DEMONSTRATION

TINK: Text Information Navigation Kit

DEAN E ALVAREZ, University of Illinois Urbana-Champaign, Urbana, IL, United States

CHENGXIANG ZHAI, University of Illinois Urbana-Champaign, Urbana, IL, United States

TINK: Text Information Navigation Kit

Dean E. Alvarez
University of Illinois Urbana-Champaign
Urbana, USA
deana3@illinois.edu

ChengXiang Zhai
University of Illinois Urbana-Champaign
Urbana, USA
czhai@illinois.edu

Abstract

In many cases it is desirable to navigate an arbitrary text collection. While current technologies allow for fast and efficient text collection search, comparable technologies do not yet exist for navigation. Our prior work has proposed Hypergraph of Text (HoT) as a structure that could enable such navigation, but there is not yet a general system that allows users to interact with such a structure. To address this challenge, we introduce Text Information Navigation Kit (TINK) a novel tool-kit that allows users to turn any collection of plain text documents into a navigable collection. We discuss the modular and extensible two-part structure of TINK. In this demonstration paper we discuss this novel system, its strengths and its weaknesses. The code for TINK can be found in our GitHub repo here: https://github.com/TIMAN-group/TINK

CCS Concepts

• **Information systems → Information systems applications**; *Document collection models.*

Keywords

Text Navigation; Hypergraph of Text; Browsing; Text Information Navigation Kit

ACM Reference Format:
Dean E. Alvarez and ChengXiang Zhai. 2025. TINK: Text Information Navigation Kit. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3726302.3730141

1 Introduction

The internet is one of the most ubiquitous and useful text collections in the modern day. One key to the utility of the internet is that hyperlinks make it so that one can navigate from page to page easily. Despite text collections being incredibly common in both personal and professional contexts (e.g. personal knowledge bases, digital libraries, corporate documentation) a relatively small amount of text collections are navigable in the way the web is. While the advances in search technologies have made it possible to quickly locate specific information within a text collection, there are many cases in which effective navigation is more desirable. One such case is when a user has a vocabulary gap and accordingly cannot formulate an effective query [3]. Another compelling use-case for navigation is when the user has an exploratory information need and does not know well the particular documents available in a dataset. An example of this could be conference presentations; if conference proceedings were navigable, there would be more room for finding works one did not know they were interested in.

While some text collections, notably the Web, support flexible navigation, the links are typically created manually and cannot fully capture all the interesting links in a corpus. Navigation, unlike search, requires understanding detailed relationships between documents, making it a difficult task. This leads to an interesting research question: How can we automatically generate semantic links at scale for arbitrary text collections? A more general point is that, depending on the information need, a user generally would benefit from interacting with a sequence of information best matching the user's need instead of following the natural linear structure of an individual document. Note that some of the most common formats for text data on the internet, PDF files, e-books, Word documents, are still primarily based on a linear system as a holdover from when text data was only present in physical artifacts such as books or magazines. Thus the general challenge is to automatically create a meaningful semantic structure for any collection of text data.

To address this challenge, we proposed Hypergraph of Text (HoT) as a general formal structure for organizing text collections and enabling navigation [2]. The structure shows promise in capturing complex relationships between text, potentially allowing for a new and rich way to navigate document collections. Despite its theoretical promise, however, there has been a lack of accessible tools for users to leverage a HoT-based navigation approach to turn an arbitrary collection into a navigable collection.

In order to mitigate these issues, we propose the Text Information Navigation Kit (TINK). TINK is a novel tool-kit designed to be a modular and extensible way to enable users to navigate text collections based on a HoT structure. In this demonstration paper we discuss TINK, its design, its strengths, its weaknesses, and exciting opportunities for future work.

2 Text Information Navigation Kit

The primary goal of TINK is to enable users to make an arbitrary text collection navigable with minimal effort while still granting sophisticated users freedom to customize and extend the base system. In service of this goal we maintain a core design philosophy which will be detailed in section 3.

At a high level, TINK can be divided into two layers: the data layer and the interaction layer. The basic flow of using TINK begins with the user building a HoT using the data layer. Once this is done, the user then specifies a config file. This config file is used by the interaction layer to dynamically create a web interface. In its most basic form, this web interface allows users to view documents (nodes of HoT) and click on keywords (a hyperedge of HoT), which will navigate the users to a page where they can see the other documents that are related to this keyword (other nodes connected to the hyperedge). If the user so specifies in the config file, the web interface may also contain various other pages providing for additional views of the text information organized with HoT. This process is summarized in figure 1.

