# Brandeis Capstone Project Report

HAYDEN MCCORMICK

May 2024

## 1 Abstract

This report presents the MMIF Graph Visualizer, a highly interactive cluster-based tool for analyzing, searching, and understanding large sets of MMIF files. NLP modeling techniques including K-Means clustering, topic modeling, summarization, and named entity extraction are used to visually differentiate nodes, and graphical visualizations such as word clouds and timelines are used to filter large search spaces to manageable node groups.

## 2 Introduction

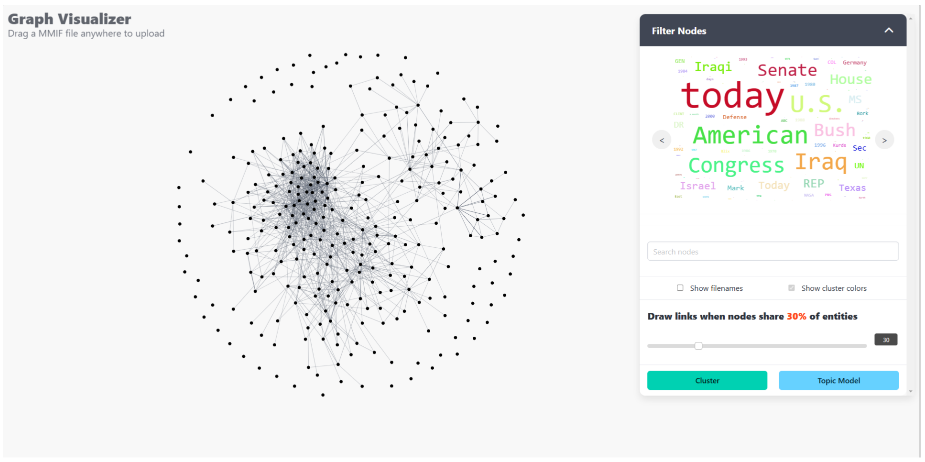
### 2.1 Motivation

The MMIF format allows a set of documents, generally corresponding to a single archive piece, to be directly annotated by an arbitrarily large set of natural language processing and computer vision applications. Although this is an efficient and convenient way for programs to access this metadata, a common complaint with MMIF is that an archivist needs to skim through tens (or even hundreds) of thousands of JSON lines to understand what a file actually represents. From this frustration came the MMIF visualizer, which aims to faithfully present the *context* of a MMIF file by rendering the documents it refers to, and the *content* by rendering several HTML visualizations corresponding to each annotation type (e.g., thumbnails for OCR annotations).

Although this tool makes a significant difference in a user's ability to understand a single MMIF file, another problem remains. These files are generally run in batches, and filenames often refer to the GUID of the source documents, making exploration and search incredibly difficult at the document-set level.

This motivates the concept of an effective visualizer which operates on the collection-level. Specifically, unlike other approaches, the Graph Visualizer aims to leverage the fact that certain documents can be intuitively recognized as more "related" than others – by sharing themes, event mentions, or people, for example. To this end, the project has the following goals:

1. Representing an arbitrarily large set of files spatially, so that relationships and clusters of files can be easily and intuitively identified.



Show filenames

Show cluster colors

Draw links when nodes share 30% of entities

30

Figure 1: Screenshot of Graph Visualizer with 300 nodes

2. Improving ease of search through a body of MMIF files, and to implement search/filtering not just for document-finding, but also for identifying new patterns and connections in a narrowed search space.

3. Adding explainability by representing each node – or sets of nodes – with abstractive document summaries.

## 3 Visualization

The core of the Graph Visualizer's visualization suite is the node document representation with entity links; every filtering, searching, and clustering operation in some way manipulates one or many of the nodes. Each node, when clicked, contains a short abstractive summary of its document content, and edges are established between nodes based on the number of shared entities in their transcript.

The result of this is an interactive representation akin to embedding documents in a shared shape via PCA; natural "clusters" are formed based on common groups of entities. Because of link-based forces in the D3 simulation, more "central" nodes which share the most common entities are generally forced to the center, whereas outliers appear on the outside.

The visualizer also contains two cluster views which apply separate models/algorithms: a K-means algorithm and a BERT-based topic model. Since the entity edges remain intact during clustering, nodes which share many entities with neighboring clusters are pulled towards the center, giving a visual analogue to cluster coherence by node.

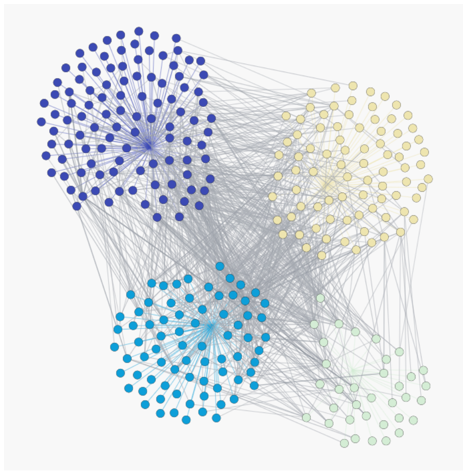


Figure 2: Clustering with entity-linked nodes

## 4 Modeling/Evaluation

### 4.1 Summarization

Although the latest large language models achieve state-of-the-art performance in summarization[**?**], they are often slow if run locally, or costly otherwise. On the other hand, non-attention-based approaches can work very well for extractive text summarization, but struggle with (abstractive) generation[**?**]. A good middle ground is BART, an encoder-decoder transformer inspired by BERT which excels especially at summarization after fine-tuning.

