Applied Ontology 1 (2024) 1–5
IOS Press

Integrating Ontologies and Large Language Models to Implement Retrieval Augmented Generation (RAG)

Michael DeBellis*[a], Nivedita Dutta[b], Jacob Gino[c], Aadarsh Balaji[d]
[a] *michaeldebellis.com, E-mail: mdebellissf@gmail.com*
[b] *Independent Researcher*
[c] *University of Wisconsin, Madison*
[d] *University of California, Berkeley*
E-mails: niveditadgp@gmail.com, jacobkgino@gmail.com, aadarsh.balaji@gmail.com

Abstract. Large Language Models have captured the imagination of the public and the technical community. As powerful as they are they have problems that prohibit their use for highly skilled users. These issues are hallucinations, bias, black-box reasoning, and lack of domain depth. One of the most popular architectures to alleviate these problems is Retrieval Augmented Generation (RAG). In a RAG architecture the LLM is utilized to generate vectors and to parse and generate natural language. The knowledge base for a RAG architecture is typically a set of documents focused on a particular type of vertical (question answering) or horizontal (domain) set of use cases as opposed to the general knowledge base of an LLM. Typically, the corpus for the RAG knowledge base is stored in a relational database. This project investigates the use of an ontology and knowledge graph to form a domain specific knowledge base for RAG in order to leverage LLMs for specific domains without the four problems that typically make them inappropriate for mission and life critical domains. The domain is support of dental clinicians in India who face specific problems that can be significantly improved by better, timely, and easily accessible access to the latest knowledge on dental material products. We demonstrate that using an ontology and knowledge graph to implement RAG has several benefits such as rapid agile development and retrieval by reformulation browsing.

Keywords. Large Language Models (LLM); ontology; Knowledge Graph; Retrieval Augmented Generation (RAG); dentistry

1. Introduction

The goal of this project is to develop a semantic search tool for dental products and materials. The first part of the research was to work with the OBO community to develop an ontology that modeled the domain. The next step described in this paper is to use this ontology as the foundation for a semantic search tool for dental clinicians in India. After evaluating various approaches to semantic search, it was clear that using a Large Language Model (LLM) integrated with the ontology in an architecture known as Retrieval Augmented Generation (RAG) was the most appropriate method. This paper describes the first prototype system that implements such an architecture.

Some brief notes on terminology and presentation in this paper: *ontology* refers to what is known as the T-Box in the semantic web community; the classes and properties. *Knowledge graph* refers to the ontology plus the data, i.e., the T-Box and the A-Box. Ontology is used when discussing creating or editing the model in Protégé and knowledge graph for the much larger model plus data stored in the AllegroGraph triplestore. Names of *classes* have each word capitalized with a space between them for

*Corresponding author: Michael DeBellis. E-mail: mdebellissf@gmail.com.

readability and are in Calibri font, e.g., My Class and properties have the first word in lower case and all other words in upper case, also in Calibri, e.g., my Object Property. *Object properties* typically follow the *has Property* and *is Property Of* naming scheme for a property and its inverse. *Data properties* tend to be simpler (and many are from reusable vocabularies such as Dublin Core) e.g., dct:abstract, dct:creator. All names of entities in the ontology use singular case as is the standard. However, we will sometimes use the plural of a name for readability. Calibri font is also used for examples of SPARQL code and output from ChatGPT.

The structure of this paper is as follows: this section provides an overview of the domain problem, why it is a difficult and important problem, the innovative approach to solve it, and a review of relevant previous literature. Section two describes the principle of the approach and the reasons for utilizing the RAG architecture integrated with an ontology. Section three describes the technical architecture of the current system and a specific use case of a clinician posing a question, the answer that is returned, how the answer is created, and how this compares to the answer that a generic LLM would create. Section four is a critical evaluation of related work integrating ontologies and RAG, a discussion of future plans, and a conclusion discussing general lessons that can be learned from this work.

1.1. The Need for better, faster, and easier access to the best knowledge about dental materials and products

The first part of the project, described in (Dutta, 2023), was to develop an OWL ontology for Dental Materials and Products. A critical distinction in this ontology is between Dental Materials and Dental Material Products. Dental Materials are raw materials such as dental cement and resin. These are the raw materials used to create Dental Material Products. Dental Material Products are manufactured and typically use two or more Dental Materials in unique ways to solve problems such as direct restoration which is our current focus. There already was an excellent reusable OBO vocabulary for Oral Health and Diseases (Duncan, 2020) that was reused for this project. However, it did not cover the complex space of dental materials and products. In the first phase of the project, the team collaborated with the OBO developers of the OHD ontology to develop the ontology so that it was consistent with OBO standards and could utilize OHD. In addition, the OHD team incorporated many of the dental material entities from this project.

The next phase (the subject of this paper) is to develop a semantic search tool for journal articles, product manuals, and other relevant knowledge on the topic. Target users are doctors of dentistry and other dental professionals in India. The reason such a tool is needed are:

1. The field is extraordinarily complex and rapidly evolving. It is complex because in addition to requiring medical knowledge it also requires understanding many other domains such as materials engineering, chemistry, metallurgy, general medicine, and surgery. Advancements in new materials used for dental products are occurring at a very rapid rate due to scientific progress leading to new kinds of materials and industrial progress that leverages the scientific advances for new products. Thus, it is difficult for busy clinicians to remain current on the latest updates and best practices in the field.
2. In India and other developing nations there is a large population who can barely afford healthcare and will do whatever possible to decrease their costs. As a result, they and the clinicians that support them often fall victim to using fraudulent or defective materials. For example, it is common for products to be taken off the market in nations with strong regulation

such as the United Kingdom only to have those products resold to clinicians in India via sites such as eBay (MHRA, 2014), (Price, 2022). The short-term savings from cheaper defective products can have catastrophic long-term consequences.

The goal of the project is to provide a tool that will help busy clinicians by providing them with specific answers to questions, references to papers and other knowledge sources that support the answer, and the ability to easily browse the knowledge base and corpus to find additional relevant knowledge.

1.2. Approach to semantic search: Integrate Large Language Models (LLMs) with ontologies to implement Retrieval Augmented Generation (RAG)

Semantic search is one of the first and most popular applications for semantic web and knowledge graph technology (Berners-Lee, 2001) (Noy, 2019). As described in (DeBellis, 2023), the most common approach is to create a knowledge graph based on the metadata of a corpus for a particular domain as well as to use Natural Language Processing (NLP) techniques to create indices to easily categorize and find appropriate documents and other data. One of the advantages of the ontology approach is that the types of searches are highly flexible since knowledge graphs can exploit graph query languages such as SPARQL which provide many more options than table-based search using SQL (W3C, 2012). In addition, after a user performs an initial search, they can use the semantics of the ontology to further explore relevant concepts and documents.

Several technologies to implement semantic search were evaluated. These included various NLP technologies such as translating natural language to SPARQL and using libraries such as Spacy (Altinok, 2021) and NLTK (Bird, 2009) for Named Entity Recognition (NER). The result of this evaluation was to utilize a Large Language Model (LLM). The advantages of this approach are that vector embedding can be used to represent the meaning of text via the APIs of the LLM. In addition, the capabilities of the LLM to understand questions and generate answers in intuitive natural language can be leveraged. This approach provides the best capabilities of alternative approaches all in one single architecture. More details on all these topics will be discussed in subsequent sections.

Given the amazing progress in LLM technology, it was a valid question to ask if an ontology approach was even necessary. Perhaps LLMs have achieved such a maturity that clinicians could be trained on how to appropriately query an LLM? This is known as prompt engineering. The answer an LLM returns will vary greatly based on the way a question is framed. However, when discussing the question with clinicians and evaluating the literature and the performance of ChatGPT on sample competency questions, it was clear that as powerful as they have become, LLMs alone were not the solution to the problem for four reasons:

1. *Hallucinations*. A conventional LLM such as ChatGPT or Microsoft CoPilot will always generate an answer. Due to the way such LLMs are trained they have little insight into the appropriateness of their answers and while they usually generate answers that are appropriate, they may sometimes (with equal confidence) generate answers that are wrong, incomplete, or don't incorporate the best available evidence. These answers are known as hallucinations. Hallucinations are the price we pay for the general power of Large Language Models and are an inevitable cost of using such models (Xu, 2024).
2. *Bias*. LLMs are trained on terabytes of text culled from sources such as the Internet. As a result, they inevitably reflect the bias of the cultures that produced their training sets.

3. *Black-box reasoning*. LLMs currently cannot do reflection. I.e., unlike symbolic computing that can provide a trace of the deductions that led to a new inference, LLMs can't analyze the chain of inferences that led to a specific answer. This lack of reflection makes LLMs black boxes that provide an answer with no justification. Evidence from previous AI research indicates that experts such as doctors are far less likely to trust an automated reasoner if it cannot justify its conclusion (Verdicchio, 2022). In addition, when the team's domain expert performed experiments where an LLM was prompted to provide references it often provided references that could not be found or were not relevant to the answer.
4. *Lack of depth*. While LLMs are trained on a huge amount of data and do indeed include abstracts of many papers in the domain, they are meant to be general purpose tools rather than domain specific (Lewis, 2020). As a result, they have some knowledge of the domain of dental products and materials but lack the breadth (they do not cover all the papers and other knowledge sources required) and the depth (they only analyze the abstracts of papers rather than the complete paper) required to support clinicians.

For these reasons, generic LLMs are usually not appropriate for providing information to professionals. From customer service representatives to researchers and doctors, one of the most common approaches to leverage LLMs while addressing these issues is Retrieval Augmented Generation (RAG).

1.3. Retrieval Augmented Generation (RAG) solves the problems that make Large Language Models inappropriate for many domain-specific use cases

Retrieval Augmented Generation (RAG) is an architecture that utilizes components of the LLM to parse questions posed in natural languages, to generate semantic vectors for text, and to provide answers by generating natural language. However, rather than use the generic knowledge acquired by an LLM, a RAG architecture substitutes a specific corpus of curated documents. These documents are the only source used to provide answers.

The RAG architecture prevents hallucinations by utilizing a nearest neighbor algorithm to compute semantic distance between a question and the text in the corpus. This includes a threshold that defines the maximum allowable semantic distance for an acceptable answer. If nothing in the corpus is above the threshold then the RAG system displays a predefined answer that not enough information exists to provide a reliable answer to the user's question. RAG reduces bias by utilizing documents which have been developed for a specific audience and adhere to standards for peer review and objectivity. It is not Black Box reasoning because along with the answer generated by the LLM, it always includes the specific text(s) from the corpus that was utilized to generate the answer. It has breadth and depth because rather than using a sampling from the entire Internet, it focuses exclusively on a specific domain.

1.4. Review of previous work

This section provides an overview of previous work on semantic search with ontologies and RAG systems. RAG is one of the biggest trends in applied research at the moment. As described above it fills an essential niche in leveraging the power of LLMs for specific domain problems from customer support to medical advice. This architecture can apply to virtually any domain in business, science, defense, and government. However, the vast majority of RAG implementations use traditional relational databases since that is what most developers are familiar with. The most important innovation of this project is to demonstrate the power of using a knowledge graph to model the domain and corpus documents rather than a relational database. This system is one of the first to do this. In this section we review previous work on semantic search based on knowledge graphs and on implementations of RAG. In section 4.1 we contrast our prototype with other systems that integrate ontologies and LLMs.

1.4.1. Review of previous knowledge graph implementations of semantic search

Semantic search for was one of the first applications that motivated the Semantic Web. In the Scientific American article that launched the Semantic Web (Berners-Lee, 2001), semantic search was one of the motivating use cases. The goal being to allow searching by concepts rather than keywords. Since that time there have been countless research and industry implementations of semantic search across many different domains using many different techniques. A full review of the topic would require a large paper of its own. However, for our purposes we focus on the two most critical issues for semantic search and representative examples of the various approaches to each issue. These two issues are:

1. User Interface (UI). Having a semantic model is only useful if the user can take advantage of it. There are essentially three approaches used for a semantic search UI: Natural Language Processing (NLP), forms-based, and graph-based. NLP attempts to parse natural language (often some subset of natural language) and convert it into a query language. Forms-based create forms that seem similar to traditional search but leverage the semantics of the ontology to define a search in the form. These options can be highly dynamic, taking advantage of the reasoner to tailor the options presented to the user based on previous queries or an understanding of their task. Graph-based UIs utilize the structure of the knowledge graph and visualize the objects of search and the relevant semantic concepts in a graph that the user can browse and manipulate to explore the domain.
2. Indexing. The documents and other objects to be searched for are usually indexed by traditional keywords. Mapping these objects to the ontology is critical to get the value of semantic search. There are several different approaches to indexing.

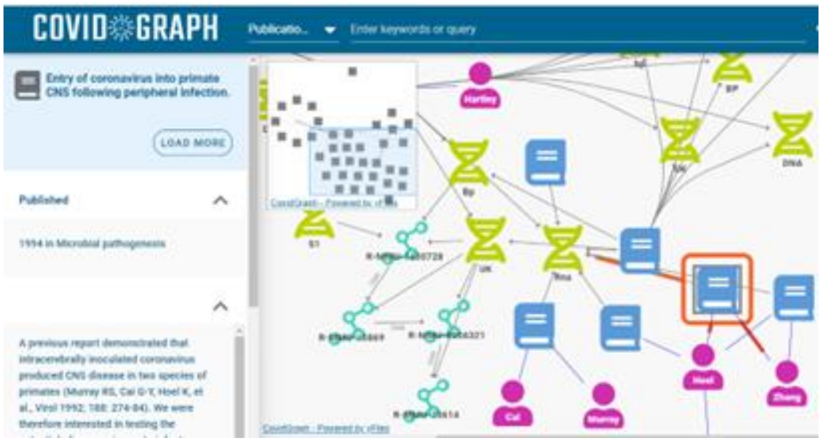
Both these topics are described in detail in the following sections.

1.4.1.1. Semantic search user interfaces

There have been various approaches to parse natural language and transform it into a query language such as SPARQL, Description Logic, or Cypher. In the past, to be tractable it was usually necessary to put constraints on the natural language. For example, some form of structured English along with completion and constraints that prohibit users from entering sentences that the system cannot understand (Sowa, 2014) or as in (Koutsomitropoulos, 2011) to use a hybrid language that is still a structured query language but designed to be more intuitive and English like than query languages designed for developers such as SPARQL. Another approach used by (Benhocine, 2022) is to attempt to parse any form of the natural language but to fall back to partially automatic query generation when the translator is unable to turn the natural language into a complete and valid query. The most ambitious of these approaches is to attempt to transform arbitrary natural language to query languages such as SPARQL (Liang, 2021).

The BACKBORD system (Yen, 1991) developed at the Information Sciences Institute is one of the first examples of ontology-based semantic search. This system utilized the Loom language to model the domain of electronic parts and provided users from the Defense Logistics Agency with a forms-based semantic search tool to find parts by descriptions of their functions and capabilities. Loom is an implementation of Description Logic. The Loom language and reasoner were major influences on the design of the OWL language and reasoner. BACKBORD users could define a search in terms of logical relations, restrictions on relations, and specific values, essentially all the logical formulas that can be expressed in OWL axioms. BACKBORD then created a class description that was classified by the reasoner and all the classes and instances that were subsumed by this new class were returned as the results of the query. Users could then iteratively refine the query by further editing axioms or values from the results of the first query. BACKBORD implemented a paradigm known as *retrieval by reformulation* which has been demonstrated to be similar to the way adults (Williams, 1981) and children (Rutter, 2014) recall information from long term memory. E.g., when recalling a person known in high school and verbalizing their recall protocol, subjects typically remember other memories that are semantically close to the memory they are trying to retrieve and use that memory to refine their search criteria until they focus on the specific memory they are attempting to recall. E.g., trying to remember Mary, they remember Carol and then remember "oh yes Carol was in the band and so was… that's it her name was Mary". Retrieval by reformulation has since been implemented in many different tools for searching the web, documents, and databases and empirical evidence supports the hypothesis that the paradigm is intuitive and a productive way for users to search large complex data (Rieh, 2006), (Rahman, 2023).

Another example of a forms-based semantic search tool is the consumer portal for the Australian Ministry of Health (PoolParty, 2016). This portal utilized ontology models describing the healthcare domain such as diseases, medication, and services to provide a self-service portal for Australian consumers resulting in decreased costs and better healthcare outcomes.