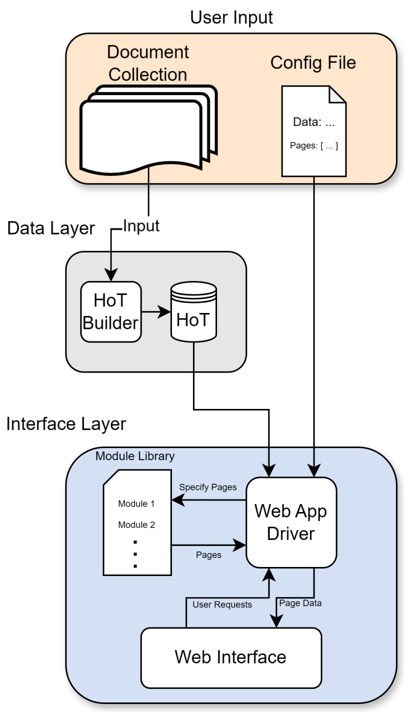


Figure 1: the flow of data in and structure of TINK

Below we provide more details on each of the two layers, how they enable our goal, and high level implementation details.

2.1 Data Layer: HoT

In TINK, text data is stored and organized as a Hypergraph of Text. The Hypergraph of Text is essentially a hypergraph (i.e. a graph where the edges can connect more than two nodes) where both the nodes and hyperedges are annotated by text. The formal definition is as follows [2]:

Definition 2.1 (Hypergraph of Text).
A Hypergraph of Text is a hypergraph $H = (V, E, X_V, X_E)$ where

(1) V is a set of nodes
(2) E is a set of hyperedges
(3) $X_V = \{x_v\}_{v \in V}$ is a set of texts such that $\forall x_v \in X_V, x \in Z^{L_v}$
(4) $X_E = \{x_e\}_{e \in E}$ is a set of texts such that $\forall x_e \in X_E, x \in Z^{L_e}$

Where Z is the vocabulary and L_v and L_e are the length of the text sequence for node v and hyperedge e respectively.

In particular, we treat each document in the document collection as node in the HoT [1]. Hyperedges are used to connect these nodes by topic. In particular, we require that each hyperedge be a topic that connects two or more nodes. The HoT layer not only includes the underlying implementation for HoT but also the tools for constructing a HoT. We find HoT to be the right structure for our goals as while hypergraphs provide an easy structure for navigation, the generality of hypergraphs gives sophisticated users a great deal of flexibility in enabling new views and analyses of a text collection.

[1]Note that in general, a node of HoT could denote any text information such as a passage inside a document.

2.1.1 HoT Implementation. The basic implementation of HoT provided as part of TINK is perhaps best conceptualized as an abstract class for future implementations of HoT. While our pure python implementation of HoT and storage with XML files is suitable for medium sized text collections, this implementation could be a bottleneck for very large text collections. Moreover, there are ostensibly many cases in which one cannot store data locally and would thus want an implementation that can interface with the cloud.

As for HoT construction, it turns out that this problem can be quite complex. How to best construct a HoT is outside the scope of this demonstration paper. For the purpose of this system, however, we provide a basic approach. In particular, we iterate through each document and use LLMs to extract. These topics are then treated as the hyperedges. See figure 2 for pseudocode of this process.

```
Create_HoT(Dataset)
    HoT
    hyperedges = {}
    for doc in Dataset:
        node = HoT.add_node(doc)
        topics = extract_topics(doc)
        for topic in topics:
            hyperedges[topic].add(node)
    for topic, node_set in hyperedges:
        HoT.add_hyperedge(topic, node_set)
    return HoT
```

Figure 2: Pseudocode for constructing a HoT

2.2 Interaction Layer

Since the primary goal of the TINK is to enable navigation, we need something to sit on top of the HoT layer which enables a user to interact with the underlying structure and to use its hyperedges to navigate between nodes. This is the job of the interaction layer.

As a toolkit, the goal of TINK at this layer is to empower users with navigational tools. As such, it is important that this layer is designed as to allow the incorporation of multiple tools. In our case, these tools are the core navigational functionality and the additional pages from the module library that a user can optionally specify.

