CS510: Subgraph Retrieval on Text Augmented Graphs

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

ABSTRACT

In this project, we formally introduce the Navigation Text Augmented Graph (Nav-TAG), a novel framework designed to enhance the exploration of semantic links between documents. We also propose a two-step solution to solving our proposed graph retrieval problem. The empirical portion of this project focuses on the second of the two sub-problems. By leveraging the Nav-TAG, we demonstrate the effectiveness of our proposed method for TAG-Subgraph retrieval and by extension for the Query Aligned SLDC Graph problem. Finally, we evaluate the weaknesses of our proposed method and propose future directions of inquiry based on these weaknesses.

ACM Reference Format:
Dean Alvarez. 2024. CS510: Subgraph Retrieval on Text Augmented Graphs. In *Proceedings of (CS510'24).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

It is no secret that we live in a world with more text data than ever. However, despite the incredible utility of tools such as search engines, or even the more recent popularity of large language model integrated search tools such as Co-pilot or Gemini, exploring the connections between documents remains something that is difficult. The reason for this is two-fold. The first reason is that many sets of documents are unstructured. Naturally, having no explicit connections between documents means that the user holds the burden of uncovering the implicit semantic links between documents. The second reason is that even when we have sets of documents that are structured, for instance scientific paper citation networks or wiki's such as Wikipedia, it is not the case that all possible links are enumerated. In other words, it can be the case that two scientific papers have portions that are highly related to each other, but that the two papers might not cite each other. This incompleteness means that potentially value links might be left out, once again making it more difficult for a user to explore the document set.

In this work, we attempt to provide a framework for alleviating these issues: the Navigation Text Augmented Graph (Nav-TAG). We start by discussing related works, then move on to define the Text Augmented Graph (TAG) and the Navigation Text Augmented Graph. We provide a framework for solving the navigation problem by breaking it into two step. We then focus on providing a solution to the second step, the TAG subgraph retrieval problem. Using this solution, we provide a demonstration of the method demonstrating its efficacy. Finally, we propose future directions, and conclude the work.

CS510'24, May 07, 2024, Champaign, IL

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORKS

Text information retrieval has a very long and rich history spanning multiple decades of innovative research [Kobayashi and Takeda 2000; Mitra and Chaudhuri 2000]. This history includes a great deal of work on document retrieval from a corpus of documents. However, less attention has been paid to how to enhance navigating these documents after they have been retrieved.

That isn't to say, however, that there has been no work in this line. There have been works focused on visualization of retrieved documents, such as works using topic modeling to support semantic exploration [Eisenstein et al. 2011]. There have also been works on so called 'wikification' which focus on the disambiguation of named entities in documents [Cheng and Roth 2013; Roth et al. 2014]. Implicitly, the wikification of a document collection can be helpful for navigation in that it allows users to disambiguate when two documents are referring to the same entity.

There has also been work exploring implicit semantic links in documents [Ensan and Bagheri 2017; Green 1999; Kilicoglu et al. 2024]. While the linking of documents implicitly creates a graph, these sorts of works usually focus on creating the links, and not necessarily exploring the graph structure. That isn't to say, however, that there isn't works which also explore the graph structure. For example, there are works that explore leveraging graph structure in order to create these semantic links as well as works that explore using semantic links to enhance a graph embedding [X. Han et al. 2011; Yu, Jin, Liu, et al. 2023]. However, the former methods are admittedly on the older side, and the latter methods are not so pertinent to navigation.

On the topic of graphs, the authors would be remiss not to mention the closely related and highly influential topics of Knowledge Graphs and Heterogeneous information Networks. While knowledge graphs aren't commonly used for navigating, the have seen a great deal of recent study utilizing modern methods such as large language models [X. Chen et al. 2020; Pan et al. 2024]. Further, there is a body of work on so called 'knowledge-graph-completion' which is related to our desire to do what could be considered 'TAG-completion' [Z. Chen et al. 2020; Y. Wei et al. 2023]. Finally, the study of heterogeneous information networks provides numerous tools which could be applicable to the study of TAGs [C. Shi et al. 2017; Sun and J. Han 2012]. One such tool is the analysis of instances of hypernymy, something that could be important for the direction of TAGs [Y. Shi et al. 2019].

3 PROBLEM DESCRIPTION

We can start by following the definition for a **Semantically Linked Document Collection** (SLDC) provided in a class project for CS 512 [Alvarez 2024][1].