Entry of coronavirus into primate CNS following peripheral infection.

1994 in Microbial pathogenesis

A previous report demonstrated that intracerebrally inoculated coronavirus produced CNS disease in two species of primates (Murray RS, Cai G-Y, Hoel K, et al., Virol 1992; 188: 274-84). We were therefore interested in testing the

Fig. 1. Example of graphical semantic search

An example of the graph-based UI approach for semantic search was the COVID*GRAPH system shown in Fig. 1. COVID*GRAPH (DeBellis, 2022) was developed by a consortium of volunteer researchers who created the system to help researchers and clinicians easily find journal articles and other relevant documents related to containing the spread and developing a vaccine for the Covid virus. Users would begin either by entering keywords or for more technical users a query in the Cypher query language. This would return an initial set of nodes in the knowledge graph that would be displayed visually in a default graph. The user could then manipulate the graph in a multitude of ways and use the domain knowledge of the graph to guide them toward papers, people, and other resources. Another example of semantic document search using a graphical UI from the Covid pandemic was the Cord-19 system (Li, 2022). Both systems modeled similar domains: Genes, Diseases, Documents, and Authors and utilized the structure of the knowledge graph to provide users with a powerful graph-based UI. The literature shows that sophisticated users such as researchers tend to enjoy the power of a graph-based UI whereas less sophisticated users such as consumers or procurement staff prefer a traditional forms-based UI.

1.4.1.2. Semantic search indexing

The most challenging task in building a semantic search system is bridging the gap between traditional information indexed by keywords and semantic models such as OWL. The simplest approach is to have all the searchable data defined in a semantic form. This was the approach used in BACKBORD. While this approach can work for some domains in the era of big data it is seldom viable beyond proof-of-concept demonstrations. Typically, what is required is some form of natural language processing. The most common types of indexing can be divided into three categories of NLP: domain specific, Named Entity Resolution (NER), and vector embedding.

In the domain specific approach, developers use specific information about the domain and various ad hoc NLP approaches such as stemming and regex (Bird, 2009) to analyze text descriptions and map them to the appropriate entities in the ontology. One recent example of this is (DeBellis, 2023) where stemming and other capabilities of the AllegroGraph Free Text Index (FTI) tool (Franz Inc., 2023) were used to map data collected about Non-Governmental Organizations (NGOs) to an ontology model of the UN Sustainable Development Goals. This approach goes by many names. E.g., in (Vanjulavalli, 2012) this approach is described as *Web Crawling*.

Named Entity Recognition (NER) is a technique provided in NLP libraries such as Spacy (Altinok, 2021) to analyze the syntactic structure of sentences and look for patterns that indicate a phrase is an instance of an entity type such as a Person, Organization, Date, and Currency. This is one of the most promising areas for future work with ontologies (Al-Moslmi, 2020) as the default entities in an NER model can be further refined with domain specific models based on an ontology. This approach has been used in many systems such as (Song, 2019), (Kejriwal, 2022), and (D'Souza, 2022).

Vector embedding refers to the representation of meaning as a vector in an N dimensional conceptual space. Vectors are implemented as a one-dimensional array of hundreds to thousands of floating-point numbers. Vectors can represent the semantics of an individual word, a sentence, a paragraph, or an entire document. When modeling word meaning, each value represents a semantic feature of the word. The values are also normalized so that they are all in the same range, e.g., between 0 (meaning that feature is irrelevant for the text string) to 1 (meaning that vector is as relevant as possible for the text string). E.g., in the vector for the word "apple" the value for the noun feature would be close to 1 and the value for the verb feature would be close to 0 because "apple" is virtually always used as a noun and never as a verb. The word "love" however, would score close to 1 on both the noun and verb feature since it can be used as either a noun or a verb.

In vector embedding the system begins with a default vector for each word. Linear algebra techniques are then used to model interactions among words and to generate higher level meaning of sentences and paragraphs as vectors. The most common introductory example is doing vector subtraction of the vector for man from the vector for king results in the vector for queen. The term embedding refers to the interaction among each word to refine their vectors and to develop a higher-level vector for sentences and paragraphs. The default word vectors go through a series of transformers (Vaswani, 2017) which are matrices that map the vectors for one word to the vectors for other words. Thus, the original default vectors for each word become highly specialized to model each word's specific meaning as *embedded* in a particular text. Examples of knowledge graph semantic search systems that utilize this approach are (Bernstein, 2006) and (Chakraborty, 2019).

1.4.2. *Review of previous RAG systems*

The RAG architecture was first defined in (Lewis, 2020). In that paper, the domain was still general, what was different was that the RAG system was used to answer specific kinds of questions in a more accurate way. The document corpus was a 2018 dump of Wikipedia, and rather than being used for general chat as an LLM, the RAG system was used to answer four specific types of questions: open-domain question answering, abstractive question answering, Jeopardy question generations, and fact verification. On all four types of questions the RAG system outperformed the previous best performance of more general LLMs such as BERT. In initial work this was the focus of RAG, not on a specific domain but on the same general domain as LLMs but focused on answering specific types of questions or analysis. In addition to those described above, early RAG system performed specific tasks such as paraphrasing and summarization (Li, 2022).

Due to the exponential growth in interest in LLMs the RAG architecture was recognized as addressing the issues that prevented LLMs from being used by skilled professionals and that has been the primary focus in the last two years. In all of these systems, the domain knowledge (typically a corpus of documents) is stored in a relational database. Examples include documents for customer support and legal documents (Gao, 2023). The advantages of a RAG system built on a knowledge graph compared to the traditional relational database method are described in section 2.2.

2. Method: knowledge graphs as the knowledge base for RAG

This section describes the fundamental principles of the system. An ontology is utilized to describe the RAG domain and a knowledge graph to store the document corpus.

2.1. *Knowledge Graph RAG Architecture*

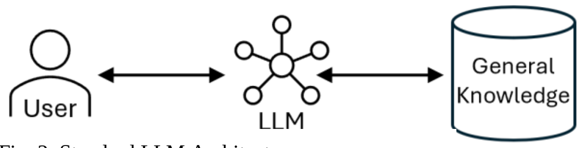


Fig. 2. Standard LLM Architecture

Figure 2 is an abstract description of the standard architecture for an LLM such as ChatGPT. The user interacts with the LLM, asks questions, gives it tasks, and the LLM creates responses based on the large amount of general text it has processed. As described above, this is very useful for many types of problem, however the problems of hallucinations, bias, black-box reasoning, and lack of depth make it unsuitable for many professional users such as doctors. Figure 3 shows a standard RAG architecture. This figure is an adaptation of figure 3 in (Gao, 2023). In the RAG architecture we substitute domain specific documents for the general knowledge that an LLM uses. The LLM is still utilized to create vectors for the user question and the domain specific documents and to generate an answer to the question in natural language. However, the use of domain specific documents allows the RAG system to be utilized where an LLM would not be appropriate.

Figure 4 illustrates the enhanced knowledge graph RAG architecture. This architecture utilizes one integrated knowledge base to store both the vectors created by the LLM as well as the corpus documents and the additional contextual domain knowledge provided by the ontology.

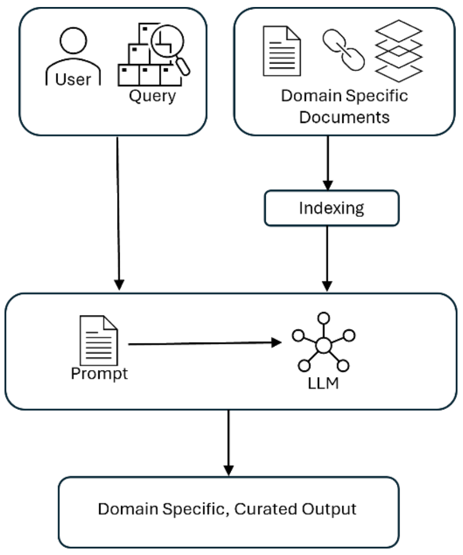


Fig. 3. Standard RAG Architecture

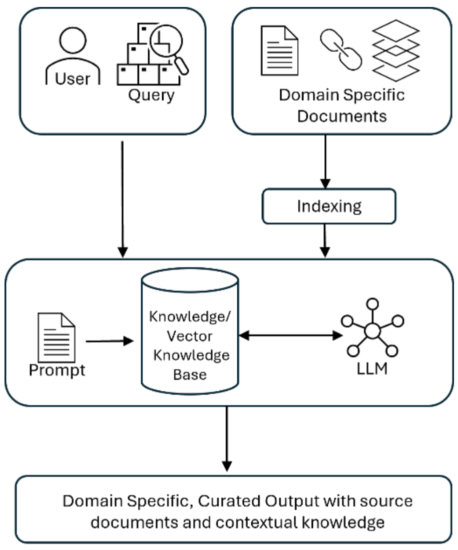

User
Query
Domain Specific Documents
Indexing
Prompt
Knowledge/ Vector Knowledge Base
LLM
Domain Specific, Curated Output with source documents and contextual knowledge

Fig. 4. Knowledge-Graph RAG Architecture

2.2. *Advantages of the Knowledge-Graph RAG Architecture*

This approach to semantic search provides the best solutions to the various architectural options for semantic search described in section 1.4.1:

- NLP User Interface. This approach combines both the benefits of natural language parsing and generation with the graphical query capabilities of knowledge graphs. The previous approaches to natural language understanding of questions described in section 1.4.1.1 were complex to build and maintain. Structured English is unnatural for non-technical users, especially compared with the capabilities of modern LLMs to parse and generate text that is comparable to a human speaker. Generation of query statements from natural language is prone to failure and when it does fail does not fail gracefully. E.g., (Sowa, 2014) has several examples that demonstrate language that is syntactically correct but does not translate appropriately to logic or a query language. This is especially an issue as the system's target users (dental clinicians in India) are primarily not native English speakers. In the competency questions created to test the system the domain expert included several minor grammatical errors common in non-native English speakers. The RAG system (powered by ChatGPT) was able to understand all the competency questions regardless of minor grammatical errors. Structured English and other systems that translate English to query languages would not handle such errors.
- Graphical User Interface. This approach brings all the power of an LLM to parse natural language with no constraints. At the same time, the answers to questions can be provided both in natural language and in a graphical form that allows the user to further explore relevant documents and other knowledge that is relevant to their question by expanding nodes and otherwise manipulating graph views of the knowledge graph as the graphical UIs described in 1.4.1.1.

- Indexing. This approach utilizes one of the modules in the LLM to generate vectors that represent the meaning of text. Vector embedding leverages the flexibility of LLMs and can generate meaning vectors that are highly accurate and on the very small number of cases where they may fail, they fail gracefully. I.e., they simply create a vector that is less accurate. The other approaches to indexing are far more brittle and when they fail far more likely to index either incorrectly or not at all.

The advantages of this knowledge-graph based approach compared to traditional RAG based on a relational database are:

- Knowledge reuse. Building on ontologies leverages the large amount of curated domain knowledge both for horizontal domains such as metadata (e.g., the Dublin Core, BIBO, and Prov vocabularies) and for vertical domains such as dentistry (the Oral Health and Disease vocabulary) and general medicine (the OBO foundry and the National Institute of Health vocabularies).
- Fast, Agile, development. As described in (DeBellis, 2022) semantic technology can enable rapid, Agile development because developers are working at the analysis level rather than the design level. For domains that have highly interconnected data, knowledge graphs and query languages such as SPARQL and Cypher are more flexible, easy to use, and efficient than SQL which again facilitates rapid development that can easily accommodate new requirements.
- Flexible, Agile, maintenance. Semantic models can be modified far easier than relational tables. This makes maintenance and enhancement faster and less error prone (McComb, 2019).
- Improved context. The semantic model brings additional context by a semantic description of the domain. This context can be used to display concepts that are relevant to discovering additional documents or other knowledge sources in the system.
- Graphic retrieval by reformulation. As described in 1.4.1.1 retrieval by reformulation is a paradigm that has been demonstrated to be a good fit for semantic technology and has been shown by many implementations and evidence from cognitive science and experiments with user subjects to be an intuitive and efficient way to search large stores of complex knowledge. Utilizing a knowledge graph as the knowledge base for a RAG system enables implementation of retrieval by reformulation with a graphic user interface. In addition to returning the documents used by the LLM to generate answers, the system returns additional entities in the ontology and can display them in a graph. The user can then further explore this graph and iteratively discover additional relevant knowledge.

3. Results: The current prototype

This section describes the current prototype implementation (called DrMo from the acronym for the initial ontology) of semantic search via Retrieval Augmented Generation based on integration of an LLM and knowledge graph. All the code, Corpus, ontologies, and knowledge graphs are available via an open-source license at (DeBellis, 2024).

3.1. Ontology Development

The starting point for the system was the Dental Restorative Materials Ontology (DrMo). The DrMo system followed best practices by reusing entities from the OBO Oral Health and Disease (OHD) vocabulary as well as several other standard vocabularies for medicine, knowledge organization, and materials engineering. See (Dutta, 2023) for details. The DrMo ontology had no model for documents nor metadata as such a model was not required for the initial project. One of the first steps in this phase was to extend the ontology by adding reusable entities for documents and metadata from Dublin Core (DCMI, 2024), BIBO (DCMI, 2024a), and Prov (Prov W3C Working Group, 2013).

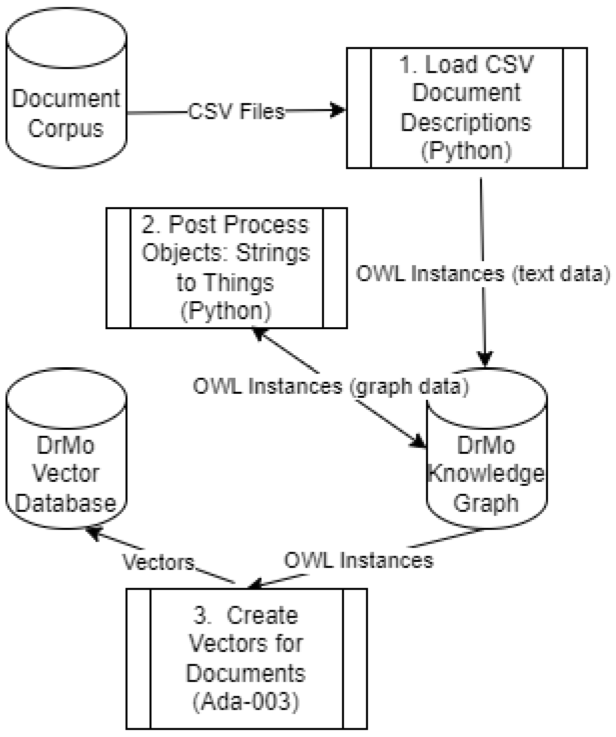
The ontology design was enhanced utilizing the Protégé ontology editor. Protégé is an open-source best in class tool for ontology modeling. However, a graph database was also required to store the large amount of metadata required to model and search the document corpus. The AllegroGraph graph database was selected for the following reasons:

- AllegroGraph is one of the best graph databases in terms of scale, efficiency, and reliability.
- AllegroGraph natively supports the W3C semantic web standards: OWL, RDF/RDFS, and SPARQL.
- The Gruff visual graph browser that is part of the AllegroGraph system provides a predefined tool to visualize and manipulate large complex graphs.
- The AllegroGraph Python client is robust and well documented. Python was the language of choice because of the many data science and machine learning libraries it supports. In addition, interns from the data science programs at the University of California, Berkeley and the University of Wisconsin, Madison were used to supplement the team and they are most familiar with Python.

