Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«Використання LLM для побудови графу знань нейронаукових статей»**

Виконав: студент 4-го року навчання
освітньої програми «Комп'ютерні науки»,
спеціальності 122 Комп'ютерні науки

Єремеєва Софія Сергіївна

Керівник: Швай Н.О.,
кандидат фіз.-мат. наук, доцент

Рецензент: Шаповал Н. В.,
к.т.н., доцент каф. штучного інтелекту, ІПСА НТУУ
«КПІ ім. Ігоря Сікорського»

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК_____
(підпис)

«_____» _____ 20___р.

Київ - 2024

Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
проф., доктор фіз.-мат. наук
_____*Гороховський С.С.*
(підпис)
“____”_____ 2024

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
для кваліфікаційної роботи
студентці 4-го курсу, факультету інформатики
Єремеєвій Софії Сергіївні

Тема: «Використання LLM для побудови графу знань нейронаукових статей»

Зміст кваліфікаційної роботи:

Анотація

1. Вступ
2. Нейронаукові платформи, що існують зараз
3. Графи знань та великі лінгвістичні моделі
4. Методологія створення графу знань
5. Аналіз

Висновки

Список літератури

Дата видачі “____”_____ 2024 Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Графік підготовки кваліфікаційної роботи до захисту

Графік узгоджено «_____» _____ 2024р.

№ з/п	Перелік робіт	Термін виконання етапу	Підпис наукового керівника	Дата ознайомлення наукового керівника	Примітка
1.	Отримання теми кваліфікаційної роботи.	жовтень			
2.	Ознайомлення з темою кваліфікаційної роботи.	жовтень-листопад			
3.	Розробка плану та структури роботи.	листопад-грудень			
4.	Робота з науковою літературою, опис основних означень.	січень			
5.	Дослідження результатів отриманих в літературі.	лютий			
6.	Робота над текстовим оформленням результатів.	березень-квітень			
7.	Попередній аналіз кваліфікаційної роботи. Виправлення помилок.	травень			
8.	Попередній захист кваліфікаційної роботи.	травень			
9.	Захист кваліфікаційної роботи.	червень			

Науковий керівник _____
(ПІБ)

Виконавець кваліфікаційної роботи _____
(ПІБ)

Contents

Annotation

In recent decades, the field of neuroscience has significantly advanced. The number of neuroscientific fields and research projects has increased, and so has the amount of knowledge gained. However, a new challenge has emerged: the overwhelming amount of information and publications. New articles are published every few minutes, and researchers no longer have the capacity to keep up with all the new discoveries. On the other hand, disciplines and specializations are becoming increasingly narrow, and scientists rarely venture into other fields, focusing all their efforts on their own research. Therefore, researchers require efficient tools for reading and summarizing academic papers and uncovering significant scientific knowledge.

In our approach, we suggest leveraging Large-scale Language Models (LLMs) to construct a knowledge base that would extract critical theses from articles and integrate them into a unified space. We propose the Neuro Knowledge Graph (NeuroKG), which can uncover hidden connections between keywords in articles through queries. This will streamline the analysis of many articles and promote broader connections across narrow scientific areas.

1 Introduction

Over the past two decades, the field of neuroscience has advanced significantly. This progress is evident in increasing research projects such as the Human Connectome Project, the Human Brain Project, etc (Yeung et al., 2017). There has also been a noticeable rise in the volume of publications in recent years. As of 2016, in the biomedical field alone, more than 1 million papers are poured into the PubMed database annually — about two papers per minute (Landhuis, 2016). In contrast, scientists read an average of only 22 scholarly articles per month (or 264 per year) (Van Noorden, R., 2014). Such a disparity between existing information and what researchers can absorb is enormous. Unperceived knowledge becomes equivalent to the absence of knowledge.

On the contrary, we face the issue of specialized science, which poses risks of isolation (Arturo Casadevall, Ferric C. Fang, 2014). This is mainly observed in neuroscience, where countless specializations exist. The lack of cross-disciplinary interaction does not allow us to understand the processes of the neural system at all levels, from genetics to behaviour.

The research purpose is to develop and validate an approach to integrating neuroscientific knowledge from academic papers using a Large-scale Language Model (LLM) based on a Knowledge Graph (KG) to overcome the problems of information overload and specialization isolation. The sample can be seen in Fig. 1. In light of these results, the NeuroKG has the potential to help neuroscientists gain a deeper understanding of their field through continuous and faster processing of new literature and expanding into other areas of neuroscience.

The object of this research is the NeuroKG, a LLM-powered neuroscientific Knowledge Graph.

To achieve the research objectives, the following methods were used:

- *Literature review on KGs and LLMs*: Reviewing existing KGs and LLMs combinations, focusing on their applications in scientific fields.
- *Collection of a scientific articles dataset*: Gathering a collection of neuroscientific papers from Scopus for the KG construction.
- *Building the KG using LLMs and Neo4j*: Developing a pipeline to convert

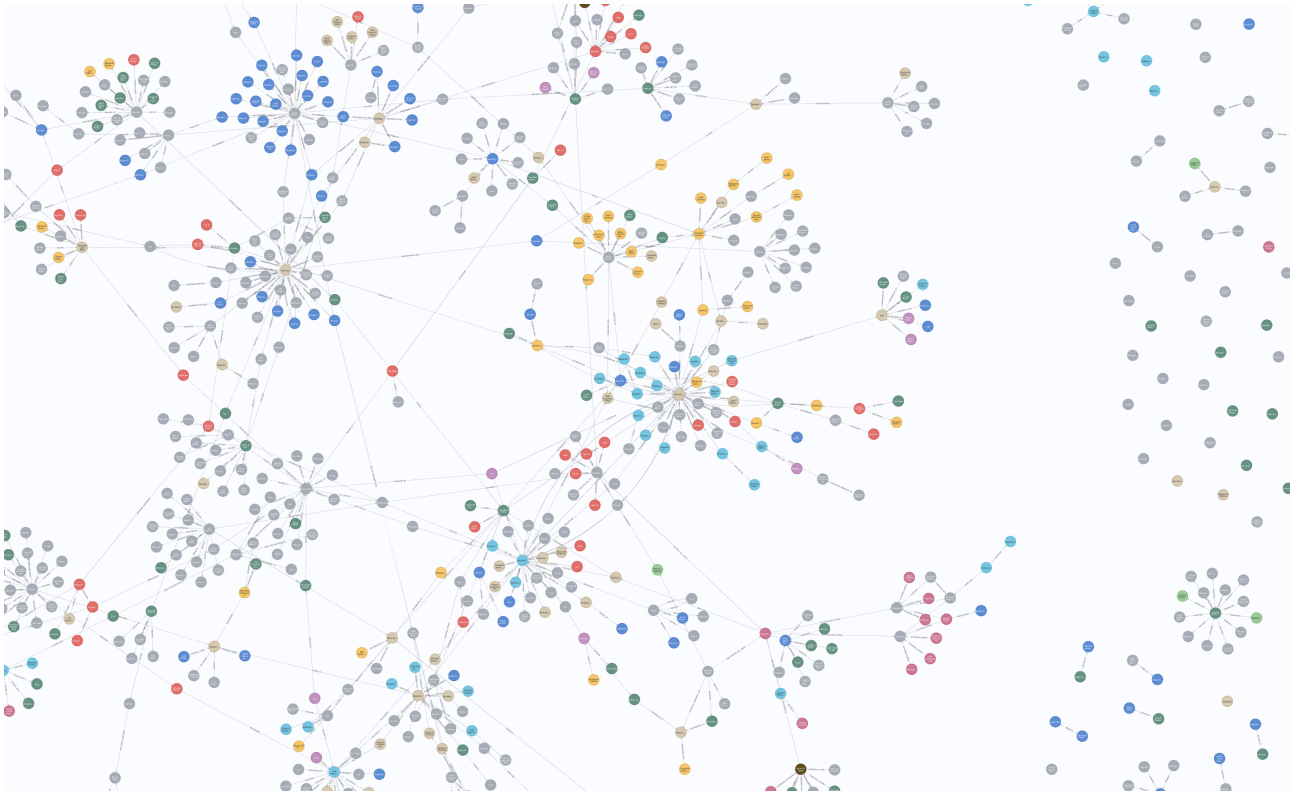


Figure 1: An example of the created NeuroKG.

PDF papers into a structured KG. This involves:

  - Splitting text into chunks of 2000 characters.
  - Using conversational memory for better sequential text processing.
  - Developing prompt templates for different LLM applications.
  - Generating article summaries with LLM.
  - Converting text to structured output and local graphs with LLM.
  - Generating multiple local graphs and selecting the best ones.
  - Integrating new local graphs into the existing KG.

- *Analysis of the created KG*: Applying metric graph theory and performing complex queries to analyze the constructed KG.

- *Validation of the proposed approach*: Evaluation of the effectiveness of NeuroKG in promoting interdisciplinary understanding and identifying hidden relationships in the neuroscience literature.

NeuroKG, based on 100 neuroscience articles, demonstrates the potential to help neuroscientists process new literature more efficiently and explore other areas

of neuroscience through the continuous and rapid integration of new information. The main tools used to develop the pipeline were the OpenAI LLM model 'gpt-4-turbo-preview', the Langchain library to create better interaction with the model, and the Neo4j graph database. Less essential libraries and the repository can be found at: `https://github.com/yeremeieva/NeuroKG`.

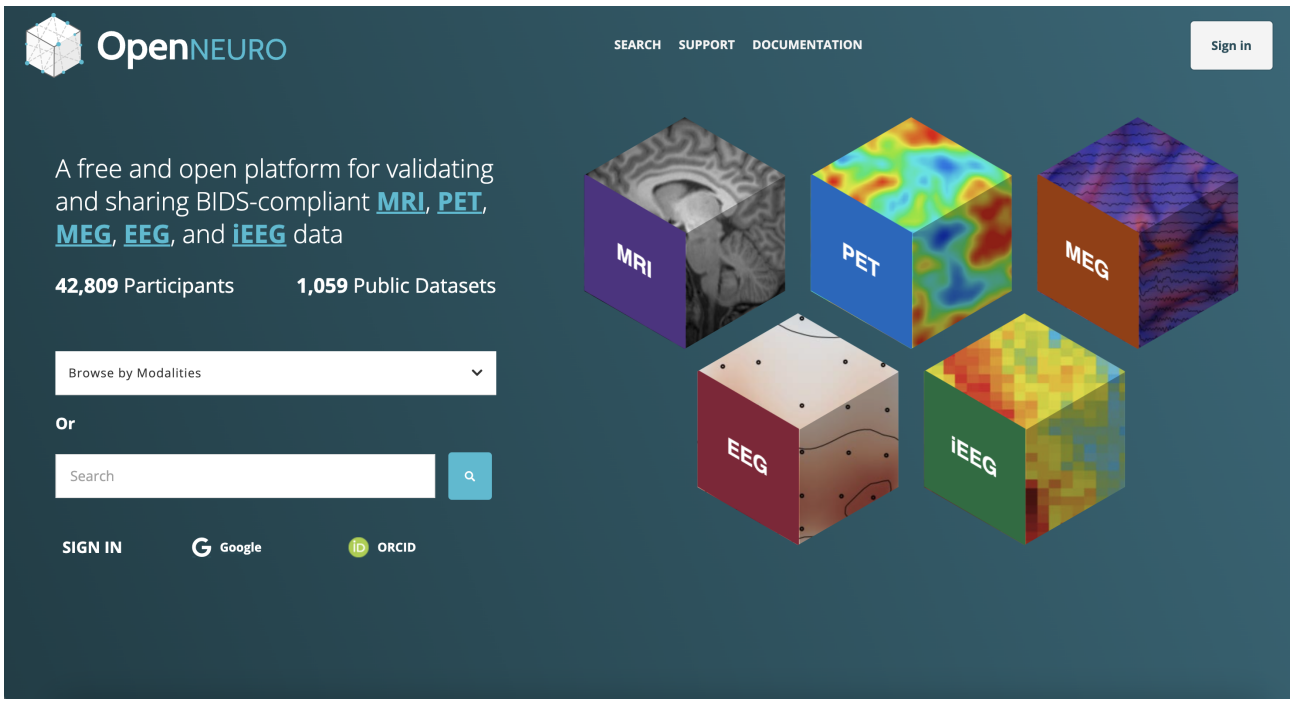


Figure 2: The OpenNeuro data search interface.

2 Existing Neuroscientific Platforms

There are already a significant number of diverse neuroscientific platforms that store and exchange information. We will review some of the largest ones to see if they address the issue of vanishing knowledge from the vast number of publications and specialized science and learn how they look and function overall.