Definition 1. Semantically Linked Document Collection (SLDC) A SLDC S is a collection of n documents $D = \{d_1, d_2, ..., d_n\}$, where each document d_i is represented as a variable sequence of words $d_i = (w_{i1}, w_{i2}, ...w_{im})$ where w_{im} is the mth word of the ith document. Further, S is associated with a finite set of forms $\mathcal{F}$. Each form is an abstract idea or entity. We define a function $\tau : D \rightarrow P(\mathcal{F})$ mapping documents in D to a subset of $\mathcal{F}$. We say that two documents $d_i, d_j \in D$ are semantically linked if and only if $\tau(d_i) \bigcup \tau(d_j) \neq \emptyset$.

Using this definition, we can create a new definition for a computational problem:

Definition 2. Query Aligned SLDC Graph Problem We define a problem, taking an input of an SLDC, S, and a query q where we want to use the SLDC to generate a graph representation of the documents and their semantic connections. Formally, we define as follows:

Query Aligned SLDC Graph (QASLDCG) Problem

Input:
An SLDC $S := \{D = \{d_1, d_2, ..., d_n\}, \mathcal{F} = \{f_1, ..., f_m\}\}$
Query $q = (w_1, w_2, ..., w_m)$

Output: A graph $\mathcal{G}' \in \Omega_q(S)$ where

$$\mathcal{G}' \neq \emptyset$$
$$\Omega_q(S) : S \rightarrow G, G = (V, E)$$
$$G' = (V', E') \in \Omega_q(S) \iff$$
$$\forall v \in V', \tau(v) \cup \tau(q) \neq \emptyset, \text{ and } \forall e \in E', \tau(e) \cup \tau(q) \neq \emptyset$$

If we are able to solve this problem, we would be able to overcome the difficulties in navigating unstructured document corpi as documents would all have relevent semantic links. The reason that the problem is defined as having a query is that if documents were linked to every single document that is semantically relevant, there could be a great abundance of links which could actually impede navigation due to being overwhelming. Consider, for example, a math textbook. A math textbook might use various entities from pop-culture in a word problem, however, unless the user is only intrested in math, these semantic links would not be relevant and would only distract from the important information.

In order to solve the QASLDCG[2] problem, we propose decomposing the problem into two sub-problems. Each of these sub problems are interesting in their own-rite and present a unique area of study.

(1) **Nav-TAG Construction Problem**
(2) **TAG Subgraph Retrieval Problem**

While this project will focus on the second sub-problem, we will also briefly cover the first. The reason for this is that as of the time that this is written, there are no known pre-built TAG nor Nav-TAG datasets. Therefore, to even attempt to solve the TAG Subgraph retrieval problem, one must first have a TAG from which to retrieve a subgraph from.

[1]This work is the author's own, but was originally defined in a different class. See appendix A for additional notes on the split of work between classes.
[2]An admittedly less intuitive acronym

(1) Citation Network: A citation network is a collection of scientific documents that contain citation relations to each other. For a Citation Network to be a TAG, one could conceptualize the scientific papers as the nodes, annotated by either the abstracts or full text of the paper. The edges, would be the citation relation, i.e. there is an edge between A and B if A cites B. It makes sense for these edges to be directed. The text annotating this edge could be the citation context or text describing how A uses the knowledge from B.
(2) Wikipedia: In Wikipedia, one could imagine the nodes as the named entities for which pages are made. These nodes would, of course, be annotated by the page text. The edges would be the internal Wikipedia links, and the edge text could be the link context.
(3) Knowledge Graph: In a sense, Knowledge Graphs can be thought of as a very particular type of TAG. The nodes of the knowledge graph would be the nodes of the tags. The text describing the entity, or perhaps additional elaborative text about the enitity, would be the text annotating the nodes. The edges, once again, would simply be the edges of the knowledge graph. KG edges are attributed so the text annotating the TAG edges could be an expansion of these attributes.

Figure 1: A few examples of datasets that could be conceptualized as TAGs

4 NAV-TAG CONSTRUCTION PROBLEM

In order to understand and define the Nav-TAG construction problem, we must first define TAGs and Nav-TAGs. We will start by introducing the basic idea of a TAG, and then go on to define the Nav-TAG. Only once these two data-structures have been defined, can we formally define the Nav-TAG construction problem. Finally, we will detail a naive approach to TAG and Nav-TAG construction. It is this approach that we use to build our input dataset so that we can solve the TAG Subgraph Retrieval Problem.