3.2. Corpus creation

The project began in January of 2024 as a Semantic Search project with the specific goal of addressing the issues of fraudulent and defective dental material products in India and other developing nations as well as providing a tool for general education of such clinicians. In addition to expanding the DrMo ontology to model documents and metadata, a document corpus was begun. Criteria for the corpus were defined and the domain expert executed searches on the relevant search sites. Matching document metadata was saved in CSV files which were then loaded into the knowledge graph with a Python function. The search criteria for the initial corpus are:

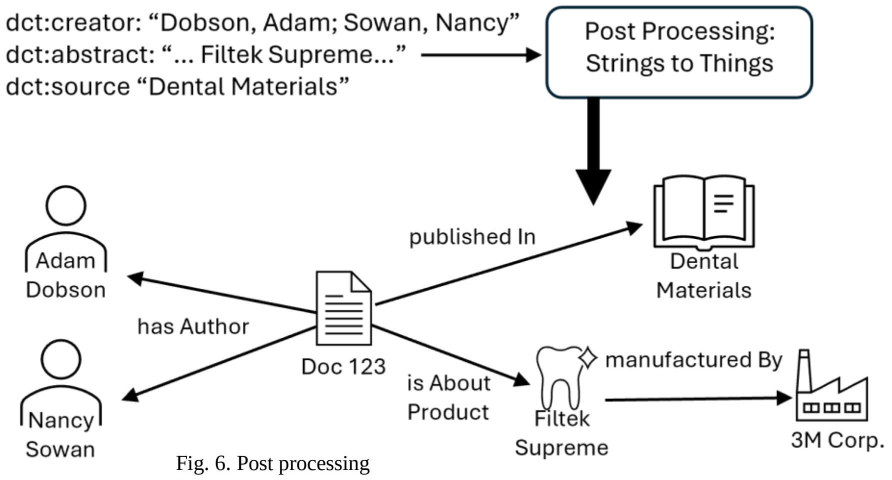
- Articles from the journals Dental Materials and Cochrane review articles. The domain expert determined these to be the best initial sources for information that would be most useful to the target users.
- Articles from the last five years. This was done to focus on the most up to date information. This was critical due to the dynamic nature of the domain. This also is one of the ways the corpus knowledge base outperforms an LLM such as ChatGPT. Due to the intensive effort required to train an LLM, an LLM typically does not contain data from the most recent months or even years.
- Primary keywords for searching are *Resin Based Composite*, *Glass Ionomer Cement*, *Resin-modified Glass Ionomer Cement*, and *Dental Amalgam*. *Direct Dental Restorative Material* was added as a required phrase with each keyword.


Document Corpus
CSV Files
1. Load CSV Document Descriptions (Python)
2. Post Process Objects: Strings to Things (Python)
OWL Instances (text data)
OWL Instances (graph data)
DrMo Vector Database
DrMo Knowledge Graph
Vectors
OWL Instances
3. Create Vectors for Documents (Ada-003)
Fig. 5. Data pipeline

- The Elsevier site is the primary tool to search for relevant articles. Metadata for articles that are deemed appropriate for the corpus are downloaded and saved in CSV files. Relevant metadata includes authors, title, url, abstract, date, and keywords.
- We also manually added documents that were deemed to be essential for our users. E.g., manuals and product safety data sheets from manufacturers such as 3M, specific articles on fraudulent and defective dental material products, etc.

3.3. Data pipeline

Figure 5 illustrates our data pipeline. CSV files created by the domain expert are passed to the technical team. The CSV files are transformed so each heading is replaced by the IRI for the property that each column is meant to populate. This simplifies the complexity of the Python function to load the CSV files. The Python loader processes the header by validating that each IRI maps to an existing data property. If a header doesn't map to a data property, then the loader aborts before loading any data to avoid corrupt data. The metadata from the CSV files is loaded into data properties. Each row in the CSV file corresponds to a new instance of the Document class, usually an instance of Journal Article, a subclass of Document.


dct:creator: "Dobson, Adam; Sowan, Nancy"
dct:abstract: "... Filtek Supreme..."
dct:source "Dental Materials"
Post Processing: Strings to Things
published In
Dental Materials
Adam Dobson
has Author
Doc 123
Nancy Sowan
is About Product
Filtek Supreme
manufactured By
3M Corp.
Fig. 6. Post processing

3.3.1. Post processing from strings to things

The next step in the pipeline is to post process the new objects. Post processing converts many of the data properties into one or more new or existing objects and an appropriate value for a document object property. This process is often referred to as “strings to things” (Singhal, 2012) and is illustrated in fig. 6. In that figure we see examples of data properties from Dublin Core: creator, abstract, and source, transformed into objects and object property values. The creator strings are parsed and for each name the system searches for an existing instance of the Person class with the same first and last name. If no such object exists, it is created. Then the new or existing Person object is made a value of the has Author object property for the Document.

Similarly, the source data property which records where a document was published is transformed to an object property. In this case there are currently only a handful of sources, so they are defined as part of the ontology. The abstract is processed using the AllegroGraph Free Text Index (FTI) feature which facilitates searching for close matches for names or phrases that correspond to the rdfs:label of any entity in the ontology. In addition, skos:altLabel and skos:prefLabel are utilized to define synonyms for entities. If there is a match, then a value is asserted on the is About annotation property. In this case an annotation property is used because it is possible for a document to refer to a class or even property as well as an instance. Initially is About was an object property and puns were utilized when it was appropriate to connect a document with a class or property. However, as puns began to proliferate the team decided that an annotation property was simpler and alleviated the complexity of dealing with large numbers of puns. The loss of reasoning power sacrificed by using an annotation rather than an object property is minimal as the only reasoning that was utilized was assertion of inverse values. Since the values are all asserted in Python code it is trivial to assert the inverse at the same time as the normal value. A sub-property of is About is: is About Product which is defined to record specific Dental Material Products that a Document refers to. Since these links are especially of interest it was considered worthwhile to have a special property for them.

The original plan was to use Spacy and Named Entity Recognition for this type of post processing. That may be a future enhancement. However, for the current data most of the required processing can be performed adequately with Python string manipulation functions and the AllegroGraph Free Text Index. A major reason for this is that with a few exceptions, the data that is post processed is not narrative data, but simple strings and Spacy is primarily designed to analyze narrative text. For example, the author string parser was first modeled using the Person entity recognizer in Spacy. However, Spacy failed to correctly parse simple strings such as “Last1, First1; Last2, First2” and these could easily be processed with Python functions such as split. In the future, the team plans to utilize NER and Spacy to do a more complete processing of the is About and possibly other values.

3.3.2. Loading and processing documents

The final step in post processing is document processing. In this step the document that is the target of the uri in the metadata of each document is retrieved. To provide the best information for the RAG system, it was required to generate vectors for the full documents as well as the abstracts. Experiments with vector generation and discussions with developers at Franz indicated that generating vectors for small sections of text is preferable to generating one vector for the entire document. Creating vectors for sections of a document provides much more detailed semantics for the corpus. In addition, this is a better design decision in terms of usability. One of the goals of the project is to enable busy clinicians to find the specific information they need quickly and easily. Providing a source that is an entire document requires the clinician to process what are often dense complex papers to find the specific information they need. Dividing the documents into smaller chunks and generating vectors for each chunk enables the system to focus the attention of the clinician directly on the specific section of the document that addresses their question. The graphical user interface then allows the users to easily expand their focus to the entire document if so desired.

The natural structure for the documents is to simply follow the structure that articles as well as most materials from manufacturers already provide. I.e., a document can be thought of as a tree where the top node is the document, the next level are the first level sections. The branches of each section are their sub-sections and so on with any sub-sections of those sections. This is the approach we utilize with the object property has Sub Section. This property creates a part of hierarchy starting from the document to sections and sub-sections. To implement this an HTML Python library called Beautiful Soup (Richardson, 2007) was used. This library can parse documents into sections by finding tags such as 'h2' and 'h3'.

The vast majority of the documents in the corpus are available in HTML format. However, documents such as product manuals are only available as PDFs. The PDF format is a print format and not designed to be manipulated programmatically. Hence, to parse PDF documents a tool called PDFMiner (Shinyama, 2019) is used. PDFMiner recursively searches through a directory and returns a list of PDF paths. For each PDF file, it generates an HTML file path by replacing the directory and file extension. It then converts each PDF into HTML format, writing the output directly to the corresponding HTML file. Once the PDF has been transformed to HTML the HTML parser can be used.

The prototype currently supports only HTML and PDF format documents, but this is simply a question of not investing effort until it is required. Those are the only formats required for the corpus to date. In the future, other formats such as Google and Microsoft Office may be required, and the system will be enhanced to process such documents when needed. As with the document processing done to date, this is essentially a solved problem where the work is integration of reusable libraries rather than developing a large amount of custom code.

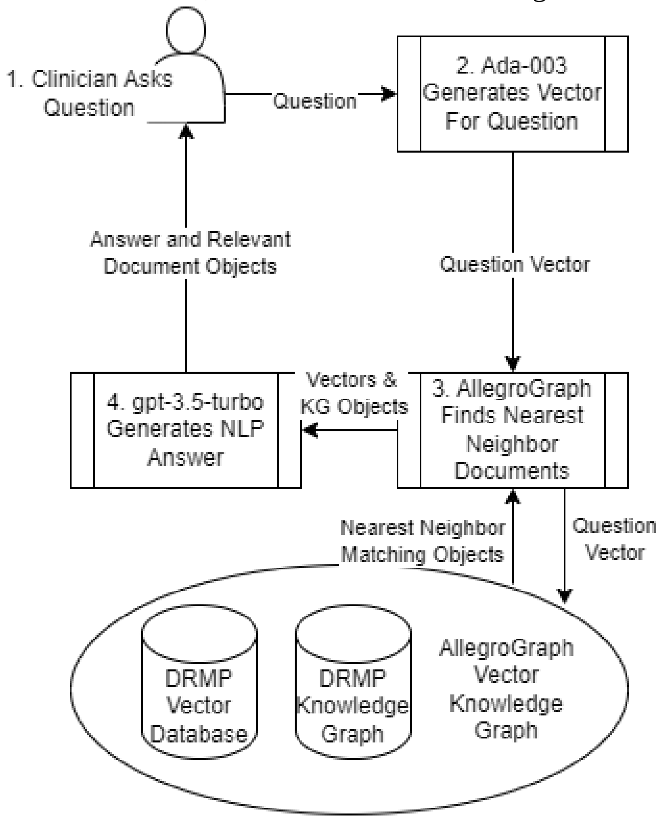

1. Clinician Asks Question
Question
2. Ada-003 Generates Vector For Question
Answer and Relevant Document Objects
Question Vector
4. gpt-3.5-turbo Generates NLP Answer
Vectors & KG Objects
3. AllegroGraph Finds Nearest Neighbor Documents
Nearest Neighbor Matching Objects
Question Vector
DRMP Vector Database
DRMP Knowledge Graph
AllegroGraph Vector Knowledge Graph

Fig. 7. Run time architecture

3.3.3. *Generating vectors*

The final step in the data pipeline is vector generation. This is done with each string value of the dct:abstract property for a document and for the text property for a document section. This is accomplished with a general-purpose Franz function called agtool that is executed at the Linux shell (Franz Inc., 2024). To generate vectors, one passes the appropriate parameters to agtool such as the knowledge graph, the properties containing text that are to be assigned vectors, and the Open AI API key. Agtool then communicates with the Ada 003 model via the Open AI API and generates a vector for each text string that is the value of one of the data properties passed to it. One of the features of the latest version of AllegroGraph is that it stores vectors in an efficient database that is tightly coupled with the ontology. To the developer it seems as if there is simply one repository that contains both the knowledge graph and the vector embeddings.

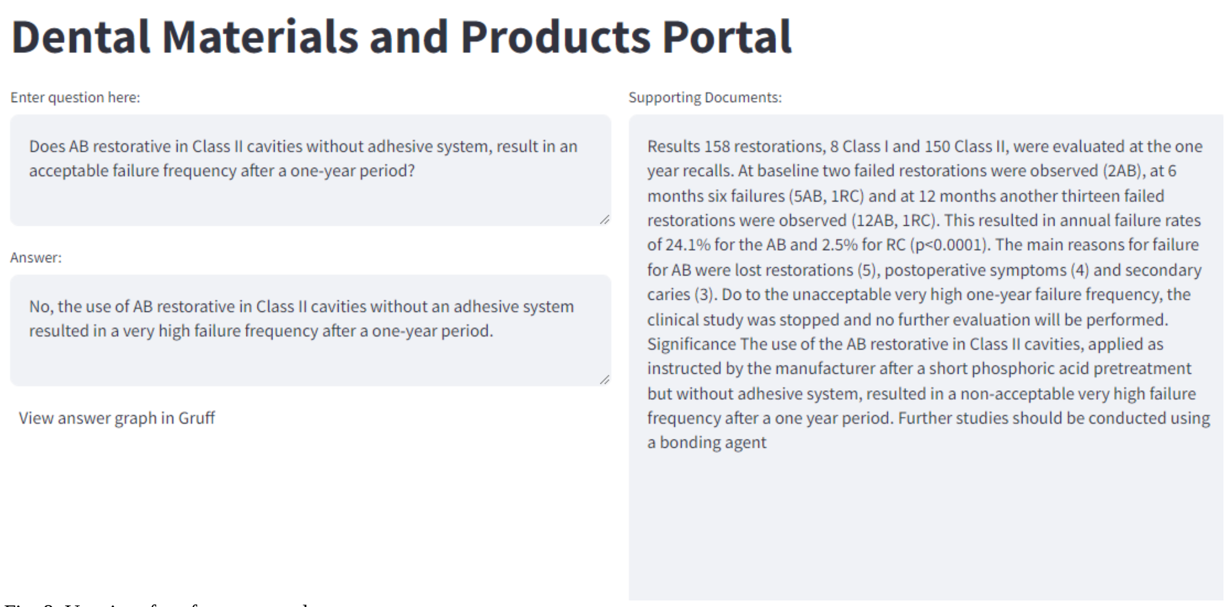
3.4. Run-time architecture

Figure 7 illustrates the run-time architecture of the system. The numbers in each description correspond to the order that each event occurs at in a user interaction. To begin the user is presented with a simple user interface defined using an open-source Python tool called Streamlit (Snowflake Inc, 2024). See fig. 8 for an example of the UI. The user types in a question to the top window in the UI. This question is then used with a predefined template to generate a SPARQL query. An example generated SPARQL query is:

```
SELECT *
WHERE {bind("Does AB restorative in Class II cavities without adhesive system, result
                  in an acceptable failure frequency after a one-year period?" as ?query)
     (?response ?score ?citation ?content) llm:askMyDocuments (?query "drmo" 2 0.8).
      ?citation drmo:hasAuthor ?author;
              drmo:publishedIn ?publication.
OPTIONAL {?citation drmo:isSubSectionOf ?document.}
OPTIONAL {?citation drmo:isAbout ?entity.}
OPTIONAL {?citation drmo:isAboutProduct ?product.}}
```

Everything in the query is predefined in the template except the actual question which is passed from the streamlit question box into the SPARQL query. The additional parameters in the query are examples of the additional context that integrating an LLM with a knowledge graph can provide. When the SPARQL query executes, the question is passed to the Ada 003 Open AI model which returns a vector that represents the meaning of the question. AllegroGraph then compares this vector with the existing vectors in the vector/knowledge graph using the cosine nearest neighbor function to find vectors that are within the threshold semantic distance of the question. Any vectors in the knowledge graph that are greater than the threshold defined in the query (.8 in this example) are considered relevant. If no such vectors exist, the system returns a predefined answer that there is not enough information in the knowledge graph to answer that question. This is an example of how the RAG architecture eliminates hallucinations. AllegroGraph then passes the relevant vectors from the corpus along with the question vector to gpt-3.5-turbo. GPT uses these vectors and only these vectors to return an answer. The answer and the relevant text excerpts (abstracts or segments of a document) are returned along with other relevant entities from the knowledge graph.

3.5. An example use case


Dental Materials and Products Portal

Enter question here:

Does AB restorative in Class II cavities without adhesive system, result in an acceptable failure frequency after a one-year period?

Answer:

No, the use of AB restorative in Class II cavities without an adhesive system resulted in a very high failure frequency after a one-year period.