2.2.1 Interaction Layer Implementation. We implement the interaction layer in Python using Flask 3.1. By using Python, we enable sophisticated users to use all of their favorite AI/ML tools (e.g. PyTorch, SciKit-Learn, and huggingface) to enhance TINK. The Flask app is fairly standard and contains three base components: index, nodes, and hyperedges. Each node in the hypergraph has its own page displaying both its text and its hyperedges. Hyperedges are displayed on the side and in-line as links where appropriate. Hyperedges also have their own pages listing the topic they represent

```
{
    "data_route": "data/HoT.xml",

    "pages": [
      {
        "route": "/analysis/stats",
        "template": "stats.html",
        "func": "stats",
        "title": "Stats"
      },
      {
        "route": "/analysis/chart",
        "template": "he_chart.html",
        "func": "chart",
        "title": "Hyperedge Chart"
      }
    ]
  }
```

Figure 3: An example config file defining a navigation environment with additional pages for displaying stats and a chart.

and every node that is a member of the set defining the hyperedge. Finally, the index page is the last base component featuring a list of every node in the hypergraph and a search bar. Since our primary focus is navigation, the search is included for convenience and is implemented as a simple key-word matching algorithm. How to best integrate search and navigation is an interesting future line of work.

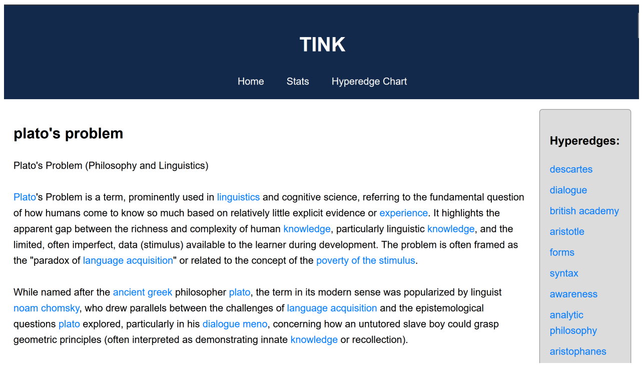
In addition to these base pages, our flask app also supports the user to define additional pages as part of a config file passed as input. An example of what this config file may look like can be seen in figure 3. Essentially, upon start-up, the app reads the page list and fetches the display code from a library of pages/modules. It then adds these routes to the webpage and renders them dynamically. The core idea here is that these config files allow users to customize the navigation experience by adding additional pages without any additional coding. If the user so desires, they might also add code to the module library and use the config file to enhance navigation with their custom pages.

3 Design Philosophy

The TINK design philosophy has two primary virtues: modularity and extensibility. In the following sections we describe how TINK is designed to enhance both its modularity and its extensibility.

3.1 Modularity

There are two key ways in which TINK is modular. The first way is that the underlying implementation of HoT can be swapped out so long as the implementation maintains the same class interface. This allows users with different needs the ability to be able to swap in different HoT implementations according to their needs. For instance, it is easy to imagine if a user has a particularly large dataset they might want a highly performant HoT implementation. Alternatively if the data lives on some sort of remote server (perhaps due to privacy reasons), a user might want a HoT implementation that can interface with some cloud API that they are using. The



plato's problem

Plato's Problem (Philosophy and Linguistics)

Plato's Problem is a term, prominently used in linguistics and cognitive science, referring to the fundamental question of how humans come to know so much based on relatively little explicit evidence or experience. It highlights the apparent gap between the richness and complexity of human knowledge, particularly linguistic knowledge, and the limited, often imperfect, data (stimulus) available to the learner during development. The problem is often framed as the "paradox of language acquisition" or related to the concept of the poverty of the stimulus.

While named after the ancient greek philosopher plato, the term in its modern sense was popularized by linguist noam chomsky, who drew parallels between the challenges of language acquisition and the epistemological questions plato explored, particularly in his dialogue meno, concerning how an untutored slave boy could grasp geometric principles (often interpreted as demonstrating innate knowledge or recollection).

Hyperedges:

- descartes
- dialogue
- british academy
- aristotle
- forms
- syntax
- awareness
- analytic philosophy
- aristophanes

Figure 4: View of a node in TINK

modular design of the TINK would allow a user to drop in a new implementation fairly easily.

The second axis of TINK's modularity is its non-document pages being defined within a config file. Each non-document page is considered a module which is optionally included or excluded.

3.2 Extensibility

Many of the aspects of TINK's modularity also enable it to be extensible. In particular, our use of a configuration-driven architecture allows users to specify additional pages dynamically. These user-defined pages are injected into the navigation and routing system at run time allowing the system to accommodate new user-defined navigational and analytical tools without modifying the core code base.

Additionally, our base page template integrates D3 [4] into its header. By doing so, we are able to give the users an additional tool to create new functionality by integrating both Python and JavaScript code. This ensures that if a user wants to define custom visualization-based navigation functionality, they can use both python and JavaScript tools and packages and that they can do so without reworking existing Flask logic.