4.1 Text Augmented Graphs

At a high level, a Text Augmented Graph (TAG) can be described as a versatile framework for organizing textual information. A TAG is a graph that features both nodes and edges annotated by text data. There are many existing datasets that can be conceptualized as tags. A few are elaborated on in figure 1

Formally, we can define a TAG as follows:

Definition 3. (Text Augmented Graph)
A TAG is defined as a graph $G = (V, E)$ with

(1) a node mapping function $\phi : V \rightarrow$ Text
(2) an edge mapping function $\psi : E \rightarrow$ Text

which meets the following conditions

(1) $\forall v_i \in V$, $\phi(v_i) = X_i$ where $X_i = (w_0, ...w_n)$ is the text corresponding to the node v_i.
(2) $\forall e_{ij} \in E$, $\psi(e_{ij}) = Z_{ij}$ where $Z_{ij} = (w_0, ...w_m)$ is the text corresponding to the edge e_{ij}.

4.2 Navigation-TAGS

At a high level, a Navigation-TAG is a directed TAG where the documents associated with the TAG's nodes are split into sections. There are various conditions that a Nav-TAG must satisfy, for instance, for every document in a corpus there must be a corresponding node. This structure allows us to capture part of the requirements to solving the QASLDCG problem.

Formally, we definei the Nav-TAG as follows:

Definition 4. Navigation-TAG (Nav-TAG)
Given a corpus of documents $C = \{X_0, ..., X_n\}$, document partitioning function $\Pi(X_i) = (S_0, ..., S_m)$, a set of forms $\mathcal{F}$, and a form mapping function $\tau : D \rightarrow P(\mathcal{F})$, A Nav-TAG is defined as a directed graph $G = (V, E)$, node mapping function $\phi : V \rightarrow Text$, and an edge mapping function $\psi : E \rightarrow Text$ where the following conditions are met.

(1) Document Coverage:

$$\forall X_i \in C, \exists v_i \in V : \phi(v_i) = X_i$$

(2) Section Coverage:

$$\forall S_j \in \{\Pi(X_i) | X_i \in C\}, \exists v_j \in V : \phi(v_j) = S_j$$

(3) Section Document link:

$$\forall S_j \in \{\Pi(X_i) | X_i \in C\}, \exists e_{ij} \in E$$

(4) Section Section link:

$$\forall S_s, S_t \in \Pi^*(C), \tau(S_s) \cup \tau(S_t) \neq \emptyset \iff \exists e_{st} \in E$$

4.3 Formal definition of Nav-TAG Construction Problem

With this definition of a Nav-tag, we can start to see how it fits into the general QASLDCG problem. In particular, we notice how a Nav-TAG is associated with a corpus $C = \{X_0, ..., X_n\}$ and a set of forms $\mathcal{F}$. In other words, a Nav-TAG is associated with an Semantically Linked Document Collection. As such, we will find it useful to define the computation problem of Nav-TAG construction as follows.

Nav-TAG Construction Problem

Input:
An SLDC $S := \{D = \{d_1, d_2, ..., d_n\}, \mathcal{F} = \{f_1, ..., f_m\}\}$

Output: A TAG $\mathcal{G} = (E, V)$ where

(1) $\forall X_i \in C, \exists v_i \in V : \phi(v_i) = X_i$
(2) $\forall S_j \in \{\Pi(X_i) | X_i \in C\}, \exists v_j \in V : \phi(v_j) = S_j$
(3) $\forall S_j \in \{\Pi(X_i) | X_i \in C\}, \exists e_{ij} \in E$
(4) $\forall S_s, S_t \in \Pi^*(C), \tau(S_s) \cup \tau(S_t) \neq \emptyset \iff \exists e_{st} \in E$

4.4 Naive approach to Nav-TAG construction

Despite it not being the focus of our study, in order to be able to perform any empirical investigation of an approach to TAG Subgraph Retrieval, we must first be able to produce a TAG. Likewise, since we have introduced the QASLDCG problem, it seems sensible to provide at least a naive solution to both sections.

At a high level, our approach to Nav-TAG construction is to eplicitly define $\mathcal{F}$ as the set of all named entities in our dataset. As it happens, this corresponds with the title of each page in the dataset. This allows for a very straightforward definition of τ: if the named entity f appears in a piece of text x, then $f \in \tau(s)$. More specifically, for all $f_i \in \mathcal{F}$ if f_i is a sub-string of s, then $f_i \in \tau(s)$.