2.1 OpenNeuro data archive

This resource[1] stores lots of various datasets using The FAIR principles (Wilkinson et al., 2016). These principles have formalized the notion that shared data needs to be findable, accessible, interoperable, and reusable to be maximally useful. To achieve these goals, the Brain Imaging Data Structure (BIDS) common standard for data and associated metadata was developed, following a common standard for organizations so that data users can easily understand and reuse the shared data (Markiewicz et al., 2021).

This platform provides a large amount of medical data that has been used in scientific articles or simply collected, which is very useful. User interfaces with

[1] `https://openneuro.org`

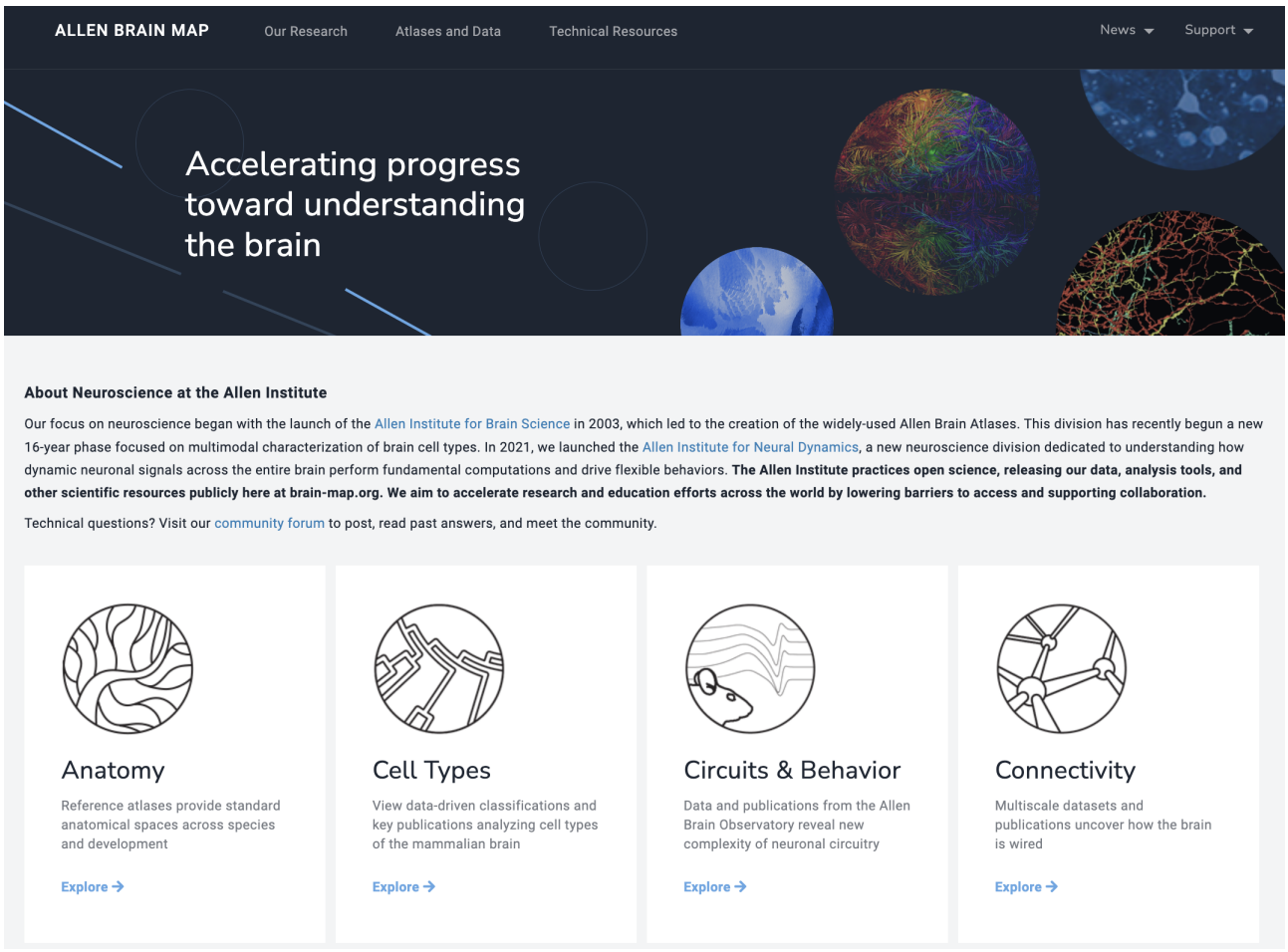


Figure 3: The Allen Brain Atlas user interface.

the number of datasets can be seen in Fig. 2. The dataset example can be found in Appendix A in Fig. 20. Unfortunately, it is not linked to key knowledge from articles and does not solve the problem that is relevant to us.

2.2 The Allen Brain Atlas

At the intersection of modern neuroscience and genetics, the underlying molecular architecture of cell type or disease-associated changes in gene regulation is key to developing and testing new hypotheses for understanding normal brain function and ultimately developing therapies for neurological disorders and disease (Sunkin et al., 2012). Therefore, The Allen Brain Atlas[1] aims to provide interactive datasets of mouse and other animal brains to visualize the architecture of cells and their relationship with genes. The interface of their platform can be seen in Fig. 3 and in Appendix A provided interfaces with interactive visual-

[1] `http://www.brain-map.org`

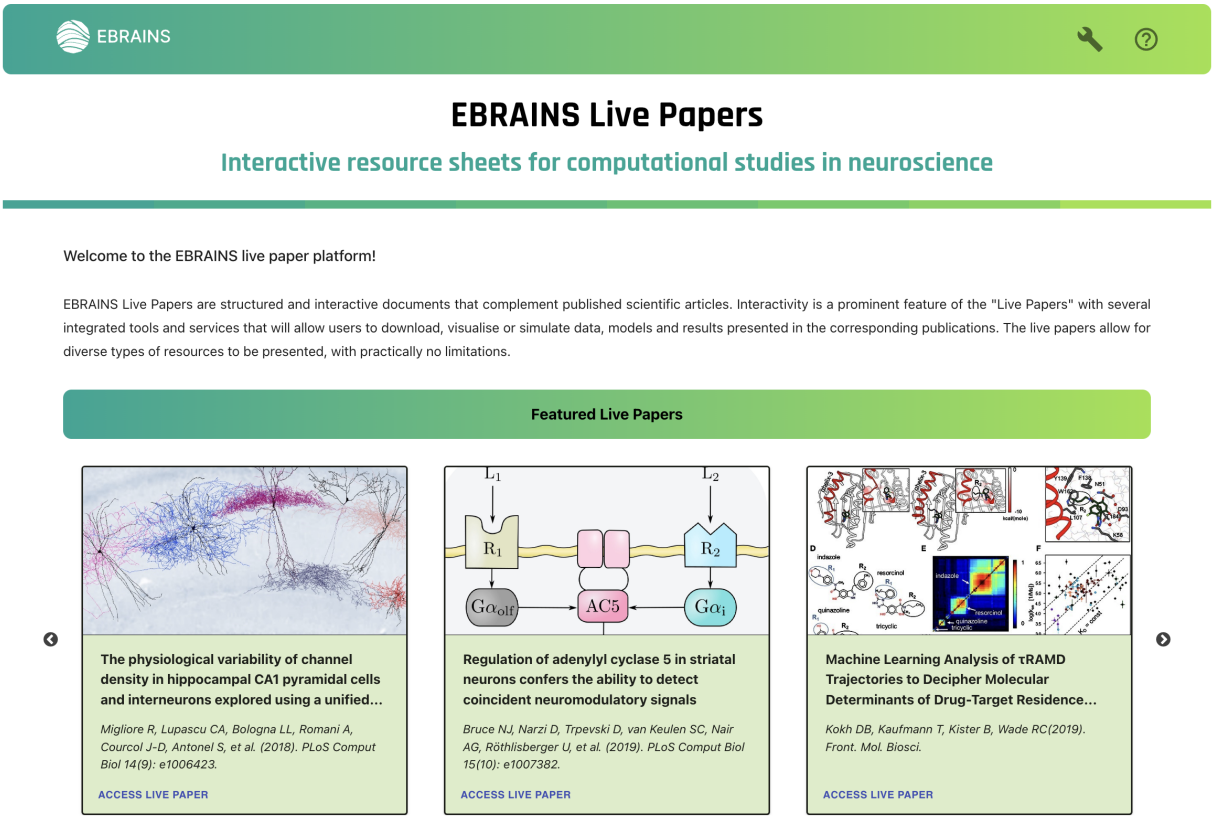


Figure 4: The EBRAINS Live Papers user interface.

izations of real brain sections in Fig. 21, individual 3D models of these sections in Fig. 22, brain slices along with 3D cell models in Fig. 23.

Overall, this platform is useful for understanding the real architecture of cells and their interactions with genes, but it does not store information from articles.

2.3 EBRAINS Live Papers

EBRAINS Live Papers[1] are interactive documents combining code, models, and data. Interactivity is a prominent feature of the "Live Papers" with several integrated tools and services that allow users to download, visualize or simulate data, models and results presented in the corresponding publications (Appukuttan et al., 2023).

In Fig. 4, you can see the user interface and recommended articles containing interactive components. Fig. 24 provides an example of an interactive neuron from a publication. Overall, this resource is useful, but like the previous ones, it

[1] `https://live-papers.brainsimulation.eu`

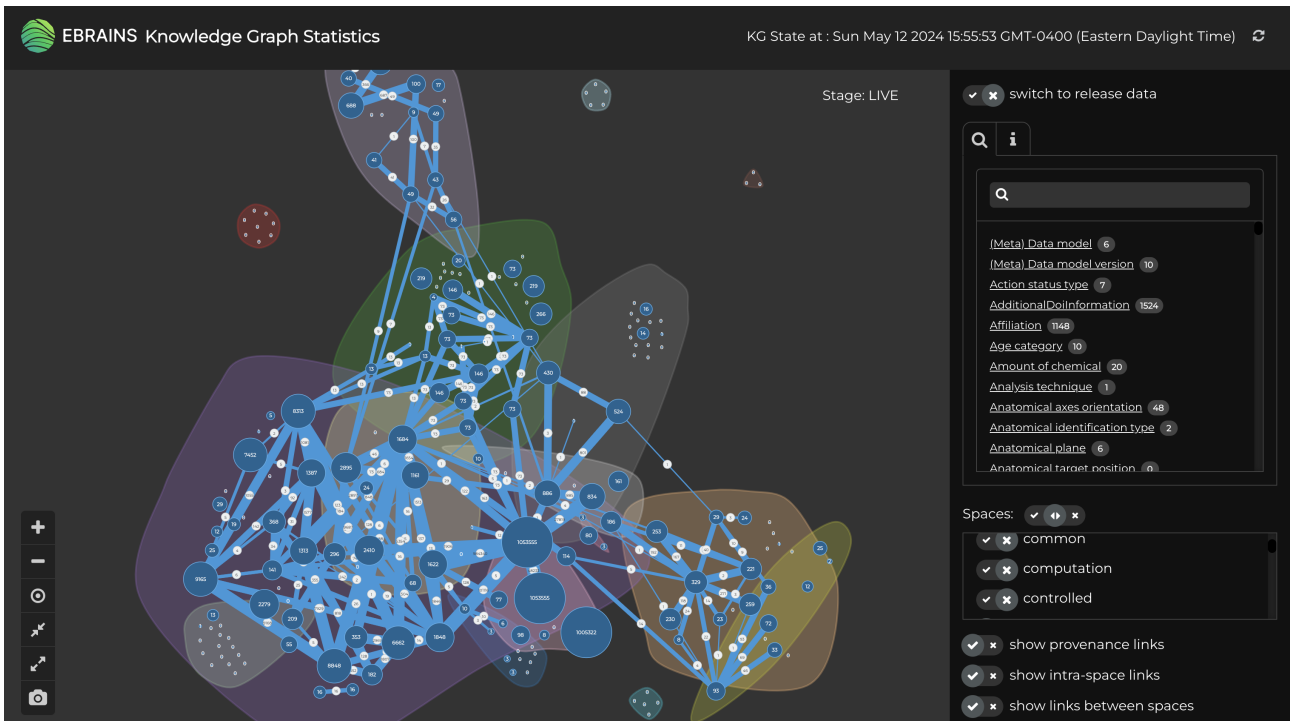


Figure 5: The EBRAINS Knowledge Graph in the graph representation.

does not address the issue of knowledge accumulation.