4 Demonstration

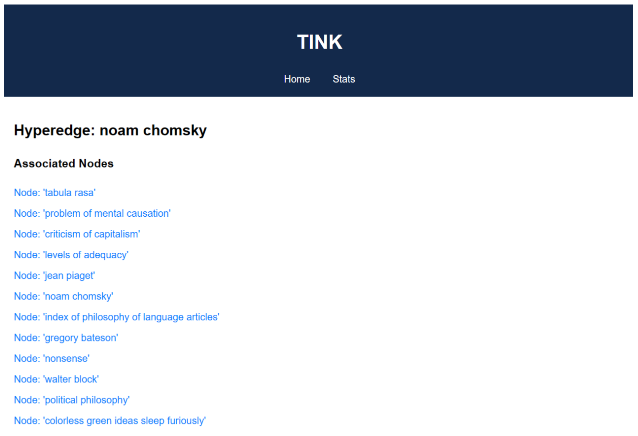
In this section we will detail how the system looks and works in practice. We will begin by discussing the primary use case, navigation. We will then discuss an example navigable analysis page.

4.1 Navigation

The core experience of using TINK to navigate is controlled by the node and hyperedge pages. These pages can be seen in figures 4 and 5 respectively. When viewing a node, the user can see the document text and has the ability to navigate to any of the hyperedges which the node is a member of. These links can show up in two places: the side bar and inline with the text. All links are displayed on the sidebar while only those that could be directly matched to text are shown in line. Hyperedge pages simply list every node that is a member of the hyperedge and the text description of the hyperedge.

4.2 Navigable Analysis

In addition to the primary mode of navigation, we also demonstrate an example of navigable analysis. We focus our efforts on



Hyperedge: noam chomsky

Associated Nodes

Node: 'tabula rasa'
Node: 'problem of mental causation'
Node: 'criticism of capitalism'
Node: 'levels of adequacy'
Node: 'jean piaget'
Node: 'noam chomsky'
Node: 'index of philosophy of language articles'
Node: 'gregory bateson'
Node: 'nonsense'
Node: 'walter block'
Node: 'political philosophy'
Node: 'colorless green ideas sleep furiously'

Figure 5: View of a hyperedge in TINK

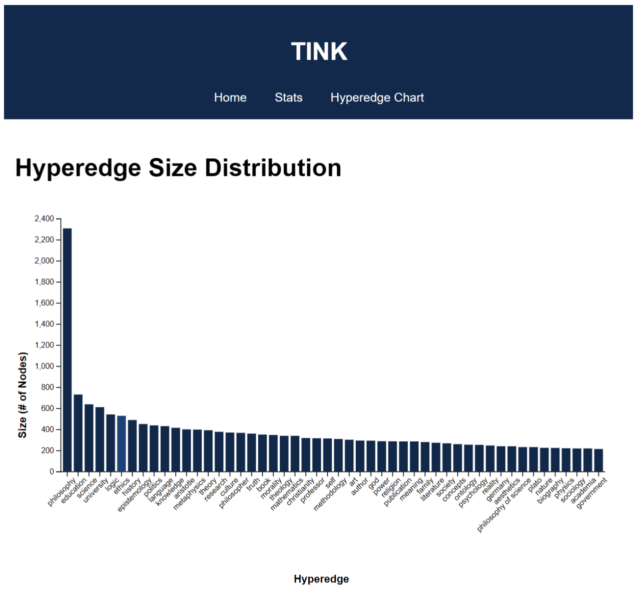


Figure 6: A page showing the distribution of hyper edge size in the HoT. The chart is navigable in that clicking on one of the bars brings the user to the hyperedge page for that bar.

the most simple example of this concept: a bar chart that links to the data it describes. This can be seen in figure 6. Our goal with this demonstration is to show the extensibility of our framework (e.g., the bar chart can be easily restricted to any interesting subset of documents) and how it enables new and interesting modes of navigation and analysis that are only possible with HoT structure.

5 Related Work

Browsing has been a topic of interest for many years in the information retrieval community [5, 7, 11]. While there have been frameworks proposed for browsing large text collections [6, 8], to the best of our knowledge, there are no practical tools or systems that can make an arbitrary text collection navigable. There are a few closely related concepts: **Semantic Organization:** Much work has been done on organizing text data with respect to its meaning, e.g. topic modeling[13], and text clustering [1]. Some methods, such as knowledge graphs [15] and heterogeneous information networks [10], focus on graph structure. While many of these works create interesting and meaningful structure, they haven't been leveraged for the purpose of navigation.