With this simple definition, we can create a Nav-TAG by first creating a simple TAG. We do this by simply iterating through every sentence s_j in every document d_i in the corpus and check for all $f \in \mathcal{F}$ if $f \in s_j$. Then, for each $f \in s_j$ we create a link between the current page d_i and the page associated with f.This method is not particularly efficient, with simple analysis showing its $O(Dn^2)$ where D is the maximum size of a document and n is the number of nodes. More efficient approaches could be taken, such as perhaps some using prefix trees, however, our total number of nodes is sufficiently small that this inefficient method still runs fast enough to be unproblematic.

Now that we have a TAG, we need to break it up by section in order to turn it into a Nav-TAG. At a high level, we iterate through each node and split it by some delimiter. Then, for each of these sections, we go through and determine which of the parent node links were in the text associated with this section. These links are then copied to the section node. The pseudo-code for this approach can be see in Algorithm 1 .

Algorithm 1 Split Documents

```
section_nodes ← {}
section_edges ← {}
for node in self.nodes do
    text ← GetNodeText(self, node)
    parent_node_edges ← nodes[node].edges
    sections ← SplitDocument(text, delimiter)
    for sec in sections do
        s_node ← SectionNode(node, sec)
        sub_node_text ← text[sec]
        for p_n_edge in parent_node_edges do
            e_text ← GetEdgeText(p_n_edge)
            if e_text in sub_node_text then
                e_prime ← Edge(e_text, ...)
                section_edges[e_prime.name] ← e_prime
            end if
        end for
    end for
end for
nodes.update(section_nodes)
edges.update(section_edges)
```

4.4.1 Approach limitations. It is worth noting the limitations of this approach. Perhaps the largest limitation of this approach is that there is no sense of entity disambiguation. For instance, if an article were to refer to "Turing" but not "Alan Turing", then this approach would not create a link between the article and the article about Alan Turing. Another limitation is continuity across sentences. For instance, If a piece of text is "John Doe was one of the most influential people of the 2010s. In 2019, he invented the internet 2.0". In the second sentence, "he" references John Doe, and as such this sentences is also about John Doe. However, since this approach does not account for inter-sentence context, it wouldn't be able to properly mark the second sentence.

5 SUBGRAPH RETRIEVAL

The second sub-problem of the QASLDCG problem is the TAG Subgraph Retrieval Problem. At a high level, this problem involves taking a TAG and according to some query returning a subgraph that is a pruned version of the original TAG, keeping only those nodes and edges which are relevant to the query. This sub-problem is the main focus of this project and will be the focus for the rest of the paper. We start with a formal definition of the problem.

5.1 Formal Definition of TAG-Subgraph Retrieval Problem

We start with the definition of the problem, then offer some commentary on the definition.

Definition 5. TAG Subgraph Retrieval Problem

TAG Subgraph Retrieval Problem

Input:
A TAG $\mathcal{G} := \{G = (E, V),\}$
A set of forms: $\mathcal{F} = \{f_1, ..., f_m\}$
A form mapping function: $\tau : D \rightarrow P(F)$
A query: $q = (w_1, ..., w_n)$

Output:
A graph $\mathcal{G}' \in \Omega_q(S), \mathcal{G}$ where

$$\mathcal{G}' \neq \emptyset$$
$$\Omega_q(S) : S \rightarrow G, G = (V, E)$$
$$G' = (V', E') \in \Omega_q(S) \iff$$
$$\forall v \in V', \tau(v) \cup \tau(q) \neq \emptyset, \text{ and } \forall e \in E', \tau(e) \cup \tau(q) \neq \emptyset$$

It is worth noting what this definition does and does not require. These properties follow from the QASLDCG Problem definition. In analysing the output requirements, we notice that while it requires that G' is not an empty set, it has no other size requirement nor does it attempt to maximize the size of the retreived graph $\mathcal{G}$. The reason for this is to allow flexibility in the particular implementations for solving the problem. Since our intention is to use this subgraph for navigation, perhaps the desired output size would be different than a use-case where the goal is to

5.2 Proposed approach

At a high level, our approach for retrieving the a subgraph that is relevant to query q, consists of an iterative method of querying the most relevant nodes, and then querying the most relevant edges. Since the edges point to nodes, querying the most relevant edges creates a new set of most relevant nodes, and the process can be repeated.

More formally, we start by defining the following distance metrics:

(1) Query-Node Distance function: $f_{qv} : Q \times V \rightarrow \mathbb{R}$
(2) Edge-Node Distance function: $f_{qe} : Q \times E \rightarrow \mathbb{R}$

Where both f_{qv} and f_{qe} are defined by the cosine similiarity between the embedding of the query and the embedding of the text associated with the node or edge. In other words, if we define the embedding for a term i in document j as

$$f_{emb}(X) = \text{all-MiniLM-L12-v2}(X)$$

we define f_{qv} and f_{qe} as follows:

$$f_{qv}(q, v) = \frac{f_{emb}(q) \cdot f_{emb}(\phi(v))}{||f_{emb}(q)||||f_{emb}(\phi(v))||}$$

$$f_{qe}(q, e) = \frac{f_{emb}(q) \cdot f_{emb}(\psi(e))}{||f_{emb}(q)||||f_{emb}(\psi(e))||} + \frac{f_{emb}(q) \cdot f_{emb}(\psi'(e))}{||f_{emb}(q)||||f_{emb}(\psi'(e))||}$$

where ψ is the edge text mapping function from the TAG and ψ' maps an edge to the text describing the title of the edge, i.e. the two nodes the edge connects. This was added to give additional embedding weight to the most important feature of the edge, the two nodes it connects. We observed that this addition seemed to improve the performance of approach, although a more in-depth study would need to be performed to find empirical evidence for this claim.

In order to speed up run-time computation, we start by pre-computing the embedding of all nodes and edges. This means that at run-time we only need to compute one embedding: the embedding of the query. At run-time, we use our pre-computed embedding and our computed query embedding to create distance matrices. At each step we look to calculate either:
This set for nodes

$$\underset{N \subset V, |S_n| = k}{\arg\max} \sum_{v \in N} f_{qv}(q, v)$$

or this set for edges:

$$\underset{E'_n \subset E_n, |E'_n| = k}{\arg\max} \sum_{e \in E'_n} f_{qe}(q, e)$$

This is accomplished by taking out distance matrices and sorting them largest to smallest. We then use this sorted list to retrieve the top-k results. The full psuedocode for our proposed method can be seen in Algorithm 2

Algorithm 2 Top-K Graph Expansion Algorithm

Require: *q_nodes*: List of initial query nodes
expansion_factor: Number of nodes to expand per query node
query: Query input
steps: Maximum number of steps

```
all_edges ← {}
q_emb ← model.encode(query)
layer_n_nodes ← [q_nodes]
while len(layer_n_nodes) ≤ steps do
    layer ← []
    for n in layer_n_nodes[len(layer_n_nodes) − 1] do
        edges_of_n ← get_edges (n)
        e_text_scores ← score(q_emb, edges_of_n)
        top_k_edges ← get_top_scores(e_text_scores)
        for edge in top_k_edges do
            new_node ← get_sink_of_edge(edge)
            if new_node in all_nodes then
                layer.append(new_node)
                all_edges.add(e)
            end if
        end for
    end for
    layer_n_nodes.append(layer)
end while
```

5.2.1 Run-Time analysis of proposed Method. While the proposed method has the potential of being quite inefficient due to needing to sort the nodes and edges, careful pre-computation allows us to maintain efficiency. Let k be the so called "expansion factor" of the algorithm, i.e. the number of additional nodes added per node per step. For example, if we start with 4 nodes, and let k=10, after 1 step we will have 4 + 40 nodes. As it happens, in our approach we also use k as the number of initial nodes. As such, we can say that if s = the number of steps, that our algorithm scales with $O(Ak^s)$ where A is the computation cost per node.

To determine A, let us start by considering the work done per node. For each node we need to retrieve the nodes neighbors, and rank them to find the k most relevant. We can retrieve the neighbors in constant time since this is stored in the TAG data structure. As such, it is really a matter of retrieving the top k-nodes from these neighbors. Since we have no bound on the size of the neighborhood of a node, we can upper bound this operation by O(|E|). An O(|E|) approach to this would be iterating through the already sorted edges and checking if each is in the node neighborhood. If you know O(|neighborhood|) « O(|E|), which is likely the case on average, you could opt to retrieve the embedding of each node in the neighborhood and then sort them. Either way, let us upper-bound O(A) by O(|E|).

The last consideration is the time it takes to compute and sort the embedding for the edges and the nodes. This is simply $O(N \log N + |E| \log |E|)$ due to having to sort the distance matrices. This gives us the total runtime as $O(N \log N + |E| \log |E| + |E|k^s)$. While it might seem like this result is exponential in the number of steps, let us consider the upper-bound of k^s. We notice that since we are adding nodes to be included in out subgraph, it really only makes sense to add nodes up to $k^s = N$ since there would be no point in adding nodes already added. Therefore, we can say that our final runtime is $O(N \log N + |E| \log |E| + |E|N)$. In typical cases, i.e. where k and s are modest, the runtime will be dominated by the sorting of the nodes and edges.

