Computational Literature Mining to Identify and Characterize Small Molecule Compounds with Neuroprotective and Neurotrophic Effects

Bachelor Semester Project S3 (Academic Year 2024/25), University of Luxembourg

Oleksandr Yeroftieiev
oleksandr.yeroftieiev.001@student.uni.lu
University of Luxembourg
Luxembourg

Enrico Glaab
enrico.glaab@uni.lu
University of Luxembourg
Luxembourg

ACM Reference Format:
Oleksandr Yeroftieiev and Enrico Glaab. 2025. Computational Literature Mining to Identify and Characterize Small Molecule Compounds with Neuroprotective and Neurotrophic Effects: Bachelor Semester Project S3 (Academic Year 2024/25), University of Luxembourg. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Abstract

This project aims to create a user-friendly database of small molecule compounds with neuroprotective and neurotrophic effects. Large Language Models (LLMs) are used to extract data on neuroprotective and neurotrophic compounds from the scientific literature, and then the information is organized into a database according to the following criteria: therapeutic type, disease target, mechanism of action, evidence description, and so on, and other parameters.

As part of the project, I pre-process articles from PubMed using Python: I remove unnecessary characters, split the text into sentences and highlight relevant content based on a list of keywords ("neuroprotection", "neuroprotective", "neurorescue", "neurorestoration", "neuroregeneration", etc.). After that, using OpenAI's LLM (GPT), I create a prompt and extract key data about small molecule compounds. The extracted information is immediately saved in a convenient format to a database (MongoDB) for further use.

I also developed a web application based on the Django framework for convenient small molecule searching. This application allows quick access to the information in the database through a simple search box.

This paper also discusses possible future improvements and scaling of the project, discusses motivation, provides examples of similar databases created previously, and compares the final results with the original research articles to assess the relevance and accuracy of the extracted information.

2 Motivation

The motivation for this project was the lack of a specialized database dedicated to neuroprotective and neurotrophic compounds or their unavailability in the public domain. Well-known databases, such as ChEMBL [1], which provides information on chemical structures, biological activity and pharmacological properties, or DrugBank [2], can serve as an excellent example of how to properly organize and structure data. However, a similar platform focused exclusively on neuroprotective and neurotrophic compounds does not exist to date.

There are also databases such as PubMed, Web of Science and Scopus that provide extensive information on various scientific disciplines, including neurobiology. However, they do not offer specialized tools for searching and analyzing data related to neuroprotective and neurotrophic compounds. For example, information on the properties of a single compound may be scattered across several articles, each covering only selected aspects: molecular mechanisms, results of in vitro or in vivo experiments, or clinical trial data.

As part of this project, I decided to create a prototype database (based on the analysis of information from 1000 scientific articles) that combines and structures data from various sources in a convenient and understandable form. Such a database will include a confidence score metric, which will allow us to assess the reliability and relevance of information. This is especially important if the data from different articles contradict each other or, on the contrary, confirm the same facts, which will help the user to make informed conclusions. In addition, the addition of a web application will provide the user with a convenient tool to quickly search the database for information of interest.

, ,

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

3 Literature Retrieval

To accomplish this project, the first step was to retrieve 1000 scientific articles from PubMed. The main goal was to collect relevant information on scientific studies related to neuroprotective and neurotrophic compounds. However, given the huge amount of available literature and the inefficiency of performing this task manually, the data search and retrieval process was automated using a Python script and PubMed API [3].

The Entrez Programming Utilities (E-utilities) tool was used to interact with the PubMed API to provide convenient functionality for searching, retrieving and processing scientific articles. This tool allows to flexibly customize search parameters, extract article metadata (titles, abstracts, etc.) and save them for further work.

```python
def download_article(pmc_id, output_dir):

    response = requests.get(f"{BASE_URL}/efetch.fcgi", params={
        "db": "pmc",
        "id": pmc_id,
        "rettype": "full",
        "retmode": "xml"
    })
```

Figure 1

It is important to note that not all articles submitted to PubMed provide access to complete information due to copyright restrictions. Many articles are only available with limited metadata such as title, abstract, and author names. To work with full text articles, the PMC Open Access Subset[4] section created by PubMed was used.