View answer graph in Gruff

Supporting Documents:

Results 158 restorations, 8 Class I and 150 Class II, were evaluated at the one year recalls. At baseline two failed restorations were observed (2AB), at 6 months six failures (5AB, 1RC) and at 12 months another thirteen failed restorations were observed (12AB, 1RC). This resulted in annual failure rates of 24.1% for the AB and 2.5% for RC (p<0.0001). The main reasons for failure for AB were lost restorations (5), postoperative symptoms (4) and secondary caries (3). Do to the unacceptable very high one-year failure frequency, the clinical study was stopped and no further evaluation will be performed. Significance The use of the AB restorative in Class II cavities, applied as instructed by the manufacturer after a short phosphoric acid pretreatment but without adhesive system, resulted in a non-acceptable very high failure frequency after a one year period. Further studies should be conducted using a bonding agent

Fig. 8. User interface for an example use case

Figure 8 shows the user interface for an example use case. The user has entered a question in the box in the upper left corner. The system generated a SPARQL query that found the relevant objects using the cosine nearest neighbor function. The system found a paper that provided the correct answer and ChatGPT used the objects from the knowledge graph to generate the natural language answer. In addition, the text for the matching object is shown in the box on the right. The answer to the clinician's question was definitive: the use of AB restorative in the situation the clinician queried resulted in a high rate of failure. The answer to the same question from ChatGPT was much more lengthy but non-committal. With some detail omitted for brevity, the response from ChatGPT to this same question was:

"The success of a Class II cavity restoration using an amalgam or composite material without an adhesive system depends on various factors including the size and location of the cavity, the material used,… Ultimately, whether a restoration without an adhesive system results in an acceptable failure frequency after one year would depend on the specific clinical circumstances, the skill of the practitioner, and the patient's oral health and habits. It's essential for dental professionals to consider the best available evidence, their clinical judgment, and the patient's individual needs when selecting materials and techniques for cavity restorations."

The user then selects the "View answer graph in Gruff" link below the answer box on the left. This launches the Gruff knowledge graph browser with all the objects that were returned in the SPARQL query in fig. 9. In fig. 9, the user is hovering the mouse over one of the nodes for the author's name. The user selects that node and chooses to show any triples currently not visible that have the author as the object and have the predicate has Author. This results in the revised graph shown in fig. 10. This is an example of the retrieval by reformulation paradigm described in section 1.4.1.1 using a graphic rather than a forms-based user interface. The user is able to refine and elaborate their query based on information retrieved from the initial query.

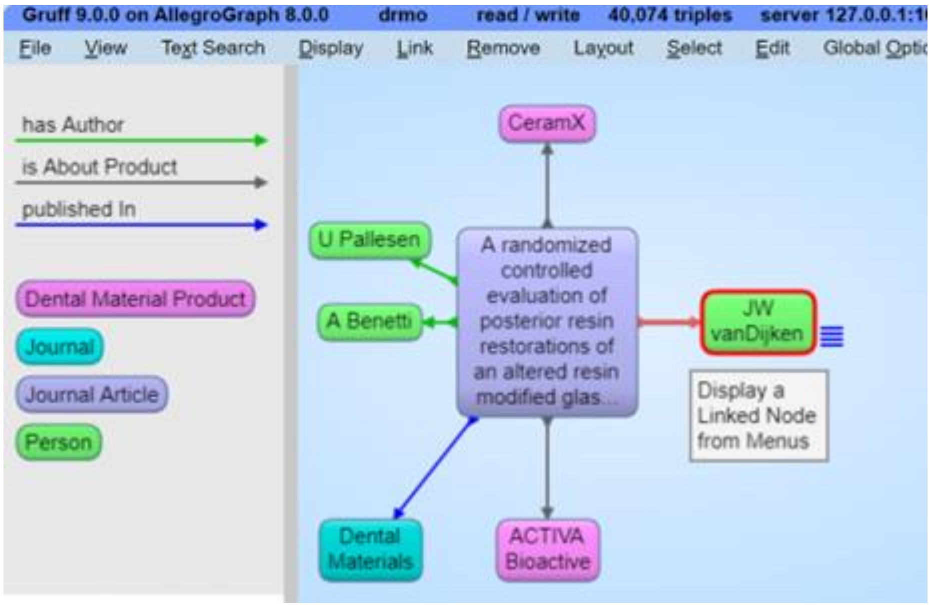

Gruff 9.0.0 on AllegroGraph 8.0.0 drmo read / write 40,074 triples server 127.0.0.1:1
File View Text Search Display Link Remove Layout Select Edit Global Optic
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
U Pallesen
A Benetti
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
JW vanDijken
Display a Linked Node from Menus
Dental Materials
ACTIVA Bioactive

Fig. 9. Graph of objects returned from question in fig. 8

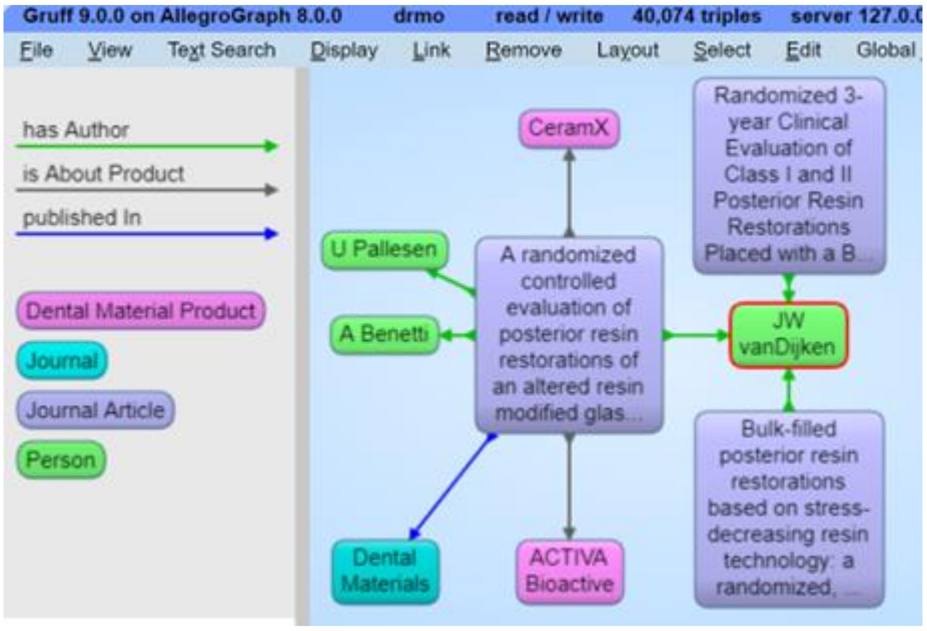

Gruff 9.0.0 on AllegroGraph 8.0.0 drmo read / write 40,074 triples server 127.0.0
File View Text Search Display Link Remove Layout Select Edit Global
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
Randomized 3-year Clinical Evaluation of Class I and II Posterior Resin Restorations Placed with a B...
U Pallesen
A Benetti
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
JW vanDijken
Bulk-filled posterior resin restorations based on stress-decreasing resin technology: a randomized, ...
Dental Materials
ACTIVA Bioactive

Fig. 10. Revised graph after user expands an author node

3.6. Summary of work to date

The current DrMo system contains 828 documents and has been tested with 50 different competency questions created by the domain expert on the specific topic of dental material products for direct restoration. On all 50 questions, the system was superior to the answers generated by ChatGPT, often in dramatic ways as in this example where ChatGPT provided no definitive answer, but DrMo did. Development began in January and continues to the present. The current system has not been evaluated by actual clinicians, only by the domain expert on our team (Dr. Dutta). In parallel with the technical work, we have been conducting a survey of clinicians in India to measure their receptiveness and trust toward using LLMs, specifically ChatGPT, in a clinical setting. This pool of clinicians will be our initial pool for real users to evaluate the next version of the system.

The one major failure is that the team had planned on utilizing Named Entity Recognition (NER) as implemented in Spacy to develop sophisticated recognition of entities in the ontology that are referenced in the document text. Given the other technologies that were required, Spacy proved a bit more of a learning curve than anticipated and using more basic NLP techniques such as stemming, and wildcard matching provided acceptable results. Another interesting issue is that to date the team has found that specific documents we hoped to incorporate from the beginning, e.g., guidelines and bulletins on how to spot fraudulent and defective dental material products are difficult to find or simply may not exist for the Indian market. One option that has been considered is if they don't exist then just write them. E.g., create a volunteer group of dental experts in India and across the globe who may be interested in writing such documents. An intriguing option is to team with leading manufacturers who have a vested interest in the topic as well.

We plan to greatly increase our corpus in the next few months and also seek out clinicians to do real world tests comparing the performance and usability of DrMo with ChatGPT. The major technical barriers we need to overcome for this are making the system accessible via the Internet and performance. The current system must be run on a local machine. It should be straight forward to move the system to the Internet, that was one of the reasons we chose the Streamlit tool for our user interface. However, the current performance of the system is only adequate. The primary issue is that the Open AI API can sometimes take a while to return a response. Also, while the Streamlit tool is very easy to use, it can also be somewhat slow, although this is mostly an issue for the first use, after the first query where information is cached performance significantly improves. After we deploy the system on the Internet, we will test its performance and investigate ways to improve it if needed.

4. Discussion

4.1. Critical evaluation against related work

The RAG architecture is one of the most investigated and discussed at present. The achievements of Large Language Models such as ChatGPT have captured the imagination of the world. At the same time the four issues described in section 1.2 prohibit LLMs from being utilized for many professional applications. RAG provides an architecture that solves these issues.

However, while RAG is one of the hottest topics among developers at this time, there are currently few implementations of RAG utilizing an ontology that we could find on Google Scholar. In Google Scholar, the one paper we could find on using ontologies to implement RAG to solve a domain problem was (Xu, 2024) which focuses on a very esoteric topic related to ancient Chinese literature. The current literature focuses primarily on ways that LLMs can help automate the creation of ontologies (Krishna, 2024), (Vamsi, 2023). This includes using a RAG architecture to help automate the development of ontologies (Sabrina Toro, 2024).

This is one of those examples, where industry seems to be ahead of academia. AllegroGraph is one of the first vendors to offer features in their product specifically to integrate knowledge graphs and LLMs. In addition, at least two other leading vendors of Semantic Web technology have created similar product offerings: NebulaGraph (NebulaGraph, 2023) and OntoText (OntoText, 2024). All three vendors have emphasized the RAG architecture as one of the best opportunities to integrate the technologies. While these vendors have been aggressive in this space and provide various demos of their technology, this project is to the best of our knowledge one of the first to utilize a real-world domain ontology to solve an important and difficult problem.

4.2. Planned Next Steps

The planned next steps for this project are:

1. Host the system on the Internet and perform performance testing and improvement. This will enable testing with actual domain experts other than the domain expert who is a member of our team. As described in 3.6, we have concurrent to this technical work been performing surveys of clinicians in India regarding their use and trust of ChatGPT. These survey participants will be our initial users who will evaluate the system for usability, accuracy on arbitrary questions, and usability of additional features to browse the knowledge graph. Our plan is to first perform testing with a small (3-5) group of users, to revise the UI based on feedback from the initial user group and then to roll out that revised version to a larger group (30 or more) of clinicians.
2. Enhance the capabilities to recognize ontology entities in the corpus text utilizing the Python Spacy library's Named Entity Recognition capabilities. Spacy is capable of recognizing complex concepts, far more than just recognizing individual words or short phrases, it can recognize various sections of text distributed across one or more sentences as examples of entities such as relations. This is similar to a great deal of work currently being done by other teams to partially automate the development of ontologies using LLMs such as (Krishna, 2024), (Vamsi, 2023), (D'Souza 2022), and (Sabrina Toro, 2024). We may leverage some of this research or develop our own simpler capabilities in Spacy. This will enhance the connectivity of the corpus documents to relevant entities in the ontology.
3. Redesign the ontology. We plan to redesign the ontology to better leverage the OBO Oral Health and Disease ontology. We learned of the OHD ontology well into our development of the original DrMo ontology and as a result have been piece meal adding classes and properties from OHD and other OBO ontologies. This project has made it clear that a new ontology is required, one that starts with the entire OHD ontology as a baseline and then adds the needed entities for dental restoration materials and products as well as the entities to model documents.
4. Create vectors for the complete ontology. To date we have focused on creating vectors only for corpus documents. However, the ontology itself is an excellent store of knowledge and we have already had some good preliminary results constructing vectors for some of the labels of ontology entities. This is another reason to rework the ontology. We plan to be much more rigorous including text definitions, alt labels for synonyms, and other text that can be vectored and utilized by the LLM. Another possibility is to utilize one of the more intuitive output formats such as Manchester Syntax (Horridge, 2012) to create vectors that represent the semantics of the ontology. One other possibility is to write a simple NLP

generator that utilizes the ability of Spacy to understand tense, plurals, etc. This would be a custom tool to generate readable English from the ontology and create vectors for that English version of the ontology.

4.3. Conclusion: Knowledge Graphs complement Large Language Models

John Sowa (Sowa, 2011) offers the conceptual framework for understanding the essential differences between the semantic and LLM approaches. LLMs are based on induction and probabilistic reasoning and semantic technology is based on deduction and logical reasoning. Humans use both types of reasoning every day, and the future of AI will include both. This approach of utilizing semantic and machine learning systems in a synergistic way that exploits the benefits of both currently has significant traction in industry. The state of the market for enterprise data architectures in industry are data fabrics and data mesh (DeBellis, 2023). Both architectures treat machine learning and semantic technology as essential technologies for utilizing enterprise data. These architectures use semantic technology to define data catalogs, metadata, and enterprise data models (McComb, 2019), (Dehghani, 2020). They use LLMs and other machine learning technology to analyze big data stored in data lakes, to automate tasks such as customer support, and to predict customer behavior and other events important to the business (Fortune Business Impact, 2023), (DeBellis, 2023).

This project offers evidence for the power of another approach: using semantic and machine learning technology together in an integrated application. In this way the power for induction and statistical reasoning can be supplemented by the power of domain models, explanation, and logical reasoning. Each technology complements the other. In this project, the use of an ontology and knowledge graph eliminates the problems of hallucinations and black-box reasoning. The use of an LLM enables very reliable and fairly easy to implement natural language processing for both input and output and captures meaning via vector embedding. It is our hope that this project may help pave the way for future applications that successfully integrate semantic and machine learning technology. Both for RAG architectures and other hybrid architectures. The potential applications of this integration are virtually limitless. There are countless domains: healthcare, genetic engineering, legal, security, and financial services to name just a few, where experts perform analysis that could be greatly assisted by an NLP powered assistant but where the problems with traditional LLMs inhibit their use. This integrated approach leverages the power of semantic technology to deliver answers based on evidence and explicit knowledge while leveraging the unparalleled capability of LLMs to interface with the user in natural language.

5. Acknowledgements

Thanks to Bob Neches for feedback on drafts of this paper. Thanks to Franz Inc. for providing a beta version of their LLM integration and general support. Special thanks to Jans Aasman, Yan Xu, and David Margolies. This work was conducted using Protégé. Thanks to the Protégé support mail list, especially Lorenz Buehmann. Thanks to Sara Eginova and Jacob Gino and the UC Berkeley Data Science department who provided interns to help with the project. Thanks to the Oral Health and Disease (OHD) ontology team, especially Bill Duncan.

6. References

Al-Moslmi, T., Ocaña, M. G., Opdahl, A. L., & Veres, C. (2020). Named Entity Extraction for Knowledge Graphs: A Literature Overview. *IEEE Access, 8*, 32862-32881. doi:10.1109/ACCESS.2020.2973928

Altinok, D. (2021). *Mastering spaCy*. Birmingham: Pact Publishing.

Benhocine, K., & Hansali, A. Z.-G. (2022). Towards an automatic SPARQL query generation from ontology competency questions. *International Journal of Computers and Applications, 44*(10), 971–980. Retrieved from https://www.tandfonline.com/doi/full/10.1080/1206212X.2022.2031722

Berners-Lee, T., Hendler , J., & Lassila , W. O. (2001, May 17). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*.

Bernstein, A., & Kaufmann, E. (2006). Gino - Guided Input Natural Language Ontology. *Proceedings of the 5th International Semantic Web Conference (ISWC 2006),* (pp. 144-157). Athens, Georgia, USA.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* Sebastapol, CA, USA: O'Reilly.

Cernile, G., et. al. (2021, Jan). Network graph representation of COVID-19 scientific publications to aid knowledge discovery. *BMJ Health Care Inform, 28*(1).

Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., & Fischer, A. (2019, July). *Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs*. Retrieved from arxiv.org: https://arxiv.org/abs/1907.09361

D'Souza, J., & Auer, S. (2022). Computer Science Named Entity Recognition in the Open Research Knowledge Graph (CADL 2022). *International Conference on Asian Digital Libraries* (pp. 35-45). Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-3-031-21756-2_3

DeBellis, M. (2023). Semantic Web Technologies: Latest Industrial Applications. In A. Patel, N. C. Debnath, & B. Bhushan (Eds.), *Semantic Web Technologies: Research and Applications* (pp. 43-72). Boca Raton, FL, USA: CRC Press Taylor and Francis Group. doi:10.1201/9781003309420-3

DeBellis, M. (2024, April 17). *github.com*. Retrieved from DrMO_Docs: https://github.com/mdebellis/DrMO_Docs

DeBellis, M., & Dutta, B. (2022, October 5). From ontology to knowledge graph with agile methods: the case of COVID-19 CODO knowledge graph. *International Journal of Web Information Systems, 18*(5/6). doi:https://doi.org/10.1108/IJWIS-03-2022-0047

DeBellis, M., & Dutta, B. (2022). Semantic Data Science in the COVID-19 Pandemic. In A. Patel, N. Debnath, & B. Bhusan (Eds.), *Data Science with Semantic Technologies: Theory, Practice, and Applications* (pp. 393-427). Hoboken, NJ, USA: Wiley.

DeBellis, M., Foss, C., Guo, P., Jyothi, T., Kron, K., & Suresh, V. (2023). DaanKG: An Ontology Model of the UN Sustainable Development Goals to Facilitate and Improve Corporate Social Responsibility. *Workshop on Ontologies and Social Responsibility FOIS 2023.* Sherbrook, Quebec Ontario.

DeBellis, M., Pinera, L., & Connor, C. (2023). Interoperability Frameworks: Data Fabric and Data Mesh Architectures. In D. N. Debnath, & D. Patel (Eds.), *Data Science with Semantic Technologies.* Boca Raton, Florida, USA: CRC Press.

Dehghani, Z. (2020, December 3). *Data Mesh Principles and Logical Architecture*. (https://martinfowler.com/) Retrieved 11 20, 2023, from https://martinfowler.com/articles/data-mesh-principles.html

The Dublin Core™ Metadata Initiative (DCMI). (2024, April 10). *DCMI Metadata Terms*. Retrieved April 16, 2024, from dublincore.org: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Dublin Core Metadata Initiative (DCMI). (2024, April 10). *The Bibliographic Ontology (BIBO)*. Retrieved 4 16, 2024, from dublincore.org: https://www.dublincore.org/specifications/bibo/

Duncan, W., T., T., Haendel, M., & et., a. (2020). Structuring, reuse and analysis of electronic dental data using the Oral Health and Disease Ontology. *J Biomed Semant, 11*(8). doi:https://doi.org/10.1186/s13326-020-00222-0

Dutta, N., & DeBellis, M. (2023). Dental Restorative Material Ontology (DrMO). *Formal Ontologies in Information Systems (FOIS) (2023).* Sherbrooke, Qc, Canada.

Fortune Business Impact. (2023). *Tecjnology / Data Fabrics*. (Fortune Business Impact) Retrieved October 5, 2023, from www.fortunebusinessinsights.com: https://www.fortunebusinessinsights.com/data-fabric-market-105979

Franz Inc. (2023, March 22). *AllegroGraph Freetext Indexing*. Retrieved April 27, 2023, from https://franz.com/agraph/support/documentation/current/text-index.html

Franz Inc. (2024, April 1). *Embedding by creating a VectorStore*. Retrieved April 17, 2024, from franz.com: https://franz.com/agraph/support/documentation/current/llm-examples.html#embedding-example

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., . . . Wang, H. (2023, December 18). *Retrieval-Augmented Generation for Large Language Models: A Survey*. Retrieved April 15, 2024, from paperswithcode.com: https://paperswithcode.com/paper/retrieval-augmented-generation-for-large

Horridge, M., & Patel-Schneider, P. F. (2012, December 11). *OWL 2 Web Ontology Language Manchester Syntax (Second Edition)*. Retrieved June 6, 2024, from w3.org: https://www.w3.org/TR/owl2-manchester-syntax/

Kejriwal, M., & Szekely, P. (2022). Knowledge Graphs for Social Good: An Entity-Centric Search Engine for the Human Trafficking Domain. *IEEE Transactions on Big Data, 8*(3), 592-606. Retrieved from https://ieeexplore.ieee.org/abstract/document/8068229

Krishna Kommineni, V., König-Ries, B., & Samuel, S. (2024, March 13). *From human experts to machines: An LLM supported approach to ontology and knowledge graph construction*. Retrieved 4 18, 2024, from arxiv.org: https://arxiv.org/abs/2403.08345

Koutsomitropoulos, D. A., Domenech, R. B., & Solomou, G. D. (2011). A Structured Semantic Query Interface for Reasoning-Based Search and Retrieval. In G. e. Antoniou, *The Semantic Web: Research and Applications, ESWC 2011*. Springer, Berlin, Heidelberg: Springer. doi:https://doi.org/10.1007/978-3-642-21034-1_2

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of the 34th International Conference on Neural Information Processing Systems* (pp. 9459–9474). Vancouver, Ontaria, Canada: Curran Associates Inc.

Li, H., Su, Y., Cai, D., Wang, Y., & Lemao, L. (2022, February 2). *A Survey on Retrieval-Augmented Text Generation*. Retrieved April 15, 2024, from semanticscholar.org/: https://www.semanticscholar.org/paper/A-Survey-on-Retrieval-Augmented-Text-Generation-Li-Su/e6770e3f5e74210c6863aaeed527ac4c1da419d7

Liang, S., Stockinger, K., & de Farias, T. (2021, January 6). Querying knowledge graphs in natural language. *J Big Data, 8*(3). doi:https://doi.org/10.1186/s40537-020-00383-w

McComb, D. (2019). *The Data-Centric Revolution: Restoring Sanity to Enterprise Information Systems.* Sedona, AZ, USA: Technics Publications.

MHRA. (17, October, 2014). *Regulator warns dentists about the dangers of buying and using counterfeit and unapproved dental equipment.* London, UK: MHRA Press Release.

NebulaGraph. (2023, 9 6). *NebulaGraph Launches Industry-First Graph RAG: Retrieval-Augmented Generation with LLM Based on Knowledge Graphs*. Retrieved April 17, 2024, from nebula-graph.io/: https://www.nebula-graph.io/posts/graph-RAG

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., & Taylor, J. (2019, August). Industry-Scale Knowledge Graphs: Lessons and Challenges. *Communications of the ACM, 62*(8). Retrieved from https://tinyurl.com/ACMKnowledgeGraphs

OntoText. (2024). *What is Graph RAG?* Retrieved April 17, 2024, from ontotext.com: https://www.ontotext.com/knowledgehub/fundamentals/what-is-graph-rag/

PoolParty. (2016). *Healthdirect Australia Uses Poolparty Semantic Suite Technology As The Basis*. Retrieved April 12, 2024, from poolparty.biz: https://www.poolparty.biz/healthcare-information-system/

Price, R., Ferracane, J., Darvell, B., & Roulet, J. (March, 2022). Caveat emptor when purchasing dental products online. *J Am Dent Assoc, 153*(3), 196-199. doi:doi::0.1016/j.adaj.2021.10.013

Prov W3C Working Group. (2013, April). *PROV-Overview*. (P. Groth, L. Moreau, Editors, & W3C) Retrieved May 8, 2023, from www.w3.org/: https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

Rahman, M., & Roy, C. (2023, September). A Systematic Review of Automated Query Reformulations in Source Code Search. *ACM Transaction on Software Engineeering and Methodology, 32*(6), 1-79. doi:https://doi.org/10.1145/3607179

Richardson, L. (2007, April). *Beautiful Soup Documentation*. Retrieved April 17, 2024, from crummy.com: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Rieh, S. Y., & Hong, X. (2006). Analysis of multiple query reformulations on the web: The interactive information retrieval context. *Information Processing & Management, 42*(3), 751-768. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S030645730500066X

Rutter, S., Ford, N., & Clough, P. A. (2014). How do children reformulate their search queries? *Proceedings of ISIC, the Information Behaviour Conference. 20.* Leeds, UK: University of Sheffield. Retrieved from https://www.informationr.net/ir/20-1/isic2/isic31.html#Hua09

Saba, W. S. (2023). Stochastic LLMs do not Understand Language: Towards Symbolic, Explainable and Ontologically Based LLMs. *International Conference on Conceptual Modeling* (pp. 3–19). Springer, Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-031-47262-6_1

Sabrina Toro and Anna V Anagnostopoulos and Sue Bello and Kai Blumberg and Rhiannon Cameron and Leigh Carmody and Alexander D Diehl and Damion Dooley and William Duncan and Petra Fey and Pascale Gaudet and Nomi L Harris and Marcin Joachimiak and Leila Kia. (2024). *Dynamic Retrieval Augmented Generation of Ontologies using Artificial Intelligence (DRAGON-AI).* Retrieved June 6, 2024, from arxiv.org: https://arxiv.org/abs/2312.10904

Shinyama, Y. (2019, Nov 25). *PDFMiner Project Description*. Retrieved April 17, 2024, from pypi.org: https://pypi.org/project/pdfminer/

Singhal, A. (2012, May 16). *Introducing the Knowledge Graph: things, not strings*. (Google) Retrieved May 8, 2023, from https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

Snowflake Inc. (2024). *Streamlit: A faster way to build and share data apps*. Retrieved April 17, 2024, from streamlit.io: https://streamlit.io/

Song, D., Schilder, F., Hertz, S., Saltini, G., Smiley, C., & et., a. (2019, June). Building and Querying an Enterprise Knowledge Graph. *IEEE Transactions on Services Computing, 12*, 356-369.

Sowa, J. (2011). Cognitive Architectures for Conceptual Structures. *Proceedings of ICCS 2011* (pp. 35-49). Heidelberg, Germany: Springer. Retrieved from jfsowa.com: https://www.jfsowa.com/pubs/ca4cs.pdf

Sowa, J. (2014, September 1). *Controlled Natural Languages For Semantic Systems*. Retrieved April 15, 2024, from jfsowa.com: https://jfsowa.com/talks/cnl4ss.pdf

Sowa, J., & Majumdar, A. (2023, June 1). *Evaluating and Reasoning with and about GPT by John Sowa and Arun Majumdar*. Retrieved April 17, 2024, from youtube.com: https://www.youtube.com/watch?v=6K6F_zsQ264

Vamsi, K., Kommineni, V., & Samuel, S. (2023). *From human experts to machines: An LLM supported approach to ontology and knowledge graph construction*. Retrieved from pure.mpg.de: https://pure.mpg.de/rest/items/item_3575749/component/file_3575750/content

Vanjulavalli, N., & Kovalan, A. (2012, August). Ontology based Semantic Search Engine. *IJCSET, 2*(8), 1349-1353.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 6000–6010). Long Beach, CA, USA: Curran Associates Inc. Retrieved from https://dl.acm.org/doi/10.5555/3295222.3295349

Verdicchio, M., & Perin, A. (2022). When Doctors and AI Interact: on Human Responsibility for Artificial Risks. *Philos. Technol., 35*(11). doi:https://doi.org/10.1007/s13347-022-00506-6

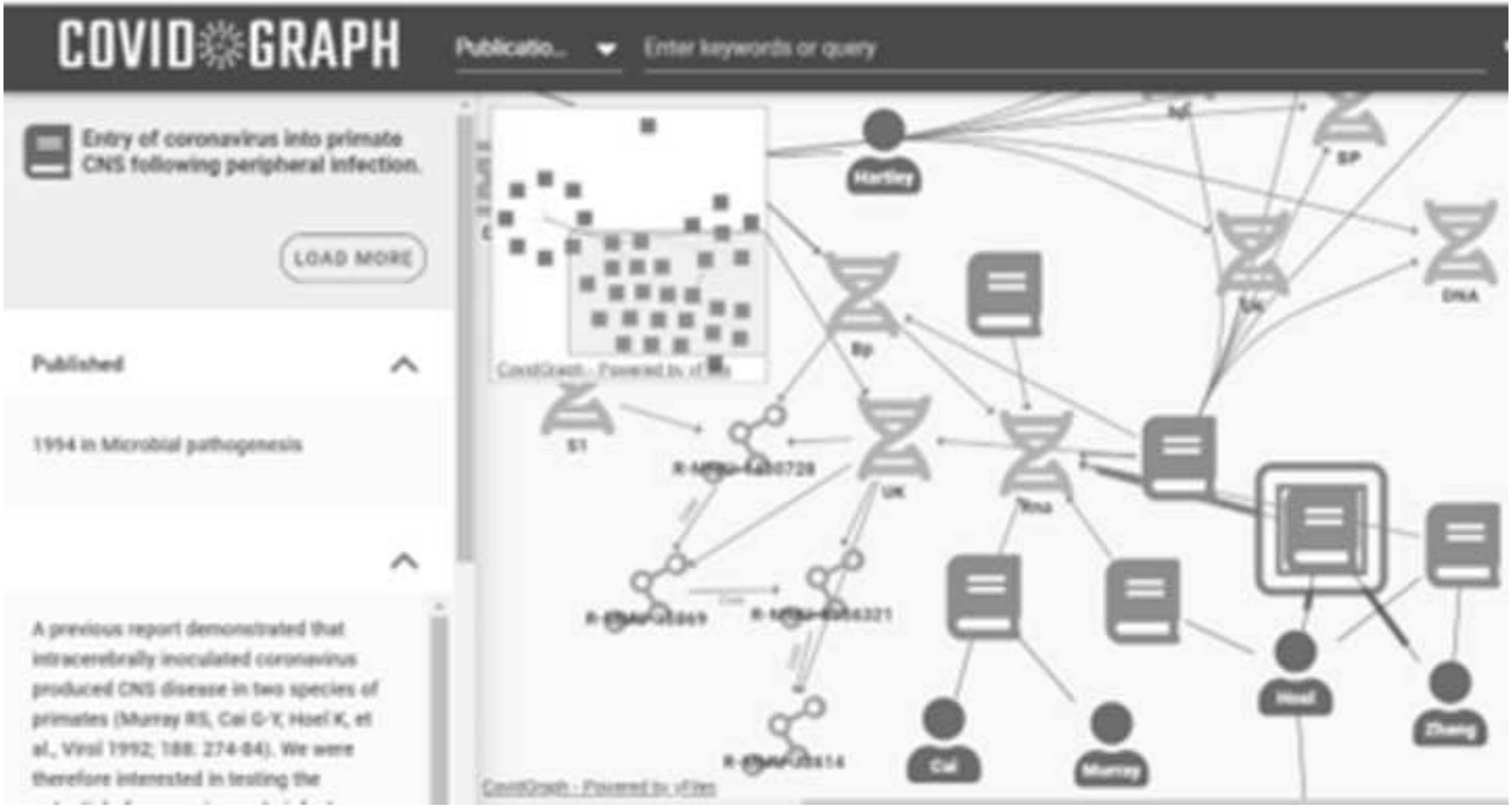
W3C. (2012, March 6). *Comparing SPARQL and SQL by Example*. Retrieved April 11, 2024, from W3C: https://www.w3.org/2012/Talks/0604-SPARQL-SQL/examples

Williams, M., & Hollan, J. (1981). The Process of Retrieval from Very Long Term Memory. *Cognitive Science, 5*, 87-119.