6 EXPERIMENTS / DEMO

Since we are looking at a new and unique problem, TAG Subgraph Retrieval, we have the difficult task of determining how to evaluate the approach. We have decided to focus on creating a demonstration of the approach working as intended instead of coming up with quantitative measures of performance. The reason for this is that since we are attempting to perform a proof of concept, as opposed to attempting to compare multiple methods, we merely need to show our method works. Any quantitative methods would be in a void since we have no method to compare against. As such, once we come up with additional methods, and are looking for a way to objectively compare the methods, we will need to come up with some quantitative measure.

In order to demonstrate our method, we have decided to curate a dataset from which we can create a Nav-TAG. This Nav-TAG will then act as an input for our method, along with a few hand written queries, in order to subjectively determine the extent to which the proposed method works. As such, we first start by curating a dataset.

Model	Nodes	Edges
Base	2933	14249
Nav-TAG	38550	55613

Table 1: Size of dataset, before and after section splitting. The Base dataset consists of all the scraped pages and their internal links. The Nav-TAG is the split dataset, where each node is a page or a page section. Edges are duplicated from the parents to the appropriate section.

6.1 Dataset

The dataset we have decided to curate for our demonstration is a subset of Wikipedia's philosophy category. In particular, we are intrested in the following categories from wikipedia:

(1) Category:Philosophers_by_field
(2) Category:Philosophical_schools_and_traditions
(3) Category:Philosophical_concepts
(4) Category:Branches_of_philosophy

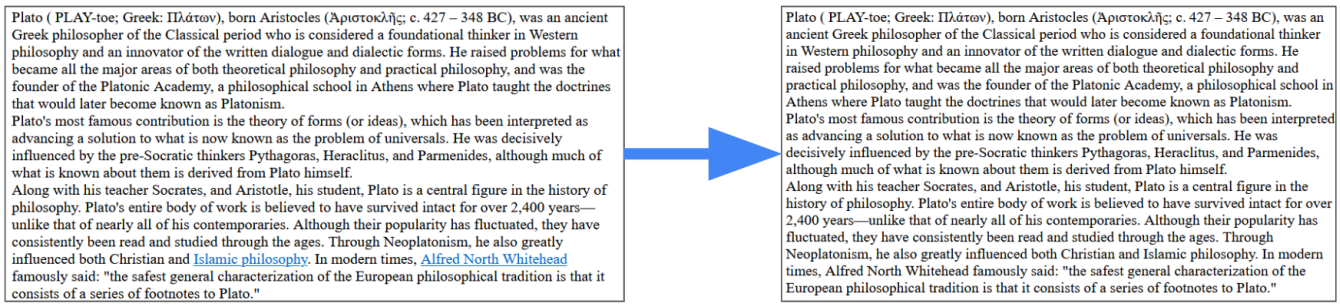
These four subcategories were chosen due to the authors prior knowledge being higher related to these categories than to other categories (for instance, history of philosophy or philosophical organizations). This data was scrapped using Wikipedia's API. This dataset was also built into a Nav-TAG. This was done in two parts. The first was creating the TAG, this was actually done as the data was scrapped for convenience. Then, this tag was loaded and split using the split documents algorithm described in algorithm 1. The resulting size of this dataset can be seen in table 1.

6.2 Results

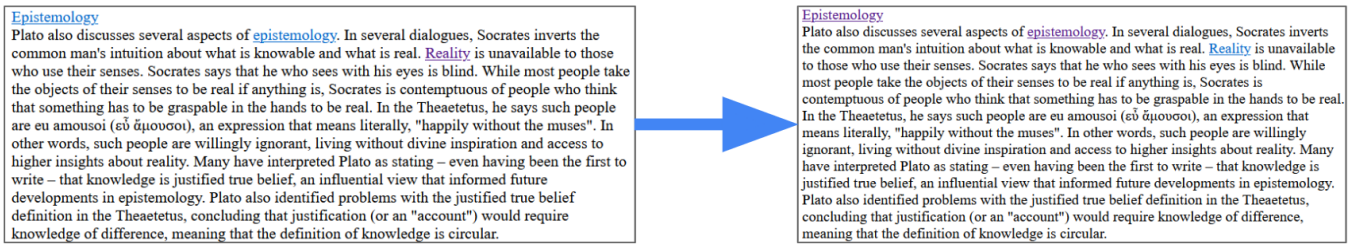
A demonstration of an application of our approach can be seen in figure 2. For this particular case, and other similarly simple cases, the model worked quite well. This is especially true for more pages that are more central in the subgraph. For instance, for the prompt about Plato and Epistemology, the page on Plato had links pruned appropriately. However, when getting away from the center, the quality degrades. For instance, even on the page about Epistemology the quality was lower. The reason for this is likely that since in the TAG, the epistemology page doesn't have any links to Plato (despite talking about Plato, likely due to the limitations of how we built the TAG). As a result, all the links on the page are about equally related to "Plato and epistemology", at least from the level that the model used can distinguish from the edge text.