This section includes articles available under Creative Commons or similar licenses, allowing them to be freely distributed and used for scientific purposes.

4 Text Preprocessing and Sentence Extraction

Python was used to perform this task. In the cleanup step, the text was processed to remove unwanted characters such as special characters and service characters. Only letters, numbers, spaces and basic punctuation marks were left. Consecutive spaces were replaced by single spaces, simplifying the structure of the text and preparing it for analysis.

After cleaning, each sentence was checked for at least one keyword. If a keyword was found, the sentence was considered relevant and added to a separate list for further work.

The **BeautifulSoup** library was used to extract the article title, abstract and main text from XML files BeautifulSoup was choosed because :

- **Intuitive interface:** BeautifulSoup provides an easy way to work with XML and HTML files. It allows you to quickly find relevant document elements such as <article-title>, <abstract>, and <body> tags that contain key information.
- **Flexibility:** The library supports a wide range of parsing functions, including searching for items by tags, attributes, or content, making it easy to retrieve the data you need.
- **Wide format support:** BeautifulSoup works with a variety of formats such as XML, making it ideal for working with data from PubMed.

```python
results.append({
    "file": file_name,
    "title": title,
    "sentences": " | ".join(relevant_sentences)
})
```

Figure 2

After that the whole list was saved to the main project folder in csv format for further LLM processing.

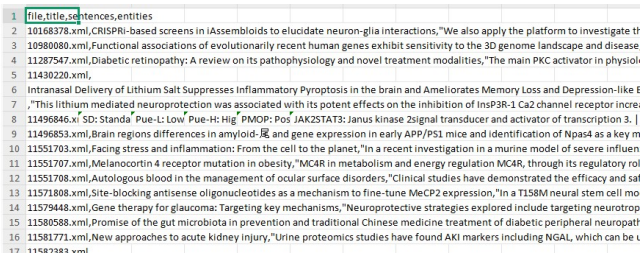

Figure 3

5 Large Language Model–Assisted Data Extraction

After all the text was character cleaned and sentences were selected by relevant words, the next step was to process the text using large language models (LLM) to distribute the information in the database.

GPT-3.5 Turbo, developed by OpenAI, was chosen for text processing. However, I also considered other options such as Llama. Llama is a local model that runs on a computer using its computing resources. This approach has both pros and cons.

The pros include autonomy, as the model does not require a connection to cloud services, which is important for projects that require data to be stored locally. In addition, not having to pay for access to cloud services can be cost-effective in the long run.

However, a significant disadvantage of Llama[5] is that the word processing takes longer. Everything happens on only one computer, which limits performance, especially with large amounts of data. In contrast, GPT-3.5 Turbo works

through the cloud, where requests are sent to the server, processed, and responses are returned much faster. This allows a large number of articles to be processed in parallel.

Another important advantage of GPT-3.5 Turbo[6] is its flexibility. The cloud infrastructure makes it possible to quickly set up queries and receive responses without the need for lengthy training or model configuration. As a result, it saves not only time but also resources, which is especially important for a project with large amounts of data.

That's why I chose GPT-3.5 Turbo for this project, as it allows you to quickly process a large number of articles and get high-quality results in minimal time.

Once the LLM was selected, the next step was to obtain an API key from the OpenAI personal account. This key allowed making requests to the GPT-3.5 Turbo model through the API. Once the key was obtained, code was written to integrate the model into the project.

In the next step, it was important to choose a suitable database to structure and store the extracted information. MongoDB was chosen as the database because it uses the JSON format, which is ideal for handling semi-structured data such as titles, annotations, relevant sentences and other data. After successfully connecting to the database, the data writing process was set up in the code.

The next step was to properly compose the promt for the model. At this stage, several variations of promt were tested to understand how the model perceives queries and what information it returns in response. These tests helped determine the most effective promt format to achieve the desired results. Ultimately, the following promt was selected:

