c. Since we're dealing with texts, we refer to the attribute similarity between nodes as semantic similarity.

The semantic similarity $S_s(i,j)$ is the cosine similarity of the embeddings of the two nodes, denoted as v_i. For article nodes, the embeddings are generated using the article content. For participant nodes, the embeddings are generated using a description note of the participant. More details about the embeddings are explained in Sect. 4.2.2. The connectivity similarity S_c is the weighted Topological Overlap (wTO) [8], which is a weighted generalization of the Overlap Coefficient [21], as shown in Equation 1.

$$S_s(i,j) = \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||}, \quad S_c(i,j) = \frac{\sum_{u=1}^{N} w_{i,u} w_{u_j} + w_{i,j}}{\min(k_i, k_j) + 1 - |w_{i,j}|} \quad (1)$$

where $k_i = \sum_{j=1}^{N} |w_{i,j}|$ is the total weight of the edges connected to node i. Finally, a weighting factor α is used to balance the two similarities, as shown in Equation 2.

$$S = \alpha S_s + (1-\alpha) S_c \quad (2)$$

For the similarity between clusters, we used centroid similarity, i.e. the similarity between two clusters is the similarity between the centroids of the two clusters. The algorithm is presented in (TODO: add algorithm pseudocode here)

4.2 Preprocessing

The Methodology can work for any unstructured dataset

4.2.1 Summarization

Chatgpt for summarization

4.2.2 Document Embedding

OpenAI's embedding API

4.2.3 Participant Extraction

Chatgpt for major participant extraction and another model for entity linking

4.2.4 Topic Assignment

Chatgpt to assign topics to each cluster

5 Visualization

5.1 SFC for HyperGraph

Existing hypergraph visualization approaches can be divided into three categories [7]: (1) node-link-based, (2) timeline-based, and (3) matrix-based approaches. We first exclude timeline-based approaches because we do not consider temporal attributes of the data in the system. As for matrix-based approaches, although they are known for their visual scalability, they are not suitable for our system because of their lack of support for visualizing hierarchical clusters and user interactions. In the design guidelines proposed by Abdelaal et.al. [1] in a recent network visualization evaluation study, node-link-based approaches are recommended when: (1) tasks involve the identification of network clusters, and (2) the network is sparse. Condition (1) is fulfilled as explained in Sect. 3, and (2) is guaranteed by the main participant extraction process described in Sect. 4.2.3. Therefore, we decided to use node-link-based approaches for our system.

Although there are a variety of node-link-based approaches for hypergraph visualization, we find the extra-node representation proposed by Ouvrard et.al. [14] most flexible and intuitive. An extra-node representation improves existing clique-expansion of hypergraphs by adding extra nodes to represent hyperedges. The extra-node representation effectively transforms the hypergraph visualization problem into a bipartite graph visualization problem. After that, any node-link-based graph visualization method can be applied.

In our system, we use the space-filling curve (SFC) layout method to layout the extra-node representation of the hypergraph. The SFC layout method uses pre-computed clustering to order nodes in a sequence and then applies a space-filling curve on the node sequence to map it to a two-dimensional screen space [13]. SFC approaches are known for their efficiency and aesthetics in visualizing large graphs [12]. After the preprocessing and modeling stage described in Sect. 4, we have two hypergraphs: the article hypergraph H_A and the participant hypergraph H_P, each having its hierarchical cluster. Combining the extra-node representation and SFC layout, we visualize the article hypergraph H_A and the participant hypergraph H_P as two separate SFCs, as shown in Fig. 1.

Specifically, we divide the layout space into two parts: the peripheral and the center area. For the peripheral area, we concatenate four generalized Hilbert (Gilbert) curves [22]. A Gilbert curve is a generalized version of the Hilbert curve that can traverse any rectangular region in a way similar to the Hilbert Curve. In Fig. 2a, the Gilbert curve starts from the lower left (dark blue) and ends at the lower right (dark red). Through rotation and flipping, the start and end curve points for neighboring Gilbert curves can be concatenated smoothly, as shown in Fig. 2b. The use of concatenated Gilbert curves allows us to fill the peripheral space while having the efficiency and aesthetics of SFC layouts.

The curve to be used for the center area is technically unbounded. In early prototyping, we found that using the same curve as the peripheral region was confusing for the user, as it was hard to distinguish between the peripheral and center areas. We decided to use a simple Gosper curve (Fig. 2c) to layout the nodes for better aesthetics. The resulting visualization looks similar to GosperMap [2], but we did not employ the advanced techniques proposed in GosperMap. The interactions to support the exploration and reorganization of the dataset are the main focus of the system, which are also not limited to any specific curve.

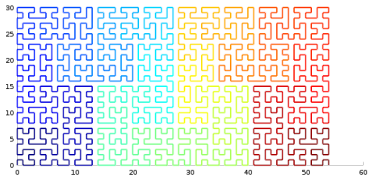
After the curves are generated, we can apply the curves on the node sequences to generate the two-dimensional layout. We chose to put the article hypergraph in the center area because the articles are the main analysis targets for the user. Consequently, the participant hypergraph is put in the peripheral area.