2.4 EBRAINS KG

The EBRAINS Knowledge Graph[1] provides access to and storage of high-quality neuroscientific data, facilitating data discovery and re-use in the scientific community by providing an online conduit for both sharing and easy access to research data, computational models, and software[2]. EBRAINS has a wide range of functionalities, but this search engine offers broader capabilities compared to the OpenNeuro platform. Additionally, the platform provides its representation in the form of a graph[3], as depicted in Fig. 5. However, the nodes and edges in this graph represent only metadata describing datasets to facilitate their search.

All of the described neuroscientific platforms are highly useful but, unfortunately, do not consider knowledge from scientific papers. Scientists' conclusions and discoveries remain on paper (even electronically).

[1] https://search.kg.ebrains.eu/?category=Dataset

[2] https://www.humanbrainproject.eu/en/science-development/focus-areas/data-and-knowledge/

[3] https://stats.kg.ebrains.eu/

3 Combining Knowledge Graphs and Large Language Models

3.1 Pros and Cons of the KGs and LLMs

KG is a data graph intended to accumulate and convey knowledge of the real world. Its nodes represent entities of interest, and its edges represent relations between these entities (Hogan et al., 2021). The advantages of the KG are structural knowledge, accuracy, decisiveness, interpretability, domain-specific knowledge, and evolving knowledge. The disadvantages are incompleteness, lack of language understanding, and many unseen facts (Pan et al., 2023). Additionally, KG is difficult to construct (Zhong et al., 2023), and current approaches in KGs are inadequate in handling the incomplete and dynamically changing nature of real-world KGs (Pan et al., 2023).

At the same time, LLMs are pre-trained models on a large-scale corpus demonstrating outstanding potential in various NLP tasks (Yang et al., 2023). Some benefits of the LLMs are general knowledge, language processing and generalizability. However, LLM brings many more drawbacks. The following can be highlighted: implicit knowledge, hallucination, indecisiveness, and lack of domain-specific/new knowledge (Pan et al., 2023).

As can be observed, each of these architectures can enhance one another through different modes of interaction.

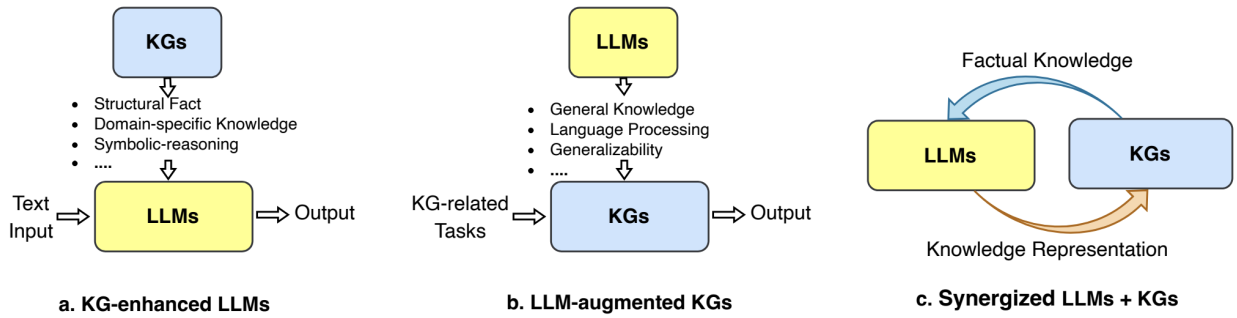


Figure 6: Various combinations of the KGs and LLMs. (a.) KG-enhanced LLMs. (b.) LLM-augmented KGs. (c.) Synergized LLMs + KGs (Pan et al., 2023).

3.2 Combination frameworks

There are several possible options for combining KG and LLM because the advantages of these approaches complement each other's shortcomings described above. We can identify the following three model hybrids that are represented in Fig. 6:

- *KG-enhanced LLMs*: A structured graph can improve a model at different stages of its creation. For example, during the pre-training stage, KGs can help LLMs learn knowledge from it. Also, during the inference stage, retrieving knowledge from KGs can significantly improve the performance of LLMs in accessing domain-specific knowledge (Lewis et al., 2020).

- *LLM-augmented KGs*: LLM brings the advantage of processing the textual corpus in the KGs and then using the representations of the text to enrich KGs representation (Pan et al., 2023). Moreover, LLMs can process the original corpus and extract relations and entities for KG construction (Kumar et al., 2020).

- *Synergized LLMs + KGs*: aims to integrate LLMs and KGs into a unified framework to enhance each other mutually (Pan et al., 2023).

For our work, we chose LLM-augmented KG because the graph is the central object of creation, and the model is merely a means of creating it for processing large volumes of text. This solution is to leverage LLM for reading articles and leave humans with the KG for their analysis.

In the article (Buehler, 2024), LLM-augmented KG is considered for analyzing scientific literature. As a result of combining a large number of publications and using LLMs in scientific analysis, the development of new ideas and hypotheses has emerged as a possible approach. Overall, generative AI can connect different areas of knowledge by focusing the generative task on finding analogies or by tasking the AI model to identify, propose, or explain relationships between disparate concepts or knowledge. And, since this has never been done in neuroscience despite the great need, we have attempted to develop a pipeline specifically for these tasks.

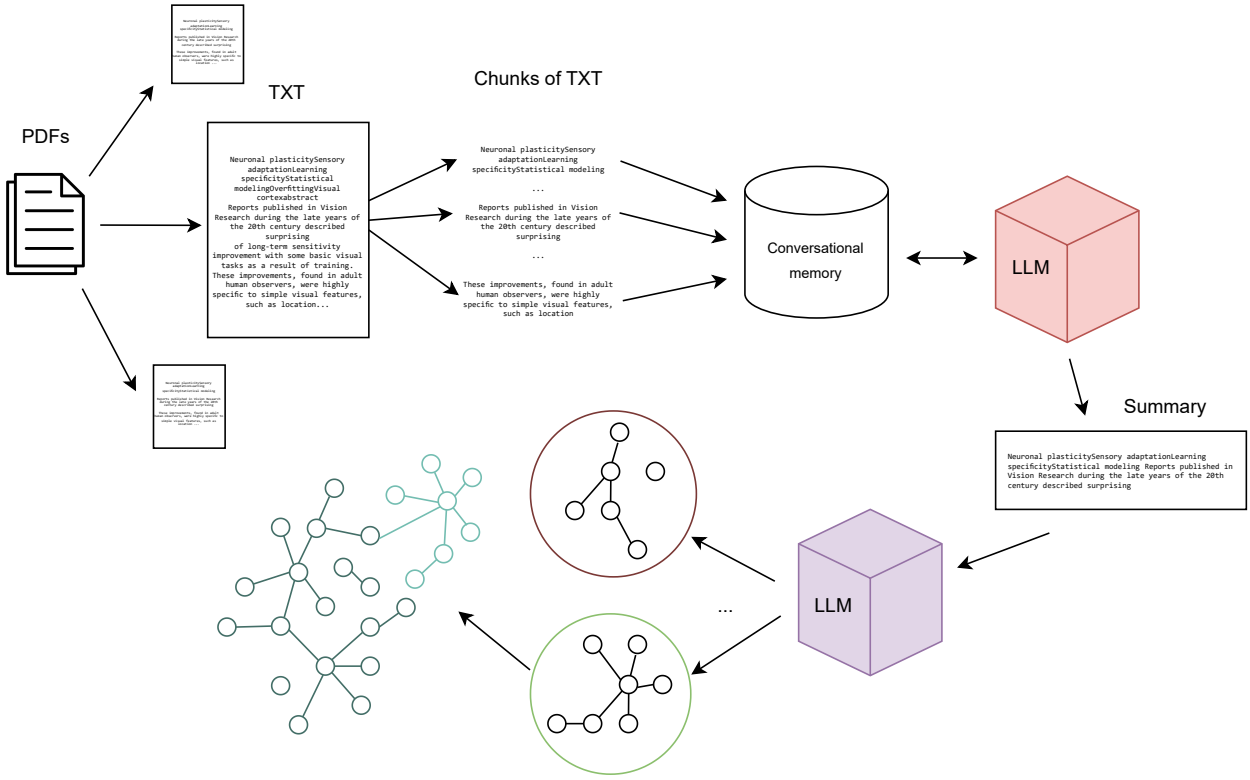


Figure 7: The pipeline of converting PDF papers into the KG.

4 Pipeline of constructing the Knowledge Graph

4.1 Summarizing the PDF articles

Preliminary Dataset

To avoid being tied to a specific journal or field, we utilized Scopus search to collect necessary literature. Additionally, to avoid being limited by year of publication or document type, such as paper or review, we used the search query 'Brain AND Neuro' and then sorted by descending number of citations. The dataset includes the first 100 articles available in open access and containing at least three pages of plain text.

As we can see from the pipeline architecture in Fig. 7, the following step after collecting the necessary papers is reading text from the .pdf file and saving it into the .txt file. Various Python libraries can handle this task, and for the NeuroKG, we applied the PyPDF2 library to parse the documents. The last two pages were not considered because most of the paper only contains references.

Most LLMs have limits on the number of input tokens in the query. For

OpenAI models, this value is mostly 4,096 tokens, while the length of the ten-page article is around 16,000 tokens. To overcome this limitation, we first applied chunking. This is a divide-and-conquer approach that solves complex problems by breaking them down into smaller, more manageable subproblems[1]. We split the paper's text into chunks of 2,000 context windows.

Summarizing the papers

Secondly, to summarize the whole article and process all created chunks by the LLM, we have included conversational memory, namely Conversation Buffer Window Memory.

LLMs, by default, have no concept of history or memory. Every query or call to an LLM is stateless, meaning they answer every question as if it is the first time it is asked. The model does not take into account past interactions. As a result, the concept of memory comes into play as a way to give an LLM remembering capabilities to hold a conversation with the model (Kansal, 2024).

Conversation Buffer Window Memory keeps a list of the interactions of the conversation over time. It only uses the last K interactions that can be useful for keeping a sliding window of the most recent interactions, so the buffer does not get too large[2]. In the NeuroKG, the buffer K equals 1, and the prompt to the LLM has the following structure:

1. Prompt template with the main instructions and the context of the LLM.
2. One previous chunk from the text.
3. All generated summaries from the processed chunks.
4. Current chunk from the text.

The Conversation Buffer Window Memory will enable the LLM to have preliminary context from the article and expand the summary on each iteration. Overall, utilizing conversation memory is a good approach for processing large texts and papers because it allows for sequential content analysis without losing track of the topic.

[1] `https://medium.com/@bijit211987/chunking-strategies-for-fine-tuning-llms-30d2988c3b7a`

[2] `https://python.langchain.com/docs/modules/memory/types/buffer_window/`

Additionally, the query to the LLM includes a Prompt Template that we have not described before. Generally, outputs can be improved LLM by prepending prompts—textual instructions and examples of their desired interactions—to LLM inputs. Prompts directly bias the model towards generating the desired outputs (Zamfirescu-Pereira et al., 2023; Pan et al., 2023). Our prompt for the summarization task includes:

1. Task for the Language model to summarize the paper.
2. Space for previous and new chunks.
3. Explanation of how the previous summary was generated.
4. Comments on how the output will be used to create the graph.

Overall, we can say that the use of Prompt Templates and Conversation Memory are effective tools for processing articles when employing LLMs, and they allowed us to analyze all 100 PDF articles. The next step is to transform these small texts into a graph.

4.2 Transforming the summary into the graph

Structuring the text to graph

The LangChain library, which establishes the fundamental infrastructure for working with LLMs for the NeuroKG, includes functionality for creating structured output for our graph. We expect to see the following structures as a result:

```
nodes=[Node(id='Neurological and psychiatric side effects, type=
'Adverse Effect')], rels=[Relationship(source=Node(id='Wishart'),
target=Node(id='Journal of Neurology, Neurosurgery, Psychiatry'),
type='Published In')]
```

The main class we create for structuring is KnowledgeGraph, which inherits from the Pydantic Baseline class. KnowledgeGraph includes additional classes, Nodes and Edges, which also utilize the Properties class. These define the necessary 'key = value' structure for our graph, which is based on the Neo4j graph database. By invoking the structured output method for LLM and passing the

KnowledgeGraph class, the model will add its parameters and output parsers that are necessary to get back the structured output[1].

Moreover, for this task, we developed the Prompt Template to give the model more instructions to help it generate the graph and identify the necessary nodes and relationships in the text. However, prompting LLMs can appear effortless; designing effective prompting strategies requires identifying the contexts in which these LLMs' errors arise, devising prompting strategies to overcome them, and systematically assessing those strategies' effectiveness (Zamfirescu-Pereira et al., 2023). Among the things we specified in this template were the following:

1. General task assignment for the model.
2. An example of the structural result output mentioned above.
3. Requirements for node naming, with examples and prohibitions to use numeric values.
4. Consistency in names, referring as he/she is not allowed, and using acronyms is not allowed.
5. Requirements for working with numbers, quotes, property structure and property name
6. The threat in strict adherence to these requirements: "Adhere to the rules strictly. Non-compliance will result in termination."