Beyond-hyperlinks browsing: Several strategies have been proposed to enhance navigation on the Web by going beyond the hyperlinks, including using search logs to create a topic map to facilitate navigation in the document space [14], using server logs to generate potentially useful new hyperlinks to link more pages together [9], and augmenting hyperlinks [12]. Compared with the existing work in this line our approach is much more general, applying to any arbitrary text collection not just web. Also, our approach does not require availability of log data and thus are also complementary with previous approaches (they can be potentially combined).

6 Conclusion and Future Work

The demonstration paper introduces TINK as a tool kit that enables users to navigate an arbitrary text collection. TINK aims to provide both casual and sophisticated users the tools to be able to transform static text collections into a rich and navigable format.

TINK's two layer architecture provides a modular and extensible foundation for text navigation while maintaining enough simplicity to enable low start-up time for new users. By leveraging HoT as our core data structure, we enable the framework to capture complex multi-way relationships between documents while maintaining an intuitive navigation flow.

By providing an open source framework for text navigation, TINK not only addresses a current gap in text collection management but also lays the foundation for future work studying algorithms for constructing an effective HoT to enable users to best navigate text data.

There are several limitations of TINK, addressing which leads to interesting lines of future work: One limitation of TINK is its scalability. While the current method is suitable for medium sized datasets, it would likely struggle with large datasets and would be unsuitable for web-scale data. How to best represent and store our underlying Hypergraph based data model is an interesting line of future work. Another limitation of TINK is that it needs to be run locally. While this is suitable, even a boon, for sophisticated users, there are likely many people from non-technical background that would be interested in navigating a text collection. For these users, TINK would be more useful if it had some sort of cloud platform instead of just a local tool. A final limitation of TINK is that after the initial building phase, the underlying HoT is fixed. There may be scenarios in which it would be useful to add new connections or remove superfluous connections to enhance navigation. How to optimize the structure of HoT after its initial construction is an interesting line of future work.

7 Acknowledgments

This work is supported in part by the National Science Foundation and the Institute of Education Sciences under Grant DRL-2229612, and by IIDAI and the SRI Program at UIUC.

References

[1] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. *Mining text data* (2012), 77–128.

[2] Dean E. Alvarez and ChengXiang Zhai. 2024. Hypergraph of Text: a Mathematical Structure for Organizing and Analyzing Big Text Data. In *2024 IEEE International Conference on Big Data (BigData)*. 8605–8607. doi:10.1109/BigData62323.2024.10824995

[3] Nicholas J Belkin. 1980. Anomalous states of knowledge as a basis for information retrieval. *Canadian journal of information science* 5, 1 (1980), 133–143.

[4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). http://vis.stanford.edu/papers/d3

[5] Claudio Carpineto and Giovanni Romano. 1996. A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. *Machine Learning* 24, 2 (01 Aug 1996), 95–122. doi:10.1023/A:1018050230279

[6] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 2017. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. *SIGIR Forum* 51, 2 (Aug. 2017), 148–159. doi:10.1145/3130348.3130362

[7] Xuemei Gong, Weimao Ke, and Ritu Khare. 2012. Studying scatter/gather browsing for web search. *Proceedings of the American Society for Information Science and Technology* 49, 1 (2012), 1–4.

[8] Marti A. Hearst and Chandu Karadi. 1997. Searching and browsing text collections with large category hierarchies. In *CHI '97 Extended Abstracts on Human Factors in Computing Systems* (Atlanta, Georgia) *(CHI EA '97)*. Association for Computing Machinery, New York, NY, USA, 301–302. doi:10.1145/1120212.1120404

[9] Ashwin Paranjape, Robert West, Leila Zia, and Jure Leskovec. 2016. Improving Website Hyperlink Structure Using Server Logs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (San Francisco, California, USA) *(WSDM '16)*. Association for Computing Machinery, New York, NY, USA, 615–624. doi:10.1145/2835776.2835832

[10] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 17–37.

[11] R.H. Thompson and W.B. Croft. 1989. Support for browsing in an intelligent text retrieval system. *International Journal of Man-Machine Studies* 30, 6 (1989), 639–668. doi:10.1016/S0020-7373(89)80014-8

[12] Praveen Vaddadi and KV Sreerama Murthy. 2011. Automatic augmentation of links in web browsing. *International Journal of Information and Communication Technology* 3, 1 (2011), 93–99.

[13] Ike Vayansky and Sathish AP Kumar. 2020. A review of topic modeling methods. *Information Systems* 94 (2020), 101582.

[14] Xuanhui Wang, Bin Tan, Azadeh Shakery, and Chengxiang Zhai. 2009. Beyond hyperlinks: organizing information footprints in search logs to support effective browsing. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 1237–1246.

[15] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *Comput. Surveys* 56, 4 (2023), 1–62.