Xu, L., Lu, L., & Liu, M. (2024, April 9). Nanjing Yunjin intelligent question-answering system based on knowledge graphs and retrieval augmented generation technology. *Heritage Science, 12*(118). Retrieved from https://heritagesciencejournal.springeropen.com/articles/10.1186/s40494-024-01231-3

Xu, Z., Jain, S., & Kankanhalli. (2024, January 22). *Hallucination is Inevitable: An Innate Limitation of Large Language Models. ArXiv, abs/2401.11817.* Retrieved from arXiv.org: https://arxiv.org/abs/2401.11817

Yen, J., Neches, R., DeBellis, M., & Szekely, P. (1991). BACKBORD: an implementation of specification by reformulation. In J. W. Sullivan, & S. W. Tyler (Eds.), *Intelligent user interfaces* (pp. 421–444). New York, New York, USA: ACM Press. Retrieved from https://dl.acm.org/doi/10.1145/107215.128723

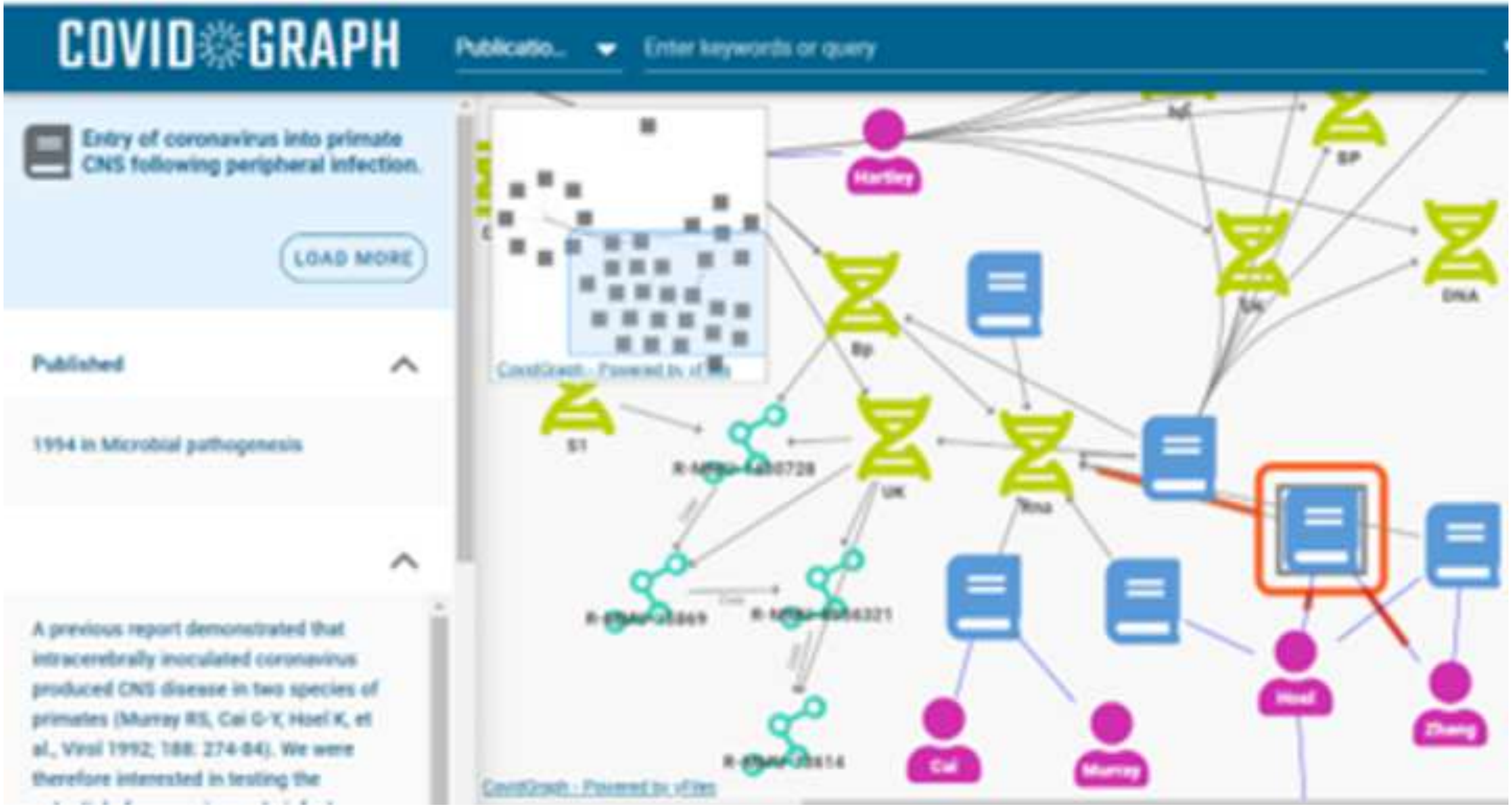


Entry of coronavirus into primate CNS following peripheral infection.

Published

1994 in Microbial pathogenesis

A previous report demonstrated that intracerebrally inoculated coronavirus produced CNS disease in two species of primates (Murray RS, Cai G-Y, Hoel K, et al., Virol 1992; 188: 274-84). We were therefore interested in testing the



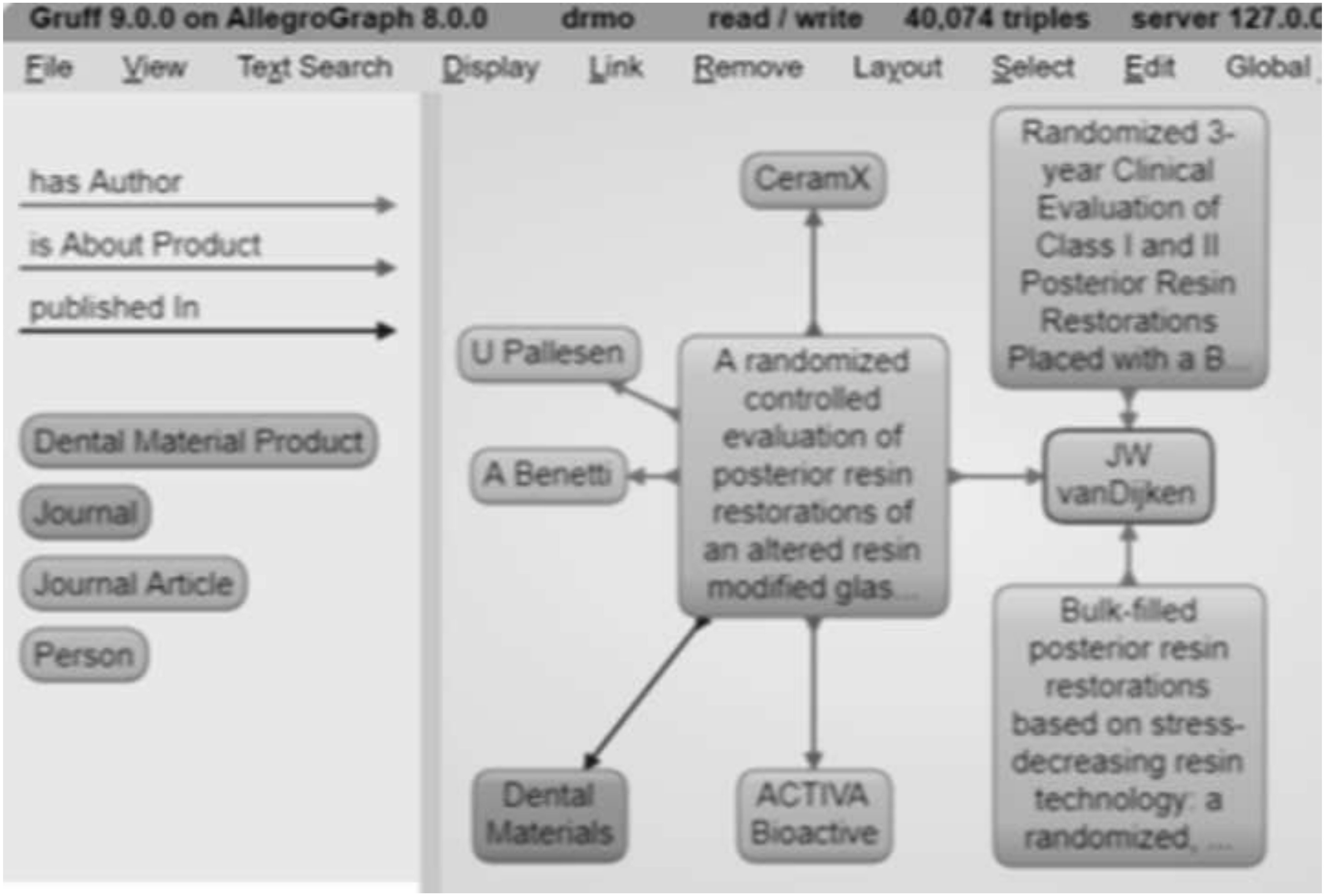
Entry of coronavirus into primate CNS following peripheral infection.