Combining the prompt template with structured output generation allows us to effectively transform short paper summaries into structured graphs with nodes and relationships between them. The final step of creating the KG is selecting the best paper or local graph and attaching it to the Neo4j database.

Adding paper graph to the KG

Finally, we transform the custom KnowledgeGraph class we obtained into a Neo4j class. However, LLMs are inherently non-deterministic, so they can generate different graphs based on the exact text. To address this issue, we decided to

[1] `https://python.langchain.com/docs/modules/model_io/chat/structured_output`

```
{
  "identity": 292,
  "labels": [
    "Concept"
  ],
  "properties": {
    "name": "Neuroinflammation",
    "description": "Complex roles in various neurological disorders, emphasizing the interplay among different cell types within the nervous system.",
    "id": "Neuroinflammation",
    "DOI": [
      "10.3389/fncel.2018.00072",
      "10.1186/1742-2094-8-26",
      "10.1007/s12035-014-8657-1"
    ]
  },
  "elementId": "4:ef679da8-6ef7-4189-81c9-d41cb78885a4:292"
}
```


Figure 8: An example of node in JSON representation in the Neo4j graph database.

create several graphs based on one summary and select the most suitable for the NeuroKG.

Primarily, we aim to avoid isolated vertices, which are not connected to other nodes through edges. At the same time, we strive to obtain as many vertices and edges as possible from one text to enrich the NeuroKG with more meaning. Therefore, we set a threshold of 5 isolated nodes. We also tested thresholds 2, 3, and 7 isolated nodes, and threshold 5 turned out to be the best. If one local graph has more vertices than another and does not exceed the threshold of isolated ones, it is included in the resulting graph. Controlling the threshold of isolated vertices and selecting the most suitable local graph are sensible approaches to ensure the quality of structured output. This helps avoid excessive fragmentation and ensures a more comprehensive representation of information from the text in the form of a graph.

Besides, two additional functions involve finding DOIs within the first 2000 tokens of the article and updating the node label according to its name. For these tasks, an LLM was also utilized but did not contain any additional constructs. Inserting DOI values via Cypher queries ensured that a node shared among multiple articles retained all citations in its properties. Finding DOIs allows for preserving the relationship between papers while updating the node label according to the name, ensuring clear differentiation between various types of nodes in the graph. An example of the appearance of the added node in code form is depicted in Fig. 8 and as a graph in Fig. 9.

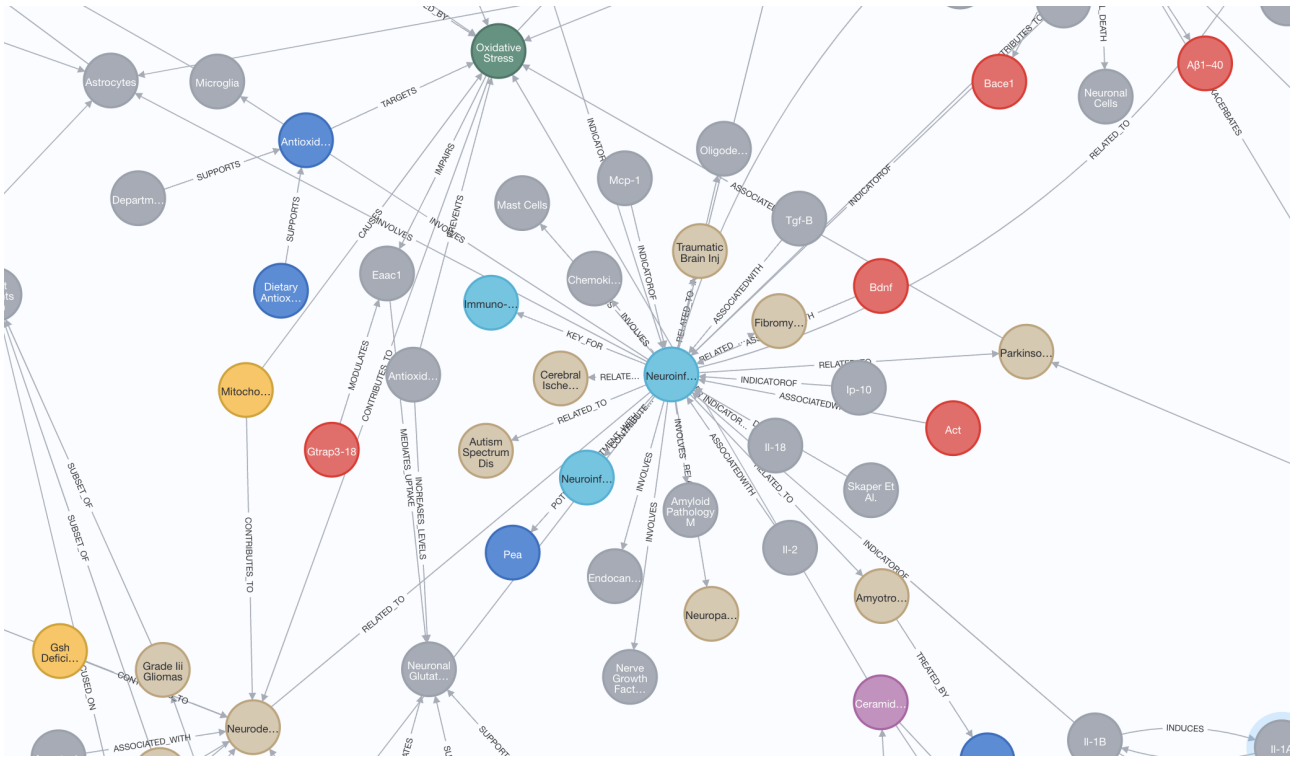


Figure 9: An example of node in visual graph representation in the Neo4j graph database.

It is important to note that LLM also adds other properties, such as description, as shown in Fig. 8. These properties do not have a fixed list between nodes, and the language model itself determines them based on the summarized text of the papers. This can better describe the node and provide more context to its meaning.

Selection of the LLM

The choice of model is an extremely important part of the work because the quality of its performance determines the quality of the KG. It is possible to use both open-source models and rely on the API of a pre-trained LLM for the pipeline. Each approach has its advantages and disadvantages: on the one hand, it is better to train the model exclusively on scientific articles for potentially better summary generation, but this requires a large amount of data resources, hardware costs, and time for training. On the other hand, relying on an API is much simpler and faster, but it is also more expensive and less accurate due to the generalization of large models.

The primary model used in this work is the OpenAI LLM model 'gpt-4-turbo-

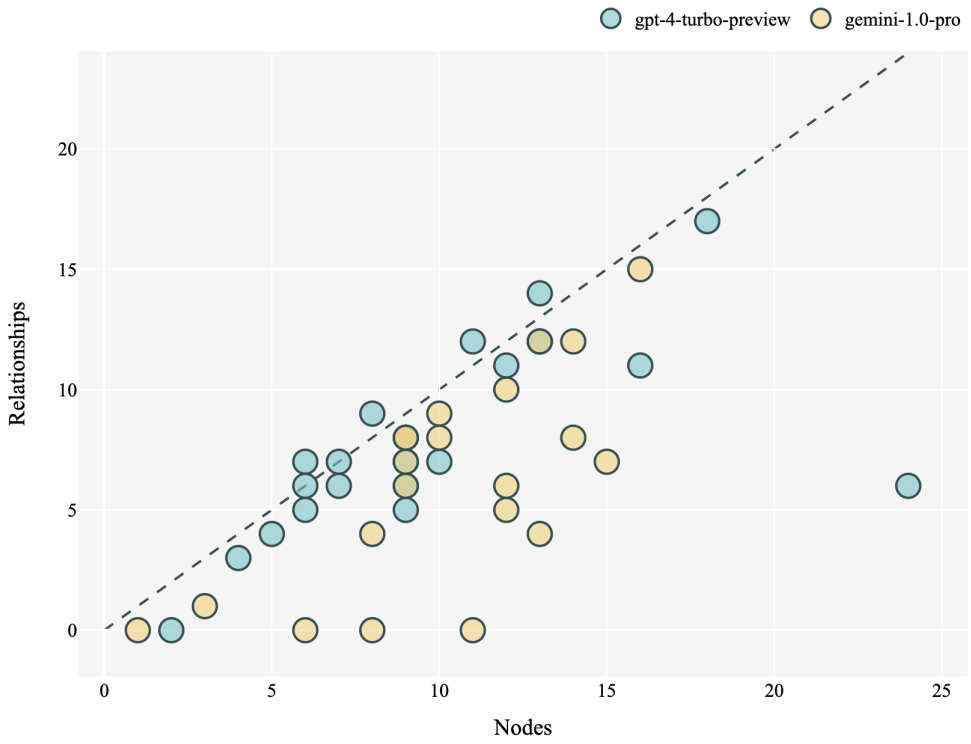


Figure 10: Comparison of the generation of the number of vertices and edges from the summary of the 2 LLM models: 'gpt-4-turbo-preview' (blue) and 'gemini-1.0-pro' (yellow).

preview', which was considered state-of-the-art at the time of writing the paper. However, the 'gemini-1.0-pro' model was also tested. The comparison factor of these models was the ratio of the number of vertices to the number of edges and the volume of isolated vertices. Isolated vertices are nodes not connected to other nodes by an edge. This is critical for the graph because edges transform raw information in the form of vertices into knowledge. Additionally, we are also interested in obtaining more information from a single article.

In Fig. 10, the comparison of the performance of two models in generating a summary, which was then transformed into a local graph by one OpenAI model, can be observed. The plot describes the ability of each model to better process a full article. Each point represents the generated number of vertices-edges from one summary article. Here, the GPT-4 model outperformed Gemini for the following reasons:

1. Gemini produced more local graphs with several vertices and 0 edges;

2. Gemini generated fewer edges overall in the local graphs;

3. GPT-4 produced more graphs with values close to the diagonal, indicating an equal number of edges and vertices;

4. Gemini generated more isolated vertices due to deviation from the diagonal towards the vertex axis.

This analysis method can evaluate which model will accumulate more isolated vertices based on a small number of local graphs. As a result, generating a complete graph from 100 articles was performed using the OpenAI LLM model.

Table 1: Table of applied graph metrics.

Metric	Value
Number of the nodes	1443
Number of the edges	1164
Number of the isolated nodes	276
Diameter of the graph	13
Number of node labels	219
Number of edges types	324

5 Analysis

Now, after constructing a complete knowledge graph, we can proceed to its analysis. The first stage is a general overview using graph metric theory to understand the size, number of connections, types of vertices, etc. The second stage involves more complex Cypher queries for analyzing relationships between papers, distances between different vertices, and examples of long paths in the graph.