While there is a certain sense in which the demonstration has shown the efficacy of the method, an unfortunate factor in our demonstration is that the Nav-TAG we are using as input is not actually that dense. In the case that the a page only has a handful or so of links, and you are picking the top 5, it is inevitable that some of those chosen will not be as pertinent as you might wish.

Despite all of this, I would say that the proposed method has at least shown the promise / potential upside of using a Nav-TAG to streamline navigating semantically linked documents.



(a) On the Plato article summary section, we see that the links to "Islamic philosophy" and "Alfred North Whitehead" are removed since they are not relevant.



(b) On the Plato article "epistemology" section, we see that the links to "Epistemology" and "Reality" are maintained since they are relevant

Figure 2: An demonstration of our TAG Subgraph Retrieval method on the Wikipedia article for Plato for the query "Plato on Epistemology"

7 FUTURE WORK

While our project has demonstrated the efficacy of the ideas proposed, it is also clear that there is ample opportunity for future work. It is the opinion of the author that the first and most important step would be to improve the Nav-TAG generation. Currently, we are working on a dataset which has a pre-known set of named entities. However, in many real world document sets, not only is there no known set of named entities, but there is also the problem of disambiguation. By leveraging prior works on both of these topics, perhaps one could invent a new and improved method for leveraging the semantic links between documents and building a Nav-TAG.

Another question regarding the building of TAGs is heterogeneity. On the Wikipedia dataset, it is convenient that each named-entity has a page associated with it. As such, when one page references a named entity, it is also referencing that page. However, for document collections in general this is not necessarily true. Suppose that one had a collection of essays and a collection of Wikipedia articles. While it is clear how one might link a named entity in an essay to the appropriate Wikipedia page, how might one link an essay to an essay that has a similar concept? One can imagine that if they let the set of forms $\mathcal{F}$ become arbitrarily complex, such that the common idea between two essays is a form, then one could link the two insofar as one is able to determine that both sections are related to that particular form. Perhaps this is something that could be accomplished with Large Language Models.

Finally, another important line of future work is improving the TAG Subgraph retriever. From the experiments, it became clear that while simply choosing the top k is a workable solution, it seems like it would be a lot better if the number of links retained per page was dynamic. It seems highly likely that it is often the case that some pages have many relevant links whereas some have very few. By forcing each page to use the same number, it is inevitable that relevant links will be excluded and irrelevant links will be included.

8 CONCLUSION

In conclusion, this paper addresses the challenge of exploring connections between unstructured document collections. In particular, we propose a few computational problems to capture this difficulty. We also propose the Navigation Text Augmented Graph as a framework for potentially addressing these problems. This work's primary technical contribution focuses on addressing the TAG subgraph problem. We propose an approach to solving this method and use it to demonstrate a Nav-TAG proof of concept. We also propose many directions for future work to expand on the foundations laid in this project.

REFERENCES

Dean Alvarez. 2024. "Question Answering on Semantically Linked Document Collections with LLMs and Text Augmented Graphs." for a project in CS 512 Spring 2024. (2024).

Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. "A review: Knowledge reasoning over knowledge graph." *Expert systems with applications*, 141, 112948.

Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. "Knowledge graph completion: A review." *Ieee Access*, 8, 192435–192456.

Xiao Cheng and Dan Roth. 2013. "Relational inference for wikification." In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1787–1796.

Jacob Eisenstein, Duen Horng "Polo" Chau, Aniket Kittur, and Eric P. Xing. 2011. *TopicViz: Semantic Navigation of Document Collections.* (2011). arXiv: 1110.6200 [cs.HC].

Faezeh Ensan and Ebrahim Bagheri. 2017. "Document Retrieval Model Through Semantic Linking." In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (WSDM '17). Association for Computing Machinery, Cambridge, United Kingdom, 181–190. ISBN: 9781450346757. DOI: 10.1145/3018661.3018692.