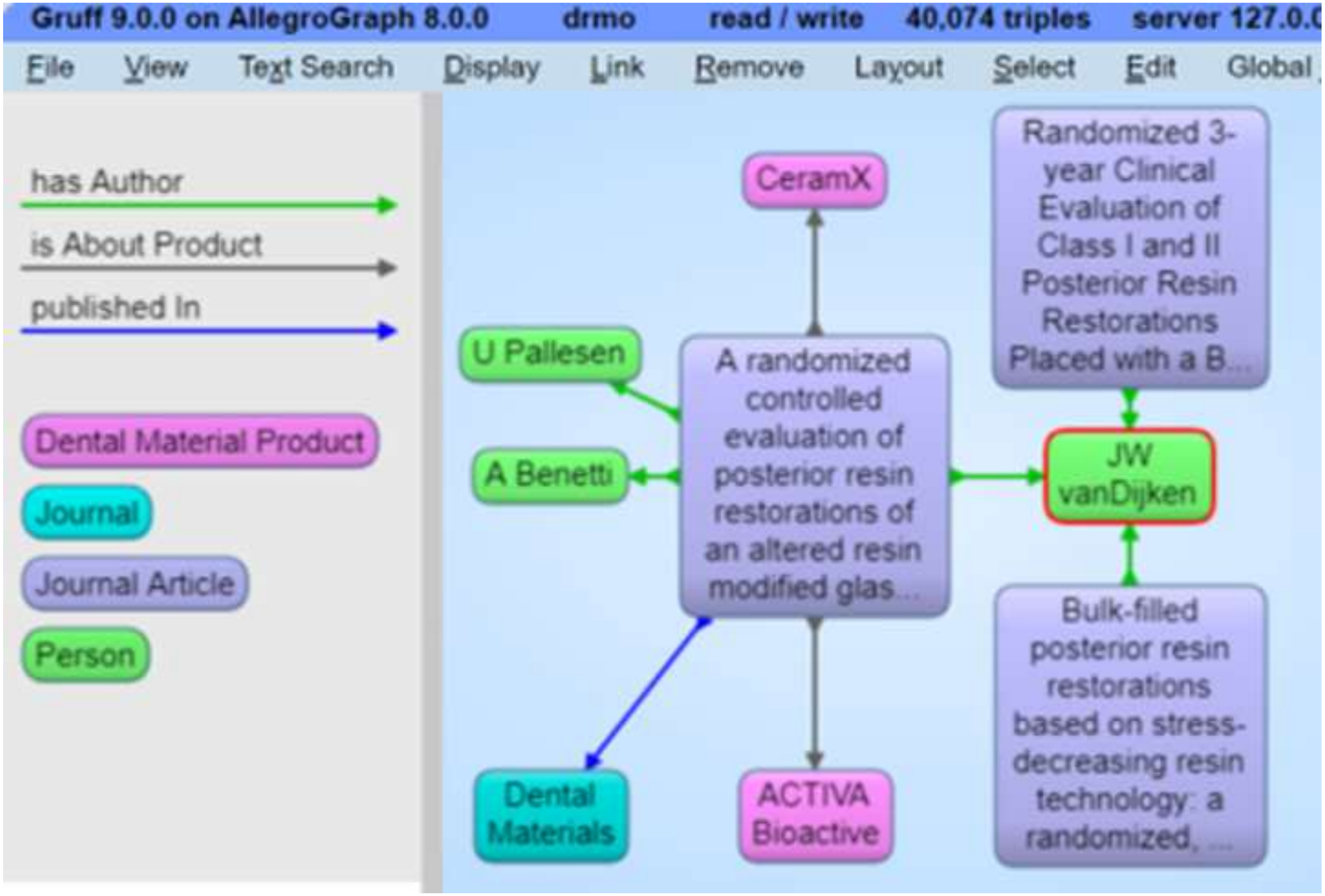
LOAD MORE

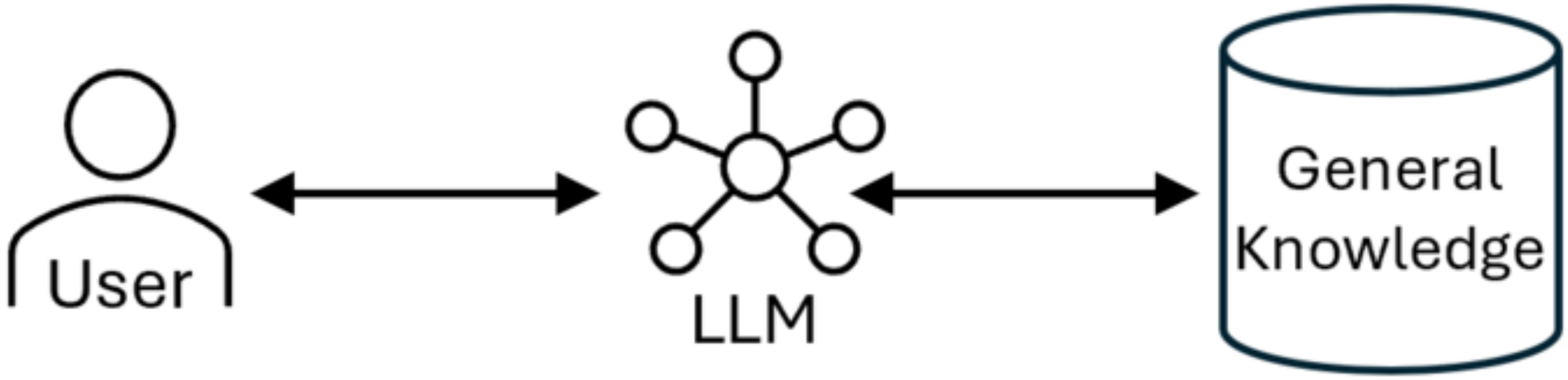
Published

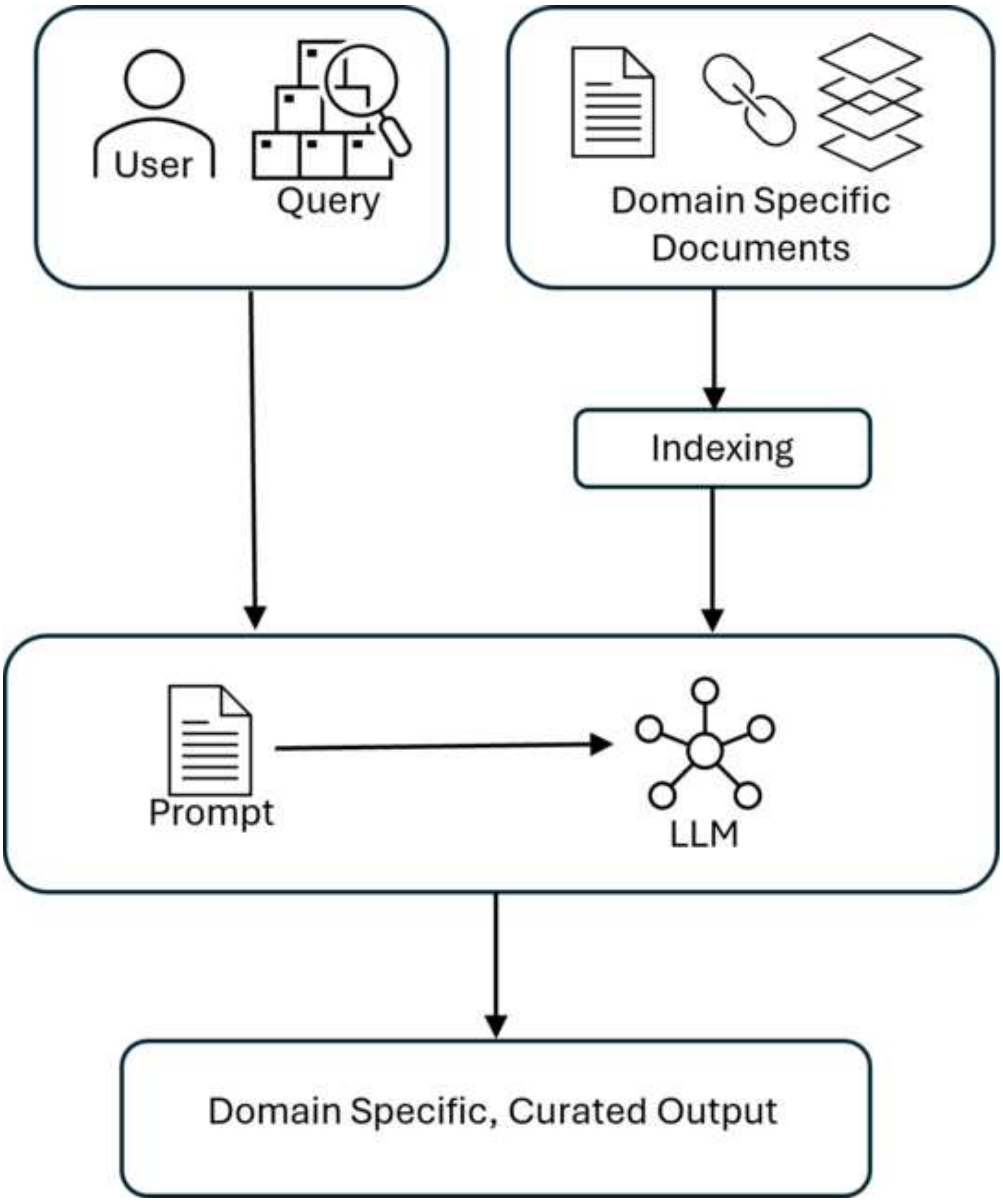
1994 in Microbial pathogenesis

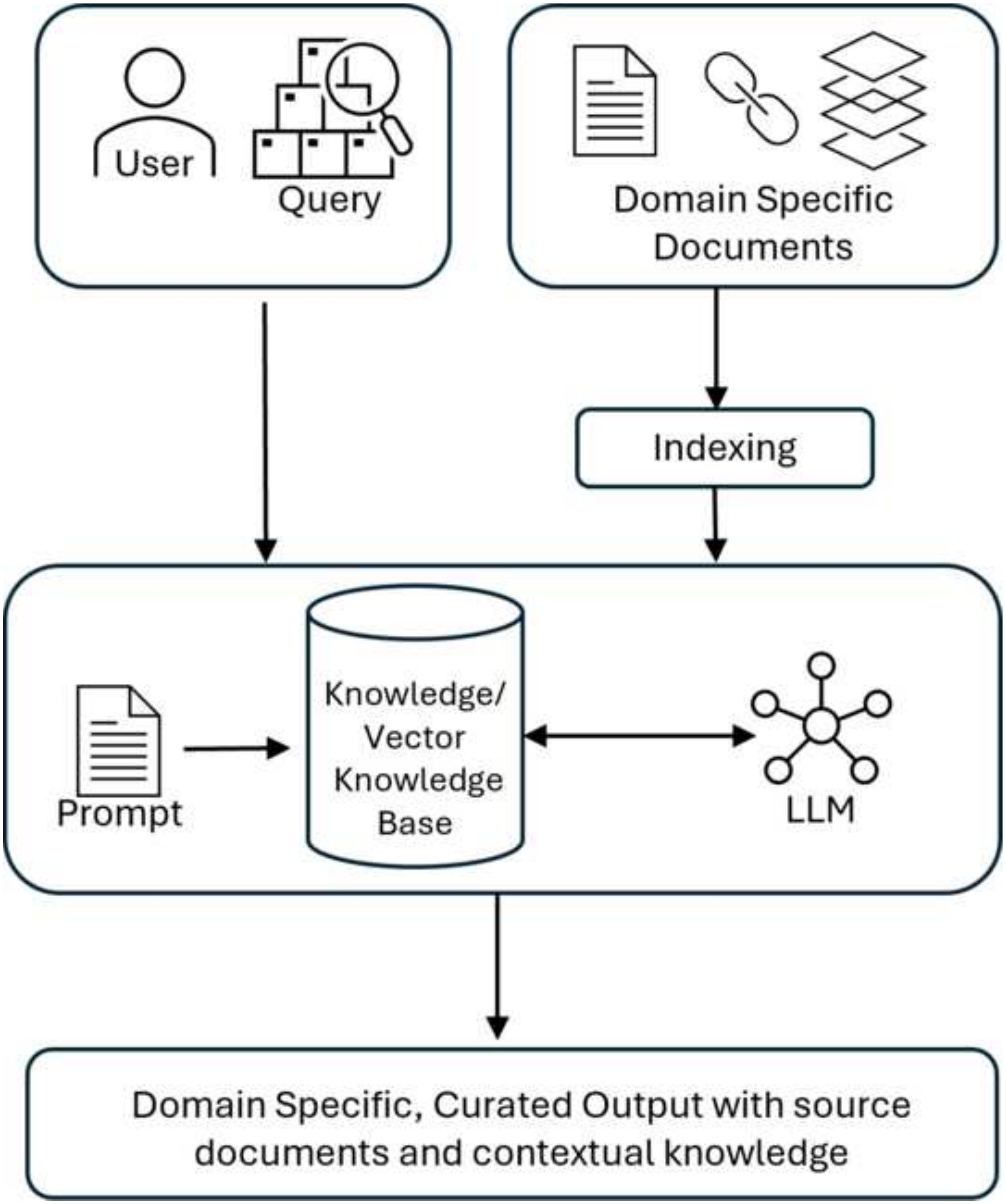
A previous report demonstrated that intracerebrally inoculated coronavirus produced CNS disease in two species of primates (Murray RS, Cai G-Y, Hoel K, et al., Virol 1992; 188: 274-84). We were therefore interested in testing the

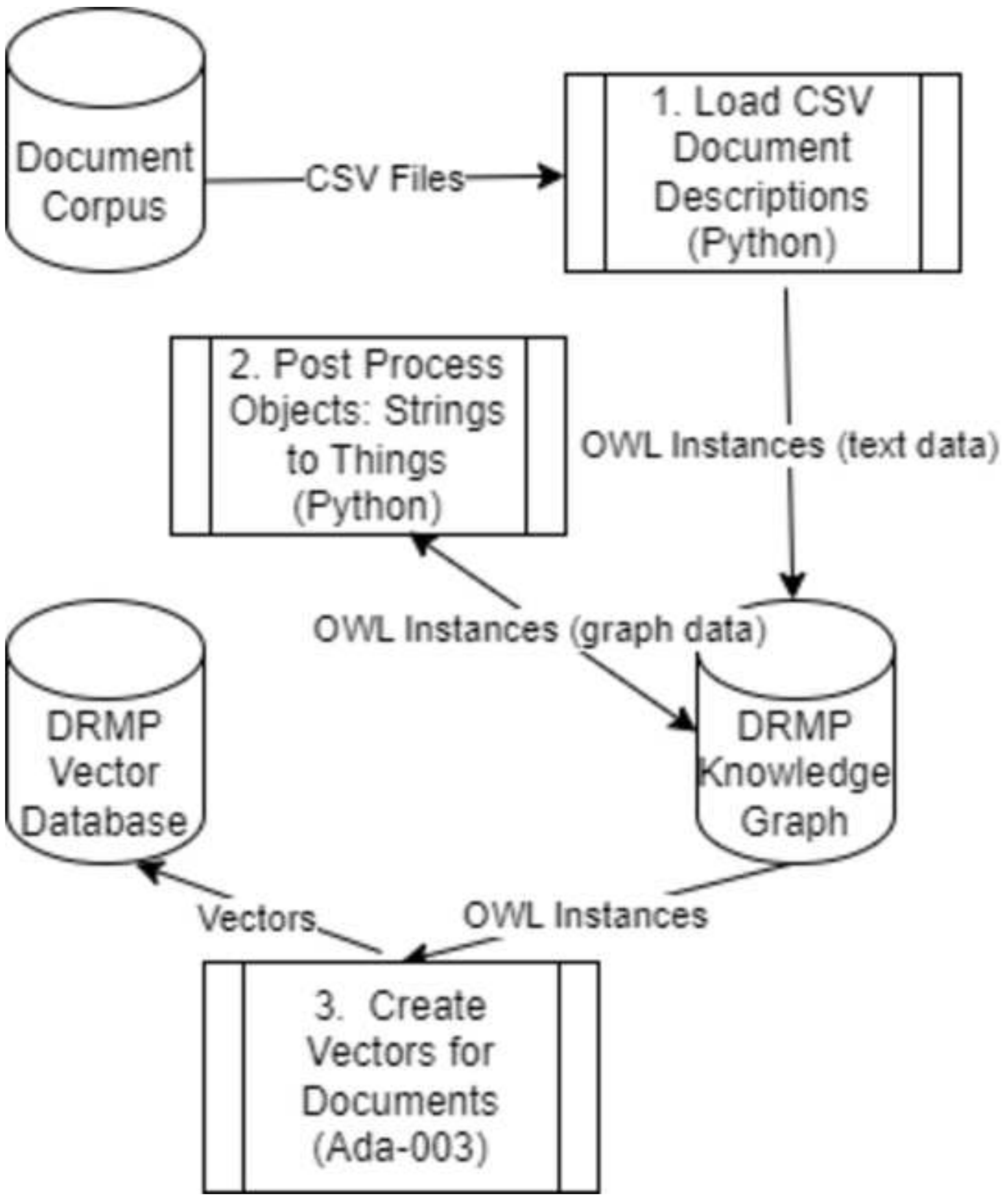

Gruff 9.0.0 on AllegroGraph 8.0.0 drmo read / write 40,074 triples server 127.0.0
File
View
Text Search
Display
Link
Remove
Layout
Select
Edit
Global
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
Randomized 3-year Clinical Evaluation of Class I and II Posterior Resin Restorations Placed with a B...
U Pallesen
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
A Benetti
JW vanDijken
Bulk-filled posterior resin restorations based on stress-decreasing resin technology: a randomized, ...
Dental Materials
ACTIVA Bioactive

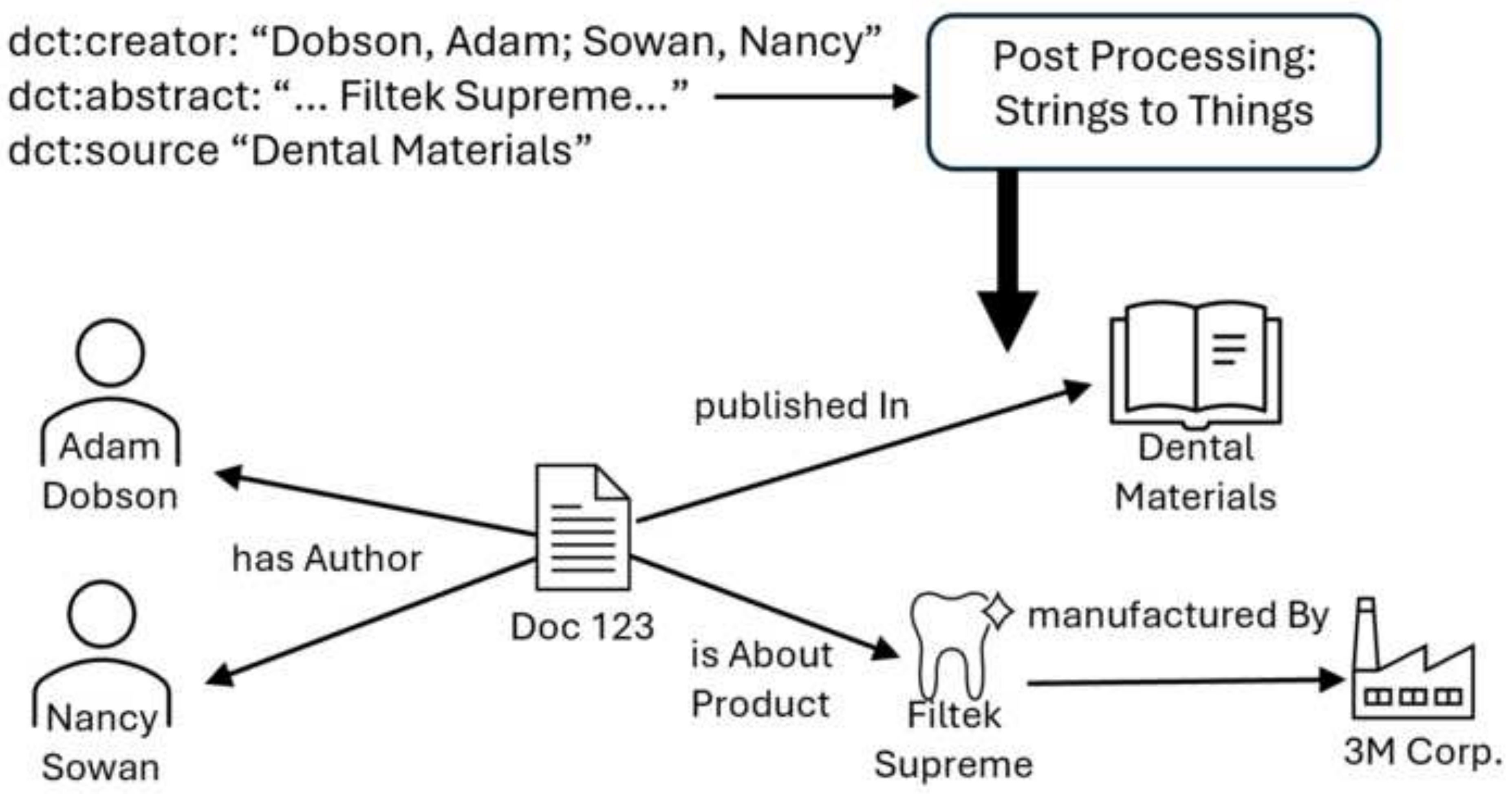

Gruff 9.0.0 on AllegroGraph 8.0.0 drmo read / write 40,074 triples server 127.0.0
File View Text Search Display Link Remove Layout Select Edit Global
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
Randomized 3-year Clinical Evaluation of Class I and II Posterior Resin Restorations Placed with a B...
U Pallesen
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
A Benetti
JW vanDijken
Bulk-filled posterior resin restorations based on stress-decreasing resin technology: a randomized, ...
Dental Materials
ACTIVA Bioactive