5.1 Graph Metrics

In Table 1, the main metrics of the graph are presented. Processing 100 articles with full text resulted in 1443 vertices, most of which are connected by 1164 edges. In Fig. 1, it can be noticed that multiple edges connect some vertices. This occurs because the same nodes are found in more than one paper. Mostly, such connections are synonymous, but they were deliberately left to preserve information because such cases can be rarely seen. For example, between the vertices `"Neuroinflammation"` and `"Alzheimer's Disease"` are three connections: `"RELATED_TO," "CONTRIBUTES_TO," "ASSOCIATED_WITH."`

On the other hand, the graph comprises 276 isolated nodes, which are not

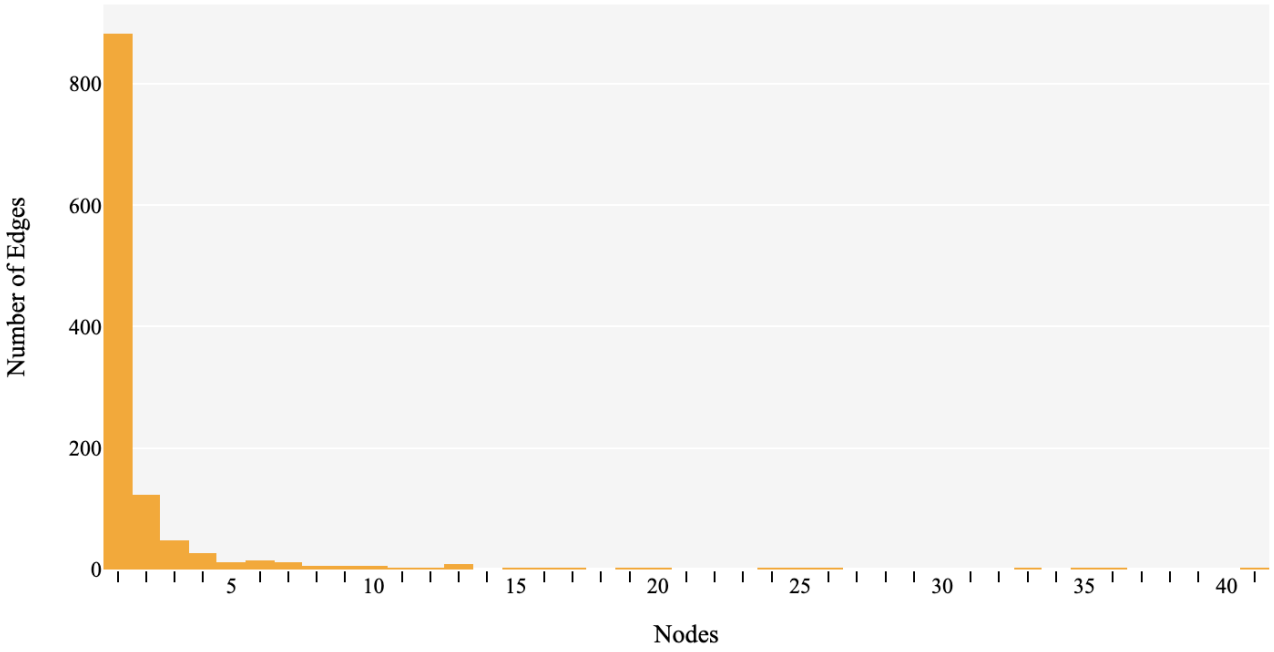


Figure 11: Distribution of the node degree in the NeuroKG.

sufficient for analysis. These nodes arise when creating local graphs from individual articles and accumulate throughout the processing of the entire literature. That is the reason why an algorithm to compare local graphs was introduced to find larger ones with a smaller number of isolated vertices. Local graphs with 15 vertices and 0 edges would be catastrophic for the knowledge graph. In contrast, we believe that these isolated nodes will be utilized as the number of processed articles increases.

Fig. 11 describes the node degree and the number of edges that are connected to this vertex. As expected, we cannot see the normal distribution because a significant number of nodes have only one edge. This is related to the model's behaviour when it selects a central value from the summarized text and builds connections around it. Fig. 12 clearly demonstrates this output structure. However, the knowledge graph contains higher node degrees and even reaches the number of 44 edges.

Another important metric in graph theory is the concept of diameter. The diameter is the shortest distance between the most distant nodes in the graph. In other words, there is no smaller path between these two nodes, and for all other vertices, the shortest path does not exceed the diameter value. For our knowledge graph, the diameter is 13, indicating significant connections between

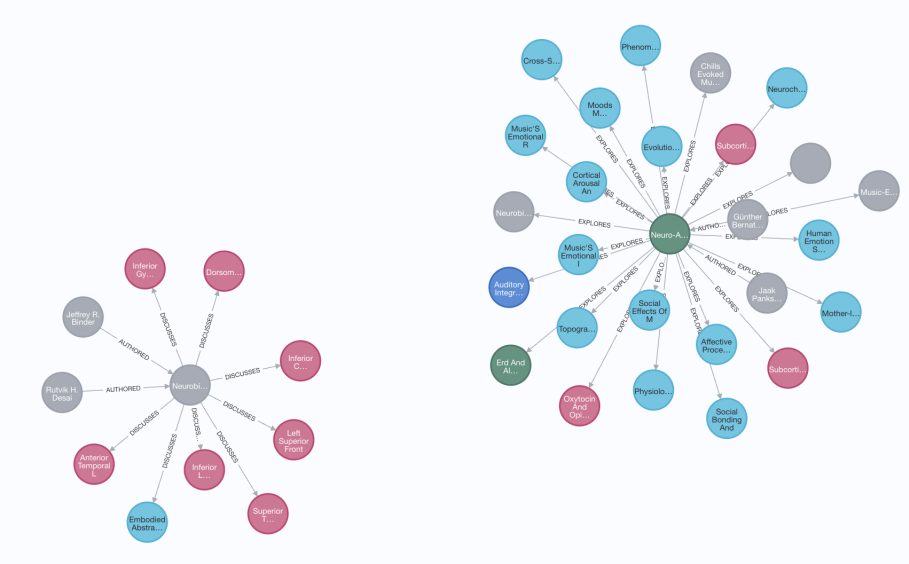


Figure 12: An example of local graphs with a high concentration of node degrees of one.

different articles. The graph's diameter is important for assessing the scale of relationships between knowledge and understanding how far-reaching influences or information can be.

Another technique that we used to describe the knowledge graph is the PageRank. This algorithm measures the importance of each node within the graph based on the number of incoming relationships and the importance of the corresponding source nodes. The underlying assumption, roughly speaking, is that a page is only as important as the pages that link to it[1]. PageRank is an in-built method in the Neo4j Graph Data Science library which provides efficiently implemented, parallel versions of common graph algorithms, exposed as Cypher procedures[2]. As can be seen from Fig. 13, the most important vertices in the graph are `Alzheimer's Disease`, `Gliomas`, and `Neuroinflammation`. Despite the instructions for the LLM in the Prompt Template not to select authors, article titles, and abbreviations, they can still be seen among the list of nodes. Nevertheless, this algorithm allows us to assess the major neuroscientific themes in the graph, and with future improvements to the graph generation process, it will be possible to avoid inappropriate vertices.

Another key property for a vertex in the graph is its type or label. It should

[1] `https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/`

[2] `https://neo4j.com/docs/graph-data-science/current/introduction/`

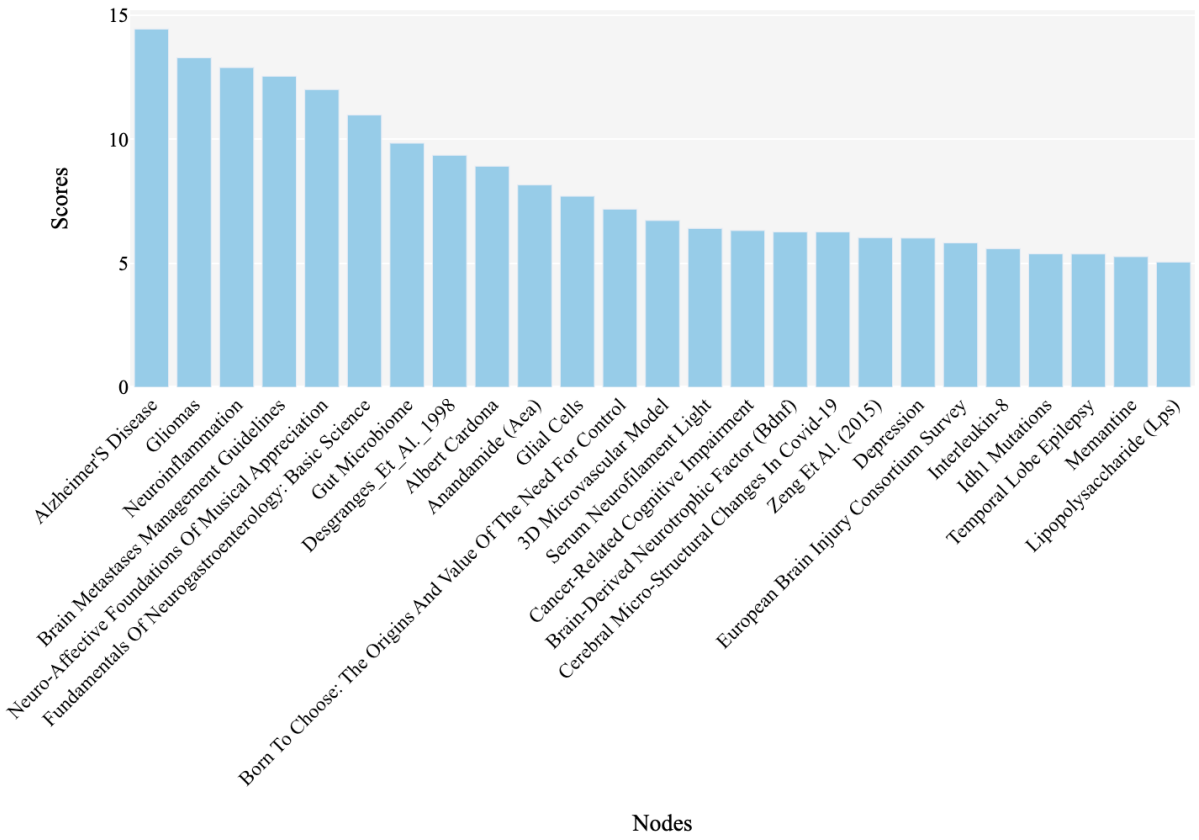


Figure 13: Distribution of the nodes' highest PageRank scores in the NeuroKG.

properly describe the node's topic so that in future analysis, vertices from the same domain can be grouped together. In Table 1, it can be seen a total of 219 various labels in our knowledge graph. Fig. 14 shows the distribution of the labels. They describe various vertices broadly, and among the labels, we can see `"Disease,"` `"Protein,"` `"Gene,"` and `"Cell type."` On the other hand, the most common label is `"Concept,"` which ambiguously describes nodes. As mentioned in the section on preparing the graph, label generation was performed separately using LLM, as for most structured outputs, the label value was simply "Node." Overall, creating accurate labels is necessary for a qualitative interpretation of relationships in the graph and for more complex analysis.

For the edges of the graph, there are 324 different types according to Table 1. Additionally, in Fig. 15, we see their distribution. Their names are quite generic and do not clearly and comprehensively describe the relationships between vertices, although names like `"AFFECTS,"` `"CAUSES,"` and `"IMPROVES"` can be highlighted. Nevertheless, the presence of an edge between vertices is more important, even if they only generically describe the relationships between them.

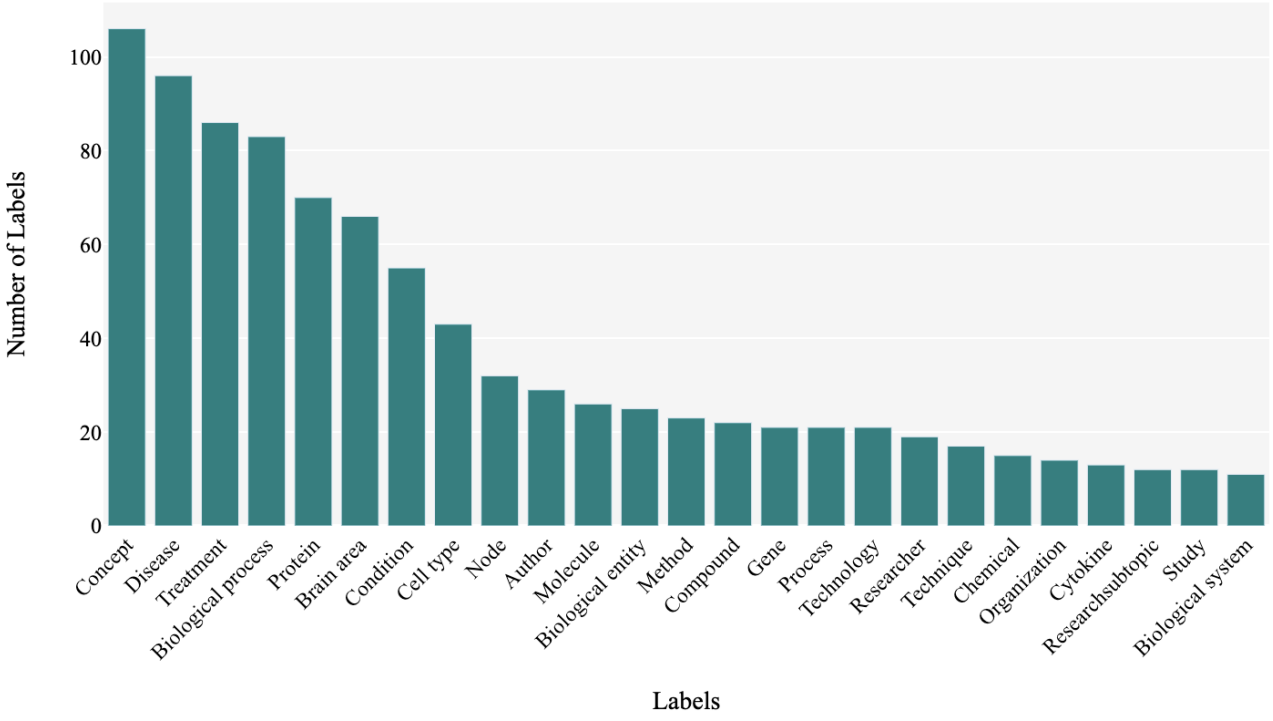


Figure 14: Distribution of the highest number of nodes labels in the NeuroKG.