For this project, I experimented with both the base un-tuned BART model and a variant fine-tuned on CNN/Daily mail. To measure this, I compiled a dataset of 2,258 NewsHour transcripts and descriptions, each of which features the headline of the episode. Since the headline essentially serves as a summary of the video's content, it can be used as a baseline to compare the two models.

### 4.2 K-Means Clustering

Another core feature of the Graph Visualizer is the ability to cluster documents in real-time using the K-Means algorithm. Unsupervised methods like K-Means are, in generally, very difficult to evaluate especially for natural language, since any categorization is almost entirely subjective. However, there are a few heuristics that can suggest the performance of a clustering system. One such metric is silhouette score, which is defined as the difference between the average distance of a point to other clusters and the average distance to other nodes within the same cluster, normalized by the maximum of the two:

$$\text{silhouette} = (b - a)/max(a, b)$$

|  | Sillhouette Score |  |
|---|---:|---:|
| Number of total nodes | 100 | 1,000 |
| 20 Clusters, all-mpnet-base-v2 | 0.05739843845 | 0.05073855445 |
| 10 Clusters, all-mpnet-base-v2 | 0.06408067793 | 0.06723831594 |
| 5 Clusters, all-mpnet-base-v2 | 0.09905090928 | 0.05979380012 |
| 10 Clusters, BOW | 0.2737226973 | 0.2676438473 |
| 5 Clusters, BOW | **0.4076516027** | **0.7654857811** |
| 10 Clusters, TF-IDF | 0.08225996796 | 0.03597537923 |
| 5 Clusters, TF-IDF | 0.08777493175 | 0.02421253796 |

Table 1: Results of clutsering with different embedding methods and n clusters.

This has the benefit of accounting for not only the degree of fittedness a node has to its cluster, but also its potential to belong to another. Calculating the average silhouette score of *all* nodes can therefore give a reasonable approximation of clustering "confidence."

To determine the best clustering strategy for this program, it's also important to incorporate the fact that the clustering needs to be robust to different numbers of input nodes – the cluster quality should remain stable independent of how many files a user has uploaded.

The results [1], indicate an improved silhouette score when clustering on Bag of Words and TF-IDF features, rather than on context-embedded transformer representations. Since clustering is performed on summaries rather than entire documents, this is an understandable result; BART-generated summaries generally contain succinct lists of individual events, where context is not preserved nor relevant on a sentence-to-sentence level. This representation also comes with the added benefit of significantly faster performance of BOW compared to mnet-base, and, importantly for us, allows for a cluster representation that is conceptually distinct from the BERT-based topic modelling.

## 4.3 Topic Modeling

Similarly, topic modeling is another clustering feature of the visualization. Sticking with the primarily transformer-based architectures of the other tools, the Graph Visualizer performs topic modeling using BERTopic. BERTopic, unlike many other topic modelling strategies, uses a pipeline of steps including SBERT embedding, clustering, and TF-IDF weighting.

As a further step, BERTopic allows for zero-shot topic modeling, which allows a user to employ their own domain knowledge of a dataset to define custom topics. Armed with these embedded manual topics, BERTopic trains two models on the data – one with manually defined topics and one without – and merges them into a single model. Especially for the axis topic representation, this can be extremely helpful for visualizing documents' relatedness to a given subject, with the caveat that a topic label is removed from the model if no documents of that topiccan be found.

Arguably, topic models are even more difficult to evaluate than general clustering algorithms, as documents with shared topics may not neatly lie in a similar embedding space. One dimension that topic models are commonly evaluated in is (UMass) coherence score, which measures the similarity of the top words in a given topic cluster to measure how semantically "coherent" the clusters are.

$$C_{UMass}(w_i, w_j) = log\frac{D(w_i,w_j)+1}{D(w_i)}$$

This is a good start and gives us one of the only possible quantitative measures of the topic model's performance, but it fails to capture how well the model actually categorizes data points into topics – it is possible to have a model with extremely well-defined topic categories that still bins its data in a naive or incorrect way. This is doubly the case in relation to the Graph Visualizer: most news/archive videos contain a multitude of topics, and it is actually detrimental to the axis visualization if a model is overconfident in assigning cluster scores. For this visualization, there needs to be a balance between well-defined categories and nuanced probability distribution, and evaluation of this can generally only be carried out through visual judgements.

## 5 Limitations

As a dynamic physics simulation, one of the main shortcomings of the program is efficiency. D3's force-directed graph layout is fairly efficient at quickly rendering a small set of nodes, it is very computationally expensive to load a large set of nodes and edges into memory (although once they are loaded by the client, performance issues are rare). This can lead to irritating load times when more files are uploaded, and therefore it is recommended to not load more than 300 nodes into the simulation at once.

Another computational hurdle comes with the many different types of modeling available. The program performs summarization at upload-time, since that is the most computationally expensive operation in the visualizer. However, because of this, file upload is slow, and should generally be done in bulk rather than in real-time.