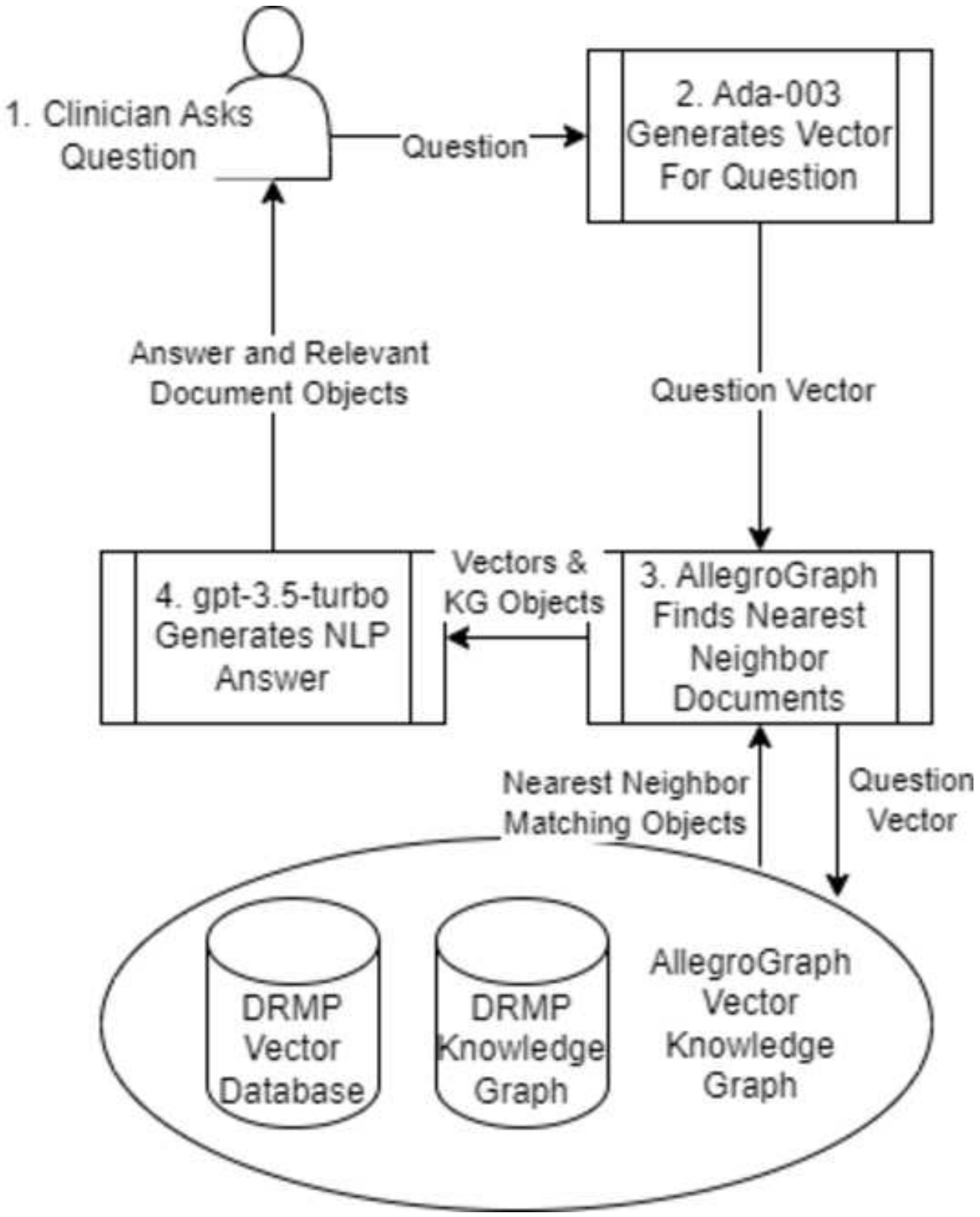

User
LLM
General Knowledge

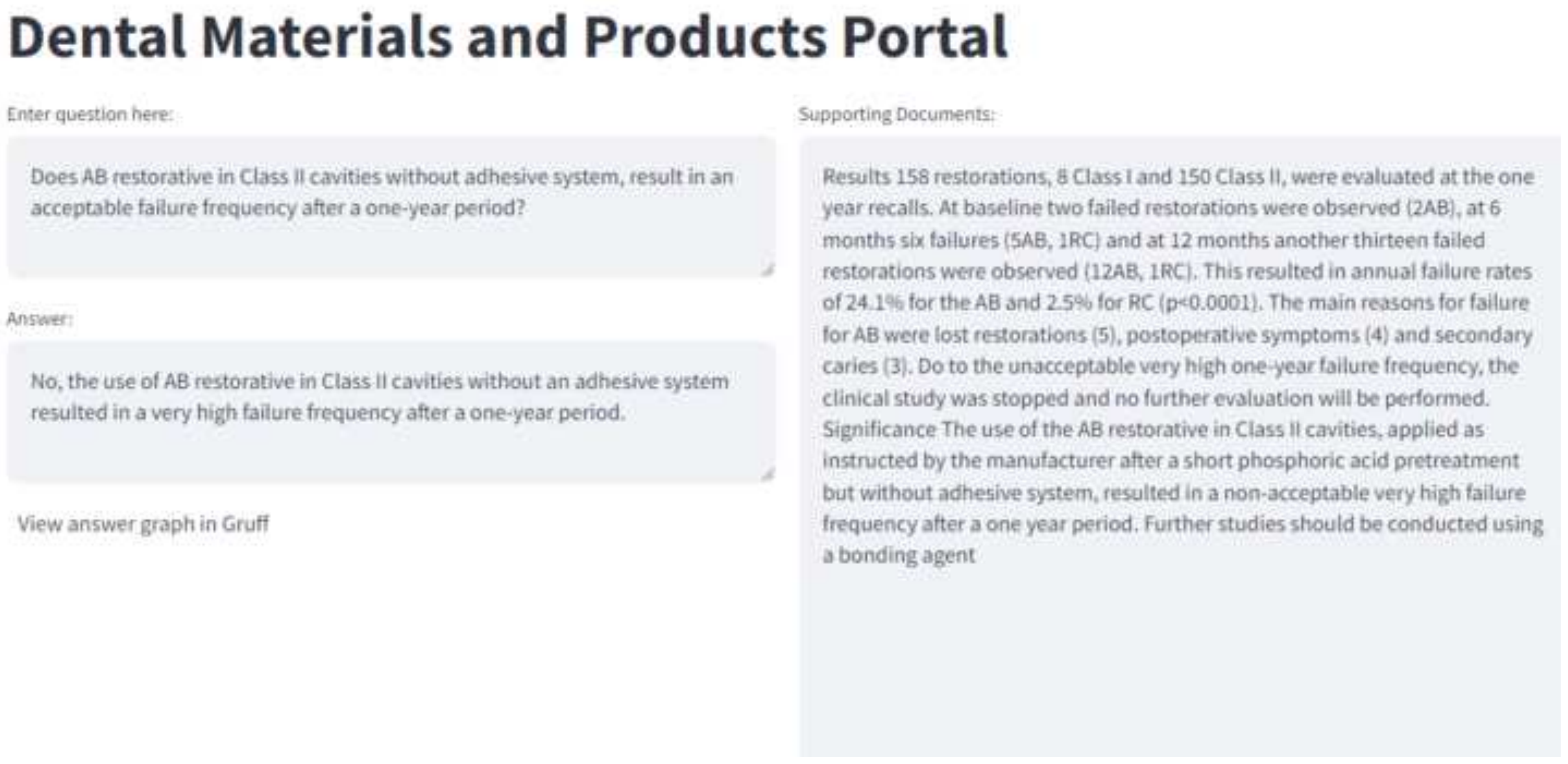

User
Query
Domain Specific Documents
Indexing
Prompt
LLM
Domain Specific, Curated Output


User
Query
Domain Specific Documents
Indexing
Prompt
Knowledge/ Vector Knowledge Base
LLM
Domain Specific, Curated Output with source documents and contextual knowledge


Document Corpus
CSV Files
1. Load CSV Document Descriptions (Python)
2. Post Process Objects: Strings to Things (Python)
OWL Instances (text data)
OWL Instances (graph data)
DRMP Vector Database
DRMP Knowledge Graph
Vectors
OWL Instances
3. Create Vectors for Documents (Ada-003)


dct:creator: “Dobson, Adam; Sowan, Nancy”
dct:abstract: “... Filtek Supreme...”
dct:source “Dental Materials”
Post Processing: Strings to Things
published In
Dental Materials
Adam Dobson
has Author
Doc 123
Nancy Sowan
is About Product
Filtek Supreme
manufactured By
3M Corp.


1. Clinician Asks Question
Question
2. Ada-003 Generates Vector For Question
Question Vector
Answer and Relevant Document Objects
4. gpt-3.5-turbo Generates NLP Answer
Vectors & KG Objects
3. AllegroGraph Finds Nearest Neighbor Documents
Nearest Neighbor Matching Objects
Question Vector
DRMP Vector Database
DRMP Knowledge Graph
AllegroGraph Vector Knowledge Graph



Dental Materials and Products Portal

Enter question here:

Does AB restorative in Class II cavities without adhesive system, result in an acceptable failure frequency after a one-year period?

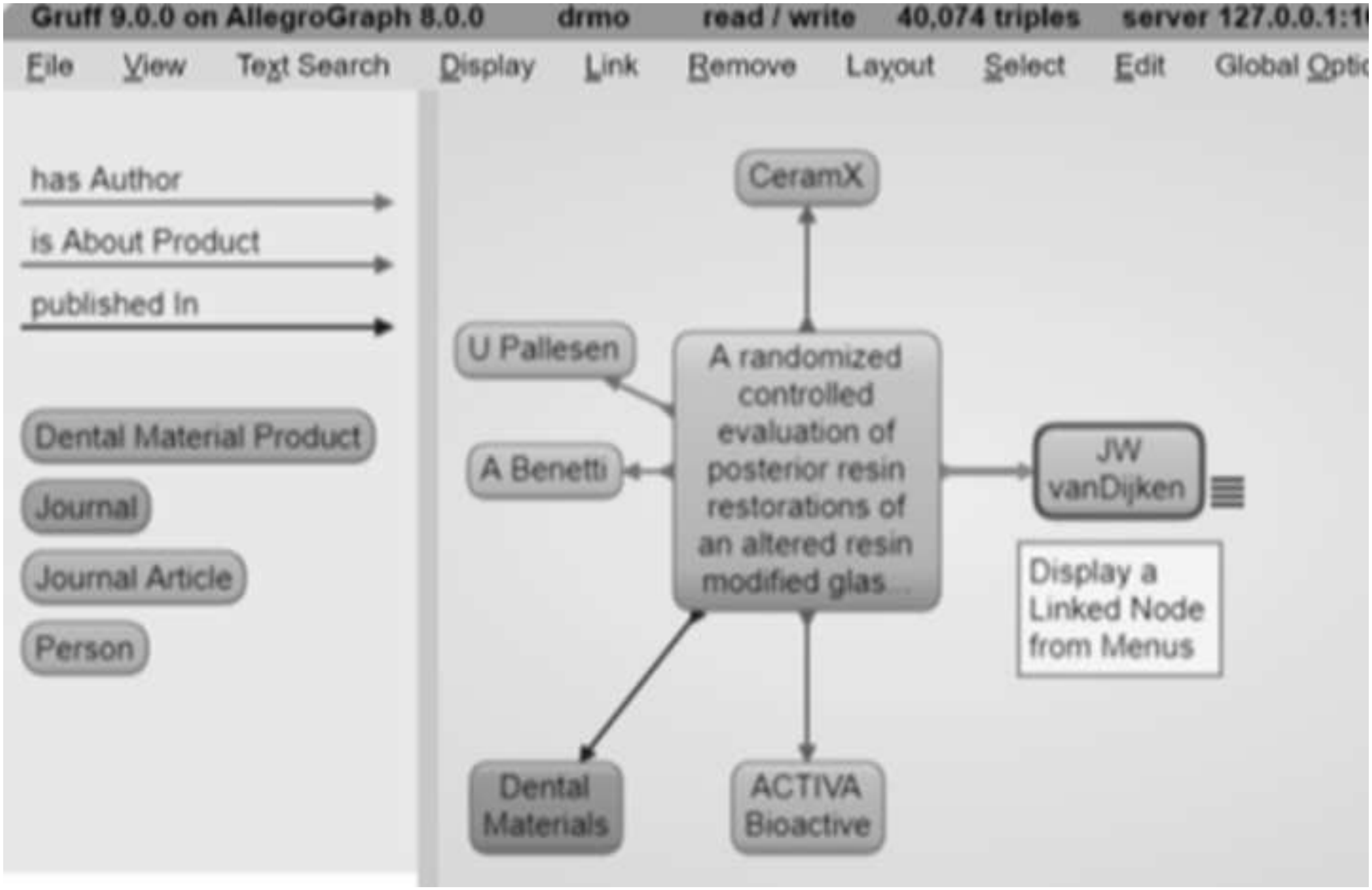
Answer:

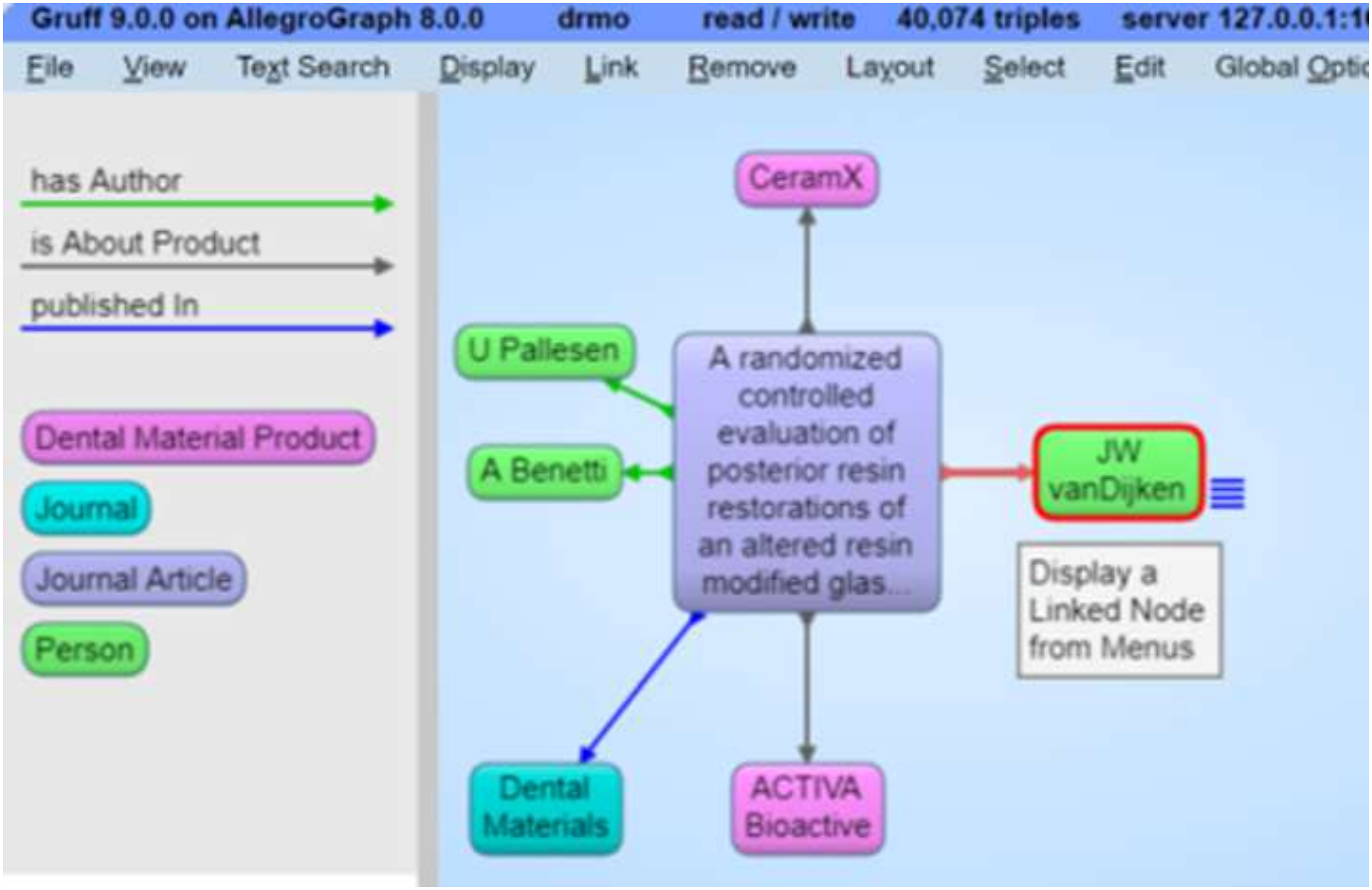
No, the use of AB restorative in Class II cavities without an adhesive system resulted in a very high failure frequency after a one-year period.

View answer graph in Gruff

Supporting Documents:

Results 158 restorations, 8 Class I and 150 Class II, were evaluated at the one year recalls. At baseline two failed restorations were observed (2AB), at 6 months six failures (5AB, 1RC) and at 12 months another thirteen failed restorations were observed (12AB, 1RC). This resulted in annual failure rates of 24.1% for the AB and 2.5% for RC (p<0.0001). The main reasons for failure for AB were lost restorations (5), postoperative symptoms (4) and secondary caries (3). Do to the unacceptable very high one-year failure frequency, the clinical study was stopped and no further evaluation will be performed. Significance The use of the AB restorative in Class II cavities, applied as instructed by the manufacturer after a short phosphoric acid pretreatment but without adhesive system, resulted in a non-acceptable very high failure frequency after a one year period. Further studies should be conducted using a bonding agent


Gruff 9.0.0 on AllegroGraph 8.0.0 drmo read / write 40,074 triples server 127.0.0.1:1
File View Text Search Display Link Remove Layout Select Edit Global Optio
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
U Pallesen
A Benetti
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
JW vanDijken
Display a Linked Node from Menus
Dental Materials
ACTIVA Bioactive


Gruff 9.0.0 on AllegroGraph 8.0.0
drmo
read / write
40,074 triples
server 127.0.0.1:1
File
View
Text Search
Display
Link
Remove
Layout
Select
Edit
Global Optio
has Author
is About Product
published In
Dental Material Product
Journal
Journal Article
Person
CeramX
U Pallesen
A Benetti
A randomized controlled evaluation of posterior resin restorations of an altered resin modified glas...
JW vanDijken
Display a Linked Node from Menus
Dental Materials
ACTIVA Bioactive