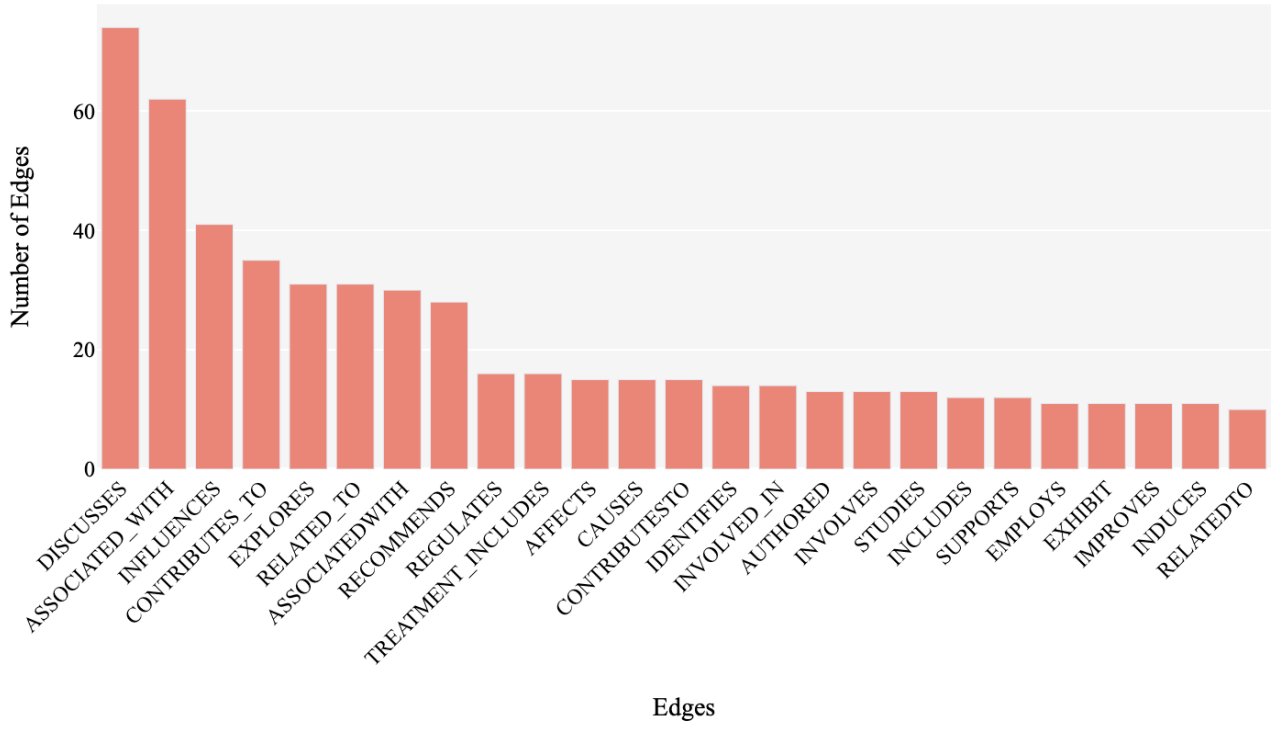
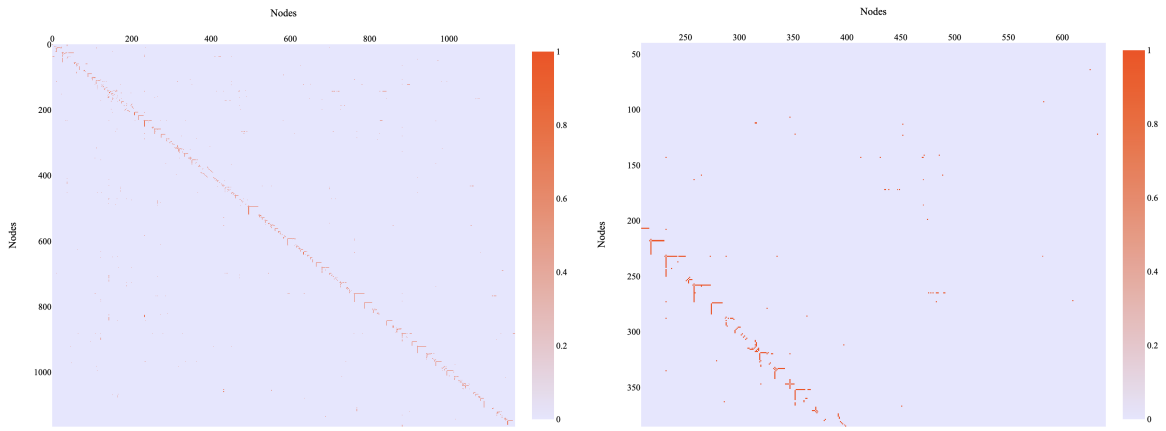


Figure 15: Distribution of the highest number of edges type in the NeuroKG.



(a) Full view (b) Enlarged view

Figure 16: The adjacency matrix of the nodes and edges in the NeuroKG.

Another important algorithm in the metric graph theory, and the last one for this section, is the adjacency matrix. The adjacency matrix, sometimes also called the connection matrix, of the labelled graph is a matrix with rows and columns labelled by the vertices, with a 1 or 0 according to whether nodes are adjacent or not[1]. In Fig. 16a can be seen the whole adjacency matrix of the knowledge graph. Most of the connections are brightly seen at the centre diagonal, meaning that most of the nodes are connected to the vertices from the same paper.

But, after looking more precisely, we can see on Fig. 16b orange dots outside of the main diagonal. It is how nodes connect to the other nodes from separate papers. This demonstrates the effectiveness of our processing algorithm and clearly illustrates the efficiency of linking different papers using the knowledge graph.

5.2 Complex Queries

Now, we will consider more neuroscientific queries to explore the knowledge graph to understand how effectively it is constructed and what knowledge it can impart.c

The graph contains a total of 66 nodes with the label 'Brain area,' so it is interesting to see how they are connected to other nodes and interact with each other. Fig. 17 illustrates the relationship of the 'Brain area' node with any other. This enables comparison of any vertex labels, which is why naming them appropriately is crucial. Such information can be useful for studying brain functions,

[1] `https://mathworld.wolfram.com/AdjacencyMatrix.html`

pathologies and other aspects of neurobiology. In Appendix B, the connections between brain areas and other vertices in Table 3 and the inverse relationship in Table 2.

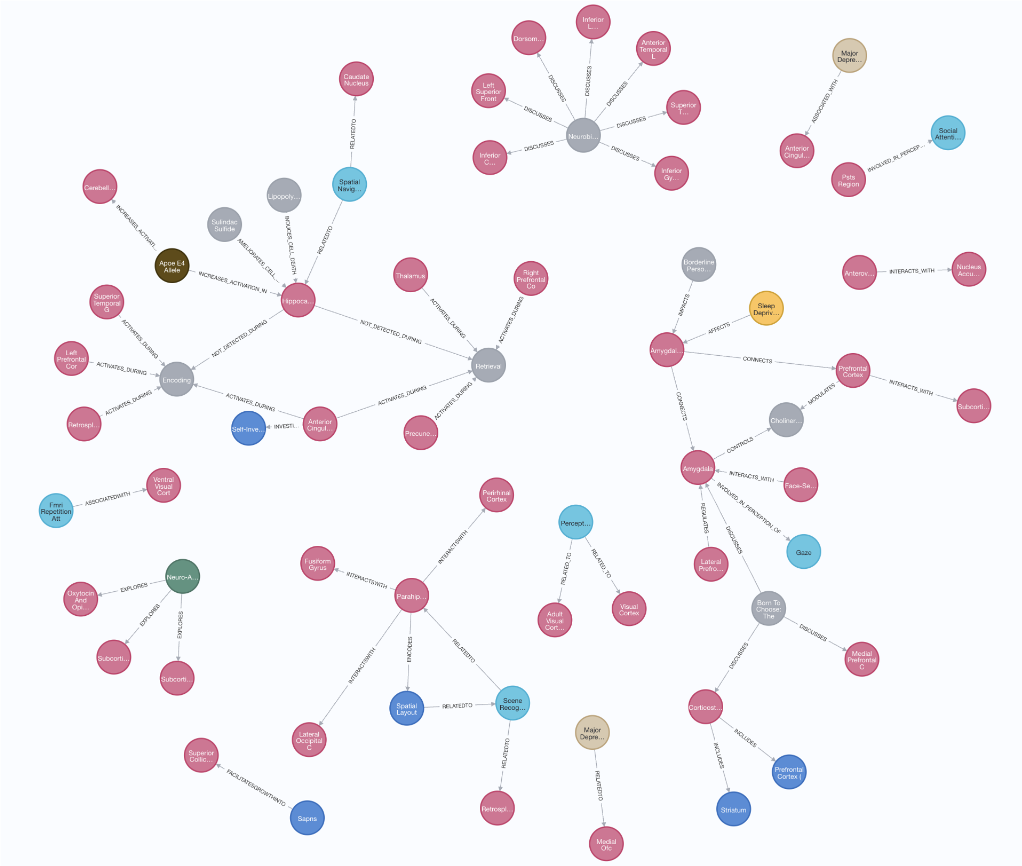


Figure 17: Graph representation of the sorted connection between ‘Brain area‘ label nodes and other nodes in the NeuroKG.

In addition, we can find a path between two vertices at a certain distance to understand how they are connected through other vertices. Fig. 18 depicts the path from the node ’Neuroinflammation’ to ’Brain Metastases’ with a path size ranging from 1-6. Such a query allows us to visually see which vertices and relationships connect the two terms, which is necessary for understanding the relation between distant fields, such as genetics and behaviour. Moreover, identifying the path between nodes can reveal meaningful connections or pathways for information transmission in the system.

Another vital way to analyze the graph is to identify long paths within it. This helps understand the scale of the graph and uncover valuable connections between vertices. For example, Fig. 19 demonstrates a query example searching for paths of length 5 edges and outputting the first 50 vertices. By analyzing long

paths, we can link various terms from articles and uncover hidden connections between them.

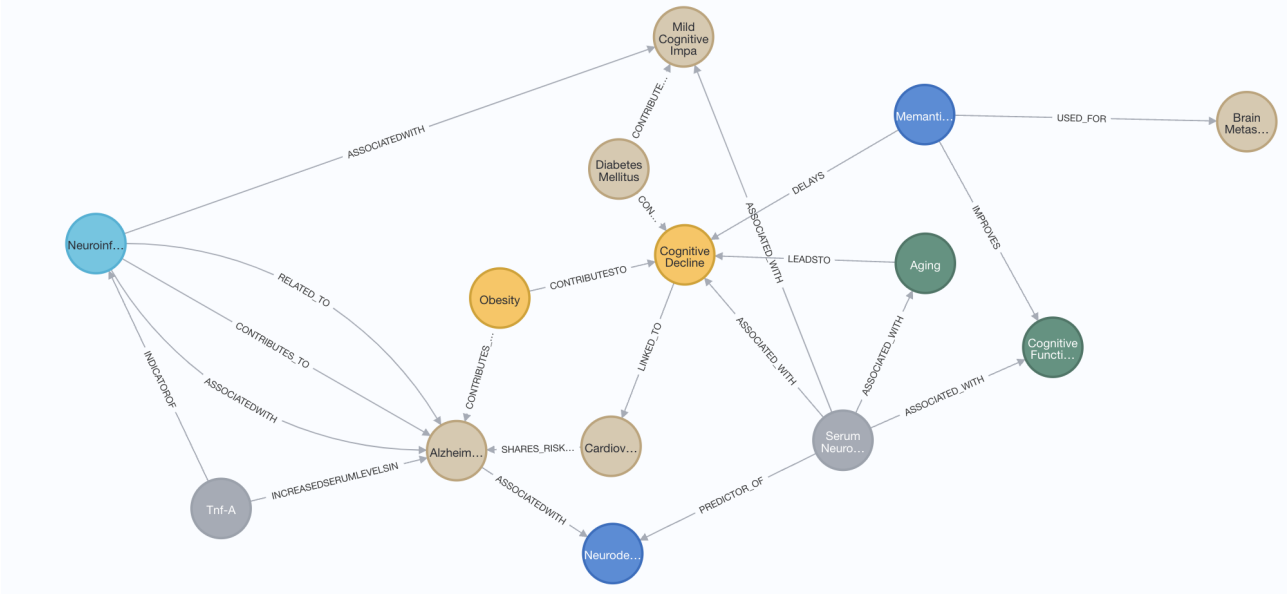


Figure 18: Graph representation of the path between two selected nodes in the NeuroKG.