5.2 Improving the readability

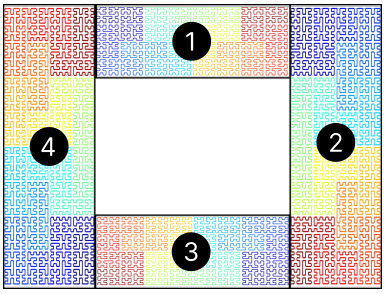
Spacing Strategy We employ a simple spacing strategy.

Concave hull approximation After applying the SFC layout, we use a concave hull algorithm [15] to generate an approximation polygon for each cluster. The polygons are used to generate borders and calculate label positions for the clusters. The concave hull algorithm is generated based on a cluster of points, in our case, the nodes in a cluster. This means that the algorithm can be applied to any curve we used for the SFC layout. This is a desirable property because we are using two different curves in our system, and the center area curve choice is flexible. Using the same algorithm guarantees a unified aesthetic across curves.

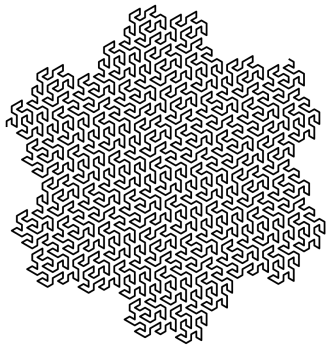
The original concave hull algorithm is designed for approximating points, but we need to approximate circles. Naively we can use the center of the circles as the points for the algorithm, but this would result in a polygon that is too small and crosses the circles on the boundary. To address this issue we use a simple trick: we add extra points to the cluster by extrapolating the original points. For a Gilbert curve, the curve moves perpendicularly, so the resulting polygon would have perpendicular corners. We can therefore add eight extra points around each original point so that the extrapolation forms a three-by-three grid. On the boundary of the cluster, these extra points prevent the concave hull from passing across the original



(a) One Gilbert curve

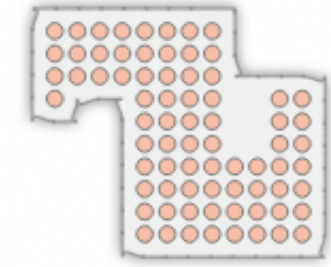


(b) Four concatenated Gilbert curves



(c) An order-4 Gosper curve

Figure 2: (a), (b): Illustration of concatenating Gilbert curves. The color indicates the traverse direction of Gilbert curves: start with dark blue and end with dark red. (c): An example of order-4 Gosper curve used for the center area.



(a)

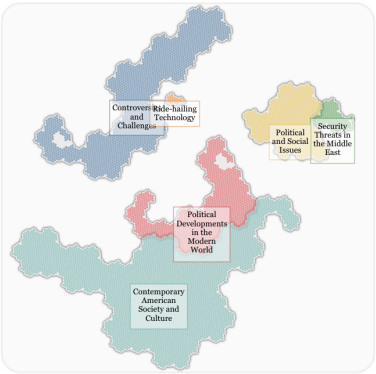
(b)

Figure 3: (a) An example of Gilbert cluster border (b) An example of Gosper cluster borders

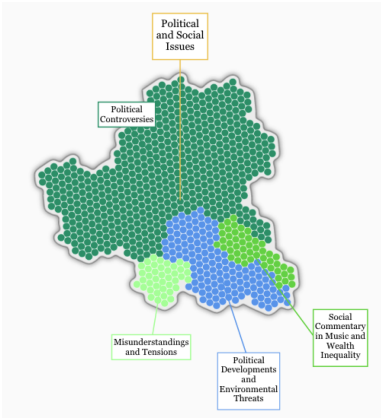
points. Since the concave hull algorithm has an $O(n \log n)$ time complexity, the performance overhead introduced by the extra points is negligible.

Borders The borders are generated by applying a smoothing algorithm on the polygons. For Gosper curves, we use the polygon as control points to generate a cubic basis spline as the border. For Gilbert curves, we use a similar approach but with a cubic Bezier curve. More specifically, for each pair of consecutive points, we use a smoothing factor to interpolate the control point. This results in a sketchy style at the border corners. Two examples are given in Fig. 3.

Labeling Labeling the clusters is essential for users to explore the dataset. We use the topic assignment described in Sect. 4.2.4 to label the clusters. When using SFC layouts, determining the label position automatically is challenging because the shape of the clusters can be irregular. For a cluster, the label position is simply the centroid of the polygon. When a cluster is expanded through user interaction, the sub-clusters within need to be clearly labeled as well. Using the centroid of the sub-cluster as the label position is not a good choice because the label would cause a serious cluttering issue. Therefore, for a sub-cluster, we first calculate the centroid of the sub-cluster, and then we extend the line from the parent cluster centroid to the sub-cluster centroid. Once the intersection point of the extended line and the parent border is found, we extend the line by a fixed amount to avoid any overlapping issues. This results in a radial layout for the sub-cluster labels, as shown in Fig. 4. The generalizability of the concave hull algorithm makes our labeling position calculation applicable to any curve we use for the SFC layout.