S.J. Green. 1999. "Building hypertext links by computing semantic similarity." *IEEE Transactions on Knowledge and Data Engineering*, 11, 5, 713–730. DOI: 10.1109/69.806932.

Xianpei Han, Le Sun, and Jun Zhao. 2011. "Collective entity linking in web text: a graph-based method." In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '11). Association for Computing Machinery, Beijing, China, 765–774. ISBN: 9781450307574. DOI: 10.1145/2009916.2010019.

Halil Kilicoglu, Faezeh Ensan, Bridget McInnes, and Lucy Lu Wang. 2024. "Semantics-enabled biomedical literature analytics." *Journal of Biomedical Informatics*, 150, 104588. DOI: https://doi.org/10.1016/j.jbi.2024.104588.

Mei Kobayashi and Koichi Takeda. June 2000. "Information retrieval on the web." *ACM Comput. Surv.*, 32, 2, (June 2000), 144–173. DOI: 10.1145/358923.358934.

M. Mitra and B. B. Chaudhuri. May 2000. "Information Retrieval from Documents: A Survey." *Information Retrieval*, 2, 2, (May 2000), 141–163. DOI: 10.1023/A:1009950525500.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. "Unifying Large Language Models and Knowledge Graphs: A Roadmap." *IEEE Transactions on Knowledge and Data Engineering*, 1–20. DOI: 10.1109/TKDE.2024.3352100.

Dan Roth, Heng Ji, Ming-Wei Chang, and Taylor Cassidy. 2014. "Wikification and Beyond: The Challenges of Entity and Concept Grounding." *ACL (Tutorial Abstracts)*, 7.

Urvi Shah, Tim Finin, Anupam Joshi, R. Scott Cost, and James Matfield. 2002. "Information retrieval on the semantic web." In: *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (CIKM '02). Association for Computing Machinery, McLean, Virginia, USA, 461–468. ISBN: 1581134924. DOI: 10.1145/584792.584868.

Yuanchun Shen. 2023. *SRTK: A Toolkit for Semantic-relevant Subgraph Retrieval.* (2023). arXiv: 2305.04101 [cs.IR].

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. "A survey of heterogeneous information network analysis." *IEEE Transactions on Knowledge and Data Engineering*, 29, 1, 17–37. DOI: 10.1109/TKDE.2016.2598561.

Yu Shi et al.. 2019. "Discovering Hypernymy in Text-Rich Heterogeneous Information Network by Exploiting Context Granularity." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (CIKM '19). Association for Computing Machinery, Beijing, China, 599–608. ISBN: 9781450369763. DOI: 10.1145/3357384.3357866.

Yizhou Sun and Jiawei Han. 2012. *Mining heterogeneous information networks: principles and methodologies.* Morgan & Claypool Publishers.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. "KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion." In: *Findings of the Association for Computational Linguistics: EMNLP 2023.* Association for Computational Linguistics. DOI: 10.18653/v1/2023.findings-emnlp.580.

Zhizhi Yu, Di Jin, Ziyang Liu, Dongxiao He, Xiao Wang, Hanghang Tong, and Jiawei Han. Feb. 2023. "Embedding text-rich graph neural networks with sequence and topical semantic structures." *Knowledge and Information Systems*, 65, 2, (Feb. 2023), 613–640. DOI: 10.1007/s10115-022-01768-4.

Zhizhi Yu, Di Jin, Jianguo Wei, Ziyang Liu, Yue Shang, Yun Xiao, Jiawei Han, and Lingfei Wu. 2022. *TeKo: Text-Rich Graph Neural Networks with External Knowledge.* (2022). arXiv: 2206.07253 [cs.SI].

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. "Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics. DOI: 10.18653/v1/2022.acl-long.396.

A AN IMPORTANT NOTE ON SEPARATION OF WORK

The two class projects I have done this semester have both been about TAGS. They have focused on different aspects of TAGS, subgraph retrieval (this project) and question answering (CS 512). Nevertheless, they share a code base.
The following files were created for the purpose of this class project

(1) src/tags/nav_tag (Directed Edge and NavTAG.(from_xml, add_ege, to_html)
(2) /experiments/tag_subgraph/*
(3) data/scrapers/wiki_phil_scrapper.py

In addition to these considerations about the code, with regards to the definitions, while I have been thinking about how to best define the TAG and the Nav-TAG the specific definitions I have used are unique to this write-up. They are, of course, heavily inspired by my prior thinking on this matter as part of my individual research. That said, I have only been thinking about how to define the Query Aligned SLDC Graph problem within the context of this project.