Finally, another type of query that LLM can handle will be discussed. In the Langchain library, querying the graph[1] is implemented, where natural language queries can be used to retrieve results from the graph also in a natural language format.

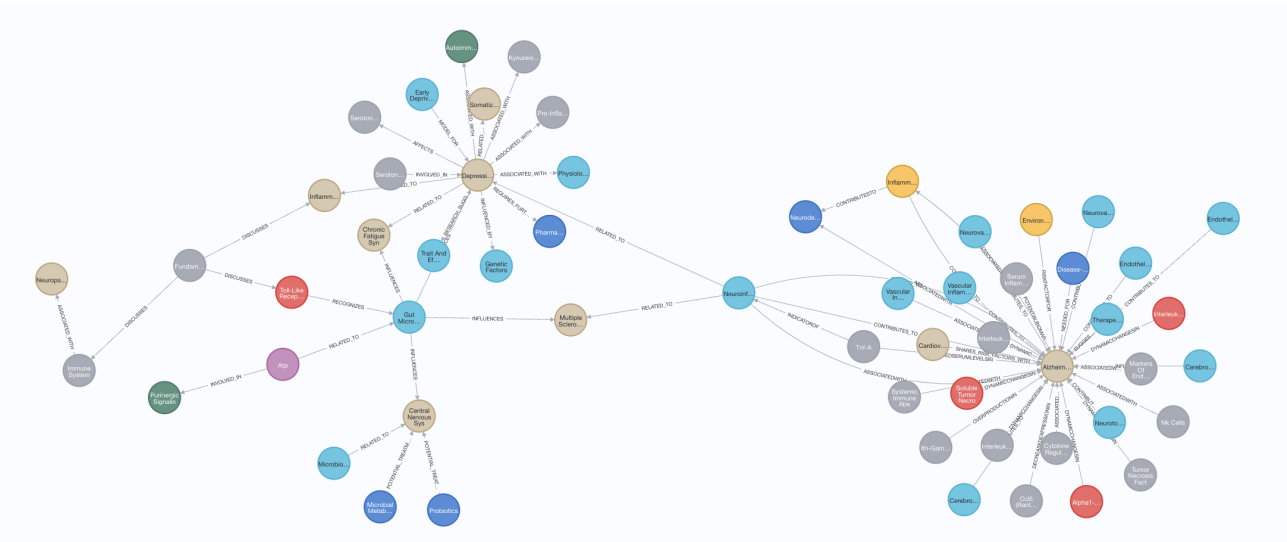


Figure 19: Graph representation of the path with defined length in the NeuroKG.

[1] https://python.langchain.com/docs/integrations/graphs/neo4j_cypher/

This would be very convenient for users with limited technical knowledge who find it harder to write Cypher queries independently. However, my attempts to use such queries often ended unsuccessfully. Due to the strict syntax of the Cypher language, it is almost impossible to ask questions about data that are missing in the graph or incorrectly define node labels. This is a very promising direction, but it is not developed enough at this moment.

Conclusion

We have explored how LLM can be used to create KGs in the context of neuroscientific literature. Despite existing limitations, such synergy between graph and natural language processing holds promising applications in science. Not only does it allow one to analyze more literature in less time and enables researchers to fully engage with new discoveries, but it also integrates various fields into a single graph structure for exploring connections and gaining insights. We hope that this study will contribute to unravelling the mystery of human brain function for researchers.

References

Yeung, A. W. K.; Goto, T. K.; Leung, W. K. The Changing Landscape of Neuroscience Research, 2006–2015: A Bibliometric Study. *Frontiers in Neuroscience* **2017**, *11*.

Landhuis, E. Scientific literature: Information overload. *Nature* **2016**, *535*, 457–458.

Van Noorden, R. Scientists may be reaching a peak in reading habits. *Nature* **2014**,

Arturo Casadevall, Ferric C. Fang Specialized Science. *Infect Immun.* **2014**, *82*, 1355–1360.

Wilkinson, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* **2016**, *3*, 160018.

Markiewicz, C. J.; Gorgolewski, K. J.; Feingold, F.; Blair, R.; Halchenko, Y. O.; Miller, E.; Hardcastle, N.; Wexler, J.; Esteban, O.; Goncavles, M.; Jwa, A.; Poldrack, R. The OpenNeuro resource for sharing of neuroscience data. *eLife* **2021**, *10*, e71774.

Sunkin, S. M.; Ng, L.; Lau, C.; Dolbeare, T.; Gilbert, T. L.; Thompson, C. L.; Hawrylycz, M.; Dang, C. Allen Brain Atlas: an integrated spatio-temporal portal for exploring the central nervous system. *Nucleic Acids Res* **2012**, *41*, D996–D1008.

Appukuttan, S.; Bologna, L. L.; Schürmann, F.; Migliore, M.; Davison, A. P. EBRAINS Live Papers - Interactive Resource Sheets for Computational Studies in Neuroscience. *Neuroinformatics* **2023**, *21*, 101–113.

Hogan, A. et al. Knowledge Graphs. *ACM Computing Surveys* **2021**, *54*, 1–37.

Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *ArXiv* **2023**, *abs/2306.08302*.

Zhong, L.; Wu, J.; Li, Q.; Peng, H.; Wu, X. A Comprehensive Survey on Automatic Knowledge Graph Construction. 2023.

Yang, J.; Jin, H.; Tang, R.; Han, X.; Feng, Q.; Jiang, H.; Yin, B.; Hu, X. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. 2023.

Lewis, P. S. H.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; Riedel, S.; Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *CoRR* **2020**, *abs/2005.11401*.

Kumar, A.; Pandey, A.; Gadia, R.; Mishra, M. Building Knowledge Graph using Pre-trained Language Model for Learning Entity-aware Relationships. 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON). 2020; pp 310–315.

Buehler, M. J. Accelerating Scientific Discovery with Generative Knowledge Extraction, Graph-Based Representation, and Multimodal Intelligent Graph Reasoning. 2024.

Kansal, A. *Building Generative AI-Powered Apps: A Hands-on Guide for Developers*; Apress: Berkeley, CA, 2024; pp 41–57.

Zamfirescu-Pereira, J.; Wong, R. Y.; Hartmann, B.; Yang, Q. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. New York, NY, USA, 2023.

Appendix A

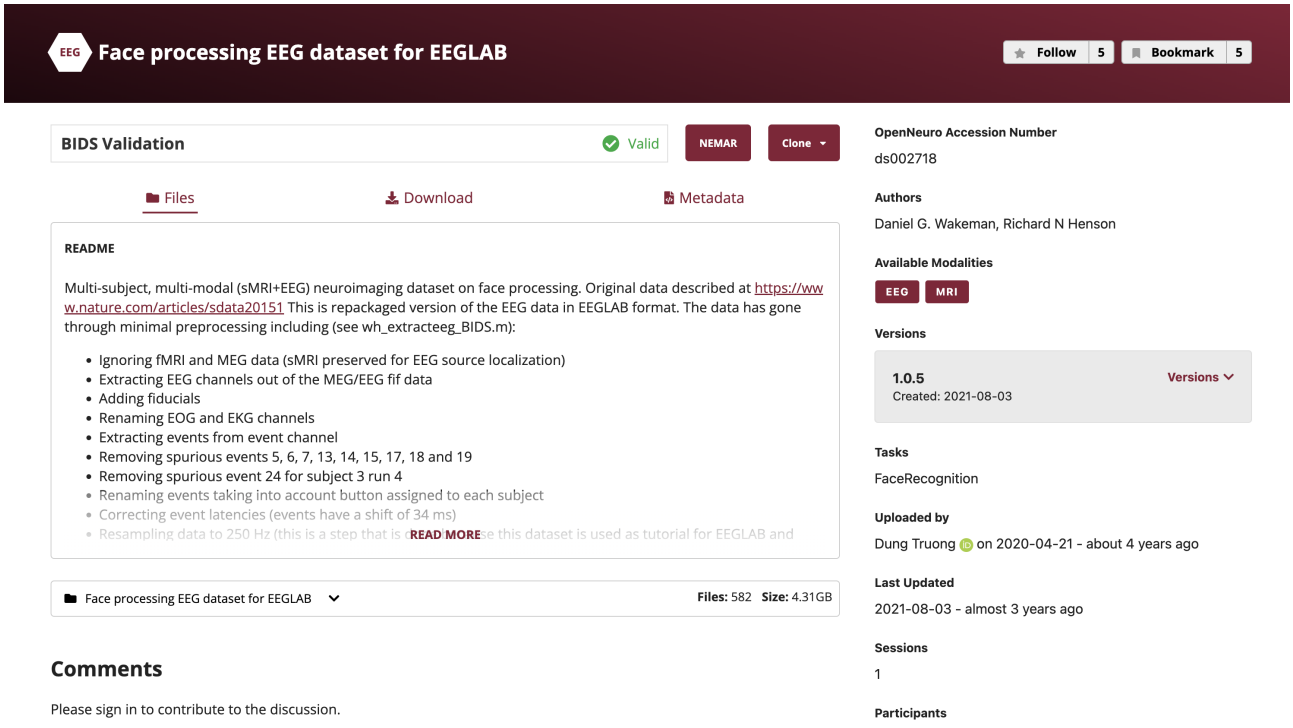


Figure 20: An example of a dataset in the OpenNeuro platform.

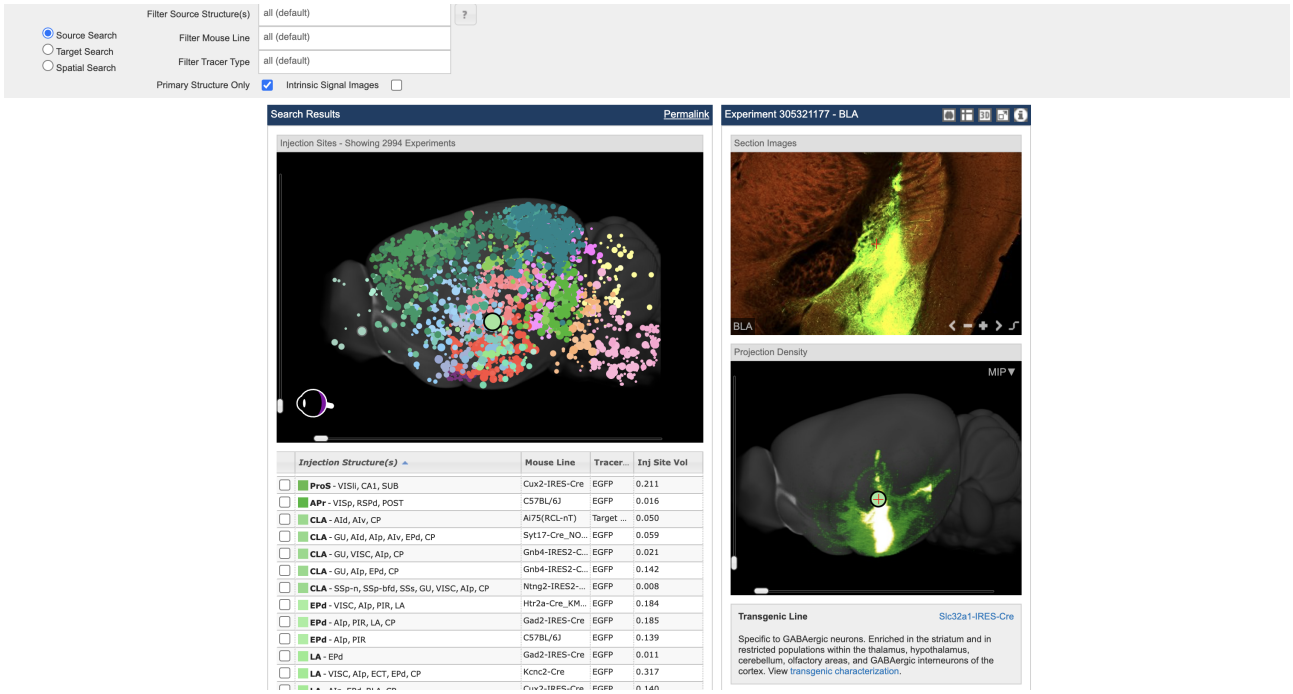


Figure 21: An example of data and its visualization (cell types and their location in the brain) in the Allen Brain Atlas platform.