(a) Label position of the cluster is the centroid of each cluster.



(b) Labels of the sub-clusters of a cluster. The color indicates different sub-clusters.

Figure 4: (a) Labels (topics) of the article clusters (b) An example of expanded cluster labels

5.3 Design Choices

Why show all the nodes as circles

Why hide the links

6 System Design

6.1 Cluster View

Use SFC hypergraph to show topical structure of the dataset.

6.1.1 Interactions

Click

Expansion

Filtering

Searching

6.2 Article View

6.3 Analysis View

Acknowledgments

The authors wish to thank A, B, and C. This work was supported in part by a grant from XYZ.

References

[1] M. Abdelaal, N. D. Schiele, K. Angerbauer, K. Kurzhals, M. Sedlmair, and D. Weiskopf. Comparative evaluation of bipartite, node-link, and matrix-based network representations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):896–906, 2022.

[2] D. Auber, C. Huet, A. Lambert, B. Renoust, A. Sallaberry, and A. Saulnier. Gospermap: Using a gosper curve for laying out hierarchical data. *IEEE transactions on visualization and computer graphics*, 19(11):1820–1832, 2013.

[3] P. P. . F. T. Bogumił Kamiński. Community detection algorithm using hypergraph modularity. In *Complex Networks & Their Applications IX: Volume 1, Proceedings of the Ninth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2020*, pp. 152–163. Springer, 2021.

[4] J. Chuang, D. Ramage, C. Manning, and J. Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 443–452, 2012.

[5] S. Citraro and G. Rossetti. Eva: Attribute-aware network segmentation. In *Complex Networks and Their Applications VIII: Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019 8*, pp. 141–151. Springer, 2020.

[6] D. Combe, C. Largeron, M. Géry, and E. Egyed-Zsigmond. I-louvain: An attributed graph clustering method. In *Advances in Intelligent Data Analysis XIV: 14th International Symposium, IDA 2015, Saint Etienne. France, October 22-24, 2015. Proceedings 14*, pp. 181–192. Springer, 2015.

[7] M. T. Fischer, A. Frings, D. A. Keim, and D. Seebacher. Towards a survey on static and dynamic hypergraph visualizations. In *2021 IEEE visualization conference (VIS)*, pp. 81–85. IEEE, 2021.

[8] D. M. Gysi, A. Voigt, T. d. M. Fragoso, E. Almaas, and K. Nowick. wto: an r package for computing weighted topological overlap and a consensus network with integrated visualization tool. *BMC bioinformatics*, 19(1):1–16, 2018.

[9] T. Kumar, S. Vaidyanathan, H. Ananthapadmanabhan, S. Parthasarathy, and B. Ravindran. A new measure of modularity in hypergraphs: Theoretical insights and implications for effective clustering. In *Complex Networks and Their Applications VIII: Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019 8*, pp. 286–297. Springer, 2020.

[10] T. Y. Lee, A. Smith, K. Seppi, N. Elmqvist, J. Boyd-Graber, and L. Findlater. The human touch: How non-expert users perceive, interpret, and fix topic models. *International Journal of Human-Computer Studies*, 105:28–42, 2017.

[11] B. Li, G. Fang, Y. Yang, Q. Wang, W. Ye, W. Zhao, and S. Zhang. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. *arXiv preprint arXiv:2304.11633*, 2023.

[12] K.-L. Ma and C. W. Muelder. Large-scale graph visualization and analytics. *Computer*, 46(7):39–46, 2013. doi: 10.1109/MC.2013.242

[13] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, 2008.

[14] X. Ouvrard, J. L. Goff, and S. Marchand-Maillet. Networks of collaborations: Hypergraph modeling and visualisation. *CoRR*, abs/1707.00115, 2017.

[15] J.-S. Park and S.-J. Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information science and engineering*, 28(3):587–600, 2012.

[16] R. Qiu, Y. Tu, Y.-S. Wang, P.-Y. Yen, and H.-W. Shen. Docflow: A visual analytics system for question-based document retrieval and categorization. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[17] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. 2000.

[18] N. Sultanum, F. Naeem, M. Brudno, and F. Chevalier. Chartwalk: Navigating large collections of text notes in electronic health records for clinical chart review. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1244–1254, 2022.

[19] N. Sultanum and A. Srinivasan. Datatales: Investigating the use of large language models for authoring data-driven articles. *arXiv preprint arXiv:2308.04076*, 2023.

[20] I. Vayansky and S. A. Kumar. A review of topic modeling methods. *Information Systems*, 94:101582, 2020.

[21] M. Vijaymeena and K. Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28, 2016.

[22] J. Červený. https://github.com/jakubcerveny/gilbert/commits/master generalized hilbert ("gilbert") space-filling curve for rectangular domains of arbitrary (non-power of two) sizes., 2019.