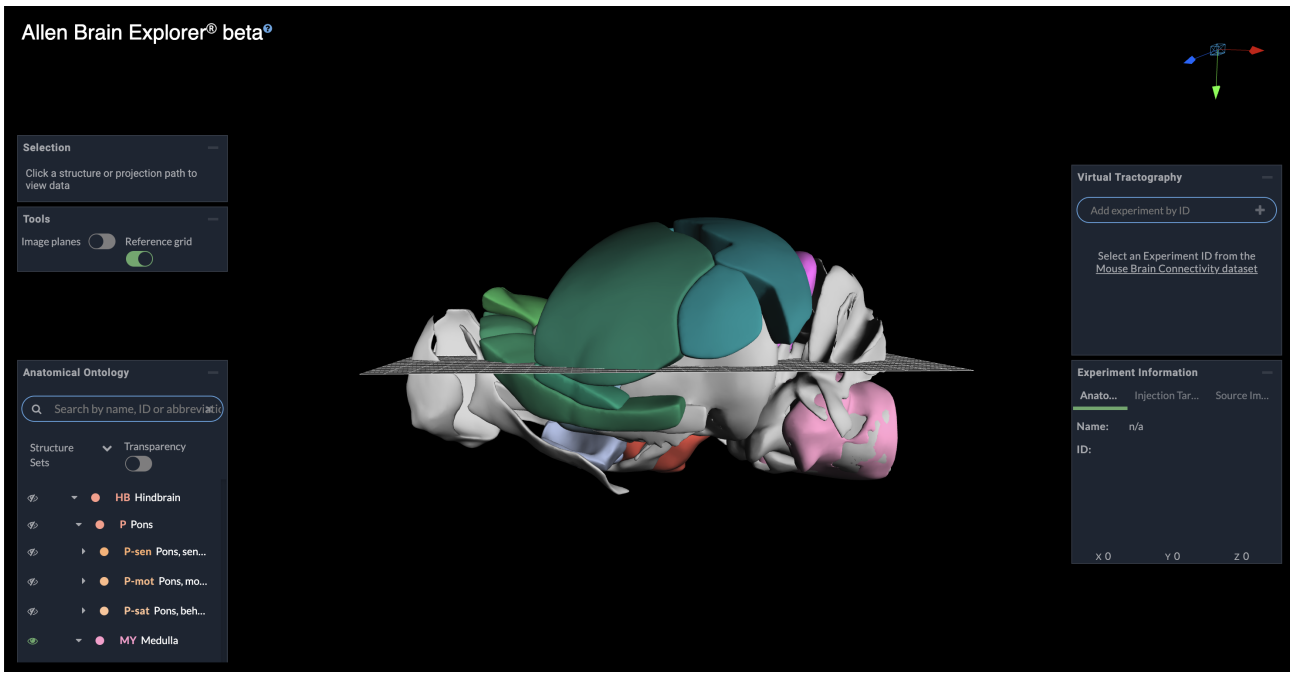


Figure 22: An example of a 3D simulation or constructor of brain regions in the mouse brain in the Allen Brain Atlas platform.

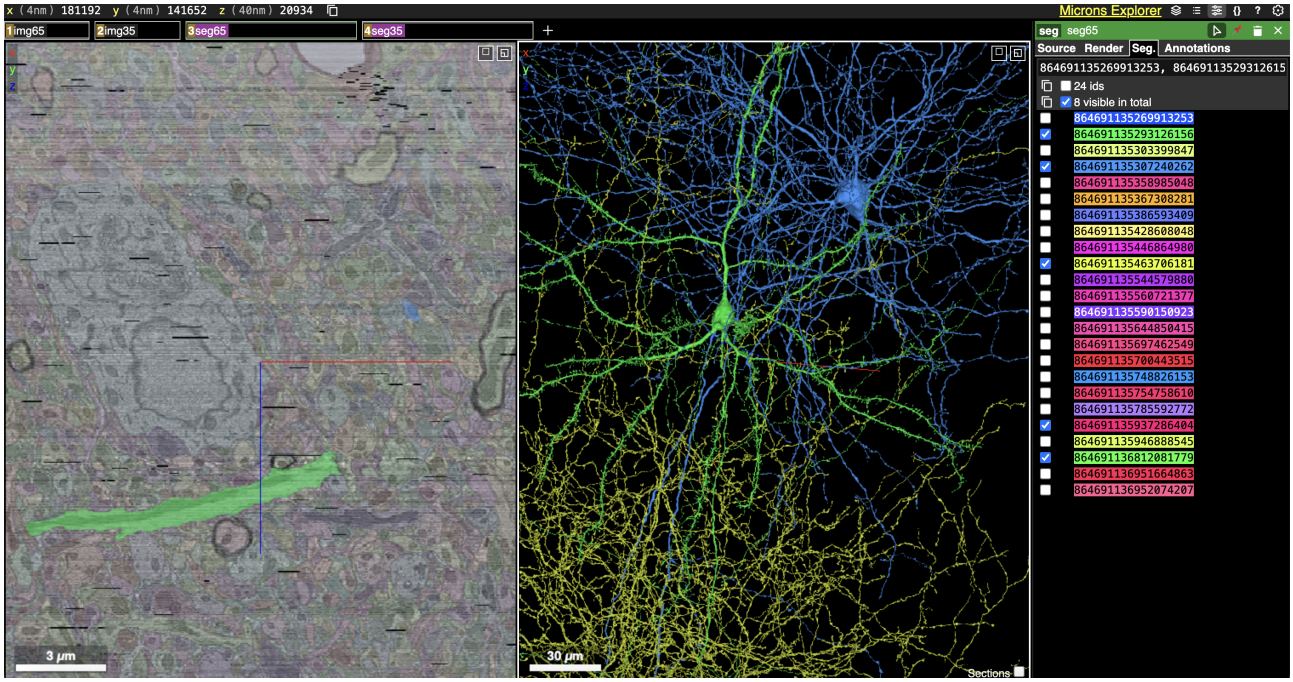


Figure 23: The An example of classification of structures on a brain slice and reconstruction of the found neurons as 3D models in the Allen Brain Atlas platform.

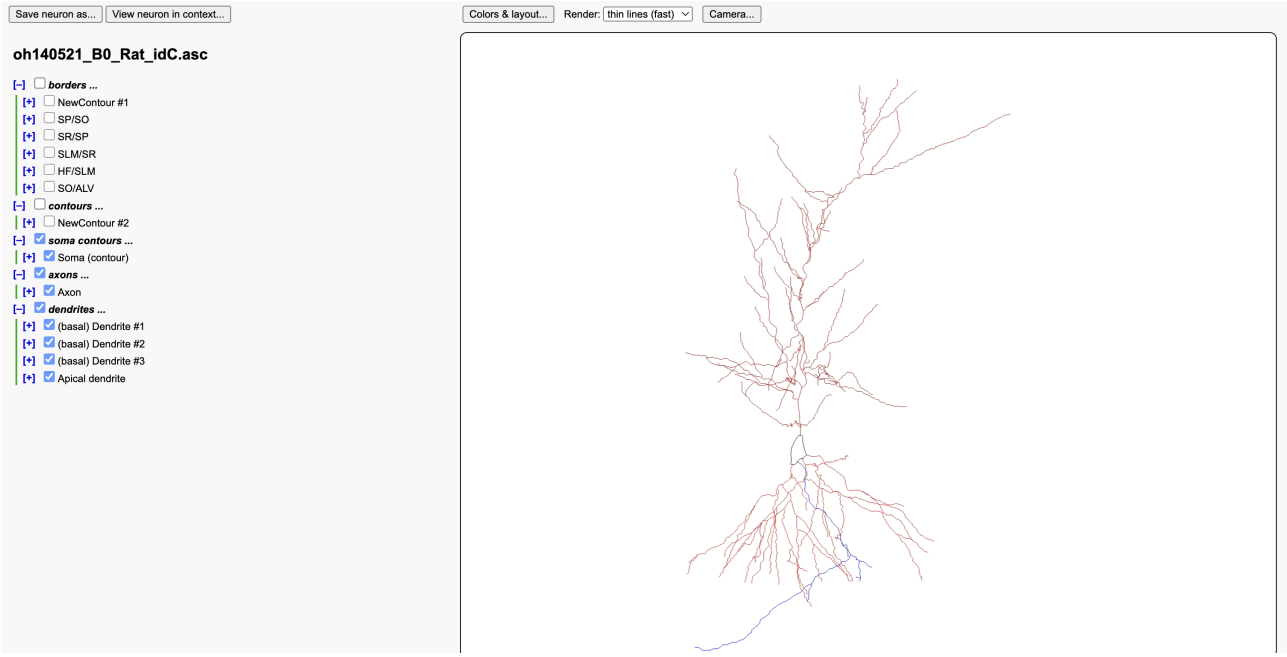


Figure 24: An example of an interactive 3D model of a neuron in the EBRAINS Live Papers.

Appendix B

Table 2: Results of query in the direction Brain area -> Relationship -> Node.

Brain area	Relationship	Node
"Hippocampus"	"NOT_DETECTED_DURING"	"Retrieval"
"Hippocampus"	"NOT_DETECTED_DURING"	"Encoding"
"Amygdala"	"INVOLVED_IN_PERCEPTION_OF"	"Gaze"
"Prefrontal Cortex"	"INTERACTS_WITH"	"Subcortical Regions"
"Lateral Prefrontal Cortex"	"REGULATES"	"Amygdala"
"Anteroventral Prefrontal Cortex"	"INTERACTS_WITH"	"Nucleus Accumbens"
"Amygdala-Prefrontal Pathway"	"CONNECTS"	"Amygdala"
"Amygdala-Prefrontal Pathway"	"CONNECTS"	"Prefrontal Cortex"
"Face-Sensitive Fusiform Cortex"	"INTERACTS_WITH"	"Amygdala"
"Psts Region"	"INVOLVED_IN_PERCEPTION_OF"	"Social Attention"
"Left Prefrontal Cortex"	"ACTIVATES_DURING"	"Encoding"
"Retrosplenial Area Of The Cingulate Cortex"	"ACTIVATES_DURING"	"Encoding"
"Precuneus"	"ACTIVATES_DURING"	"Retrieval"
"Right Prefrontal Cortex"	"ACTIVATES_DURING"	"Retrieval"
"Anterior Cingulate Cortex"	"ACTIVATES_DURING"	"Encoding"
"Anterior Cingulate Cortex"	"ACTIVATES_DURING"	"Retrieval"
"Thalamus"	"ACTIVATES_DURING"	"Retrieval"
"Superior Temporal Gyri"	"ACTIVATES_DURING"	"Encoding"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Perirhinal Cortex"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Fusiform Gyrus"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Lateral Occipital Complex"

Table 3: Results of query in the direction Node -> Relationship -> Brain area.

Node	Relationship	Brain area
"Perceptual Learning"	"RELATED_TO"	"Adult Visual Cortex"
"Perceptual Learning"	"RELATED_TO"	"Visual Cortex"
"Spatial Navigation"	"RELATED_TO"	"Hippocampus"
"Lipopolysaccharide (Lps)"	"INDUCES_CELL_DEATH"	"Hippocampus"
"Sulindac Sulfide"	"AMELIORATES_CELL_DEATH"	"Hippocampus"
"Apoe 4 Allele"	"INCREASES_ACTIVATION_IN"	"Hippocampus"
"Face-Sensitive Fusiform Cortex"	"INTERACTS_WITH"	"Amygdala"
"Amygdala-Prefrontal Pathway"	"CONNECTS"	"Amygdala"
"Lateral Prefrontal Cortex"	"REGULATES"	"Amygdala"
"Apoe 4 Allele"	"INCREASES_ACTIVATION_IN"	"Cerebellum"
"Amygdala-Prefrontal Pathway"	"CONNECTS"	"Prefrontal Cortex"
"Prefrontal Cortex"	"INTERACTS_WITH"	"Subcortical Regions"
"Anteroventral Prefrontal Cortex"	"INTERACTS_WITH"	"Nucleus Accumbens"
"Borderline Personality Disorder"	"IMPACTS"	"Amygdala-Prefrontal Pathway"
"Sleep Deprivation"	"AFFECTS"	"Amygdala-Prefrontal Pathway"
"Major Depression"	"ASSOCIATED_WITH"	"Anterior Cingulate Gyrus"
"Sapns"	"FACILITATES_GROWTH$_INTO$"	"Superior Colliculus"
"Major Depressive Disorder"	"RELATED_TO"	"Medial Ofc"
"Fmri Repetition Attenuation"	"ASSOCIATED_WITH"	"Ventral Visual Cortex"
"Spatial Navigation"	"RELATED_TO"	"Caudate Nucleus"
"Scene Recognition"	"RELATED_TO"	"Parahippocampal Place Area (Ppa)"
"Scene Recognition"	"RELATED_TO"	"Retrosplenial Complex (Rsc)"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Lateral Occipital Complex"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Fusiform Gyrus"
"Parahippocampal Place Area (Ppa)"	"INTERACTS_WITH"	"Perirhinal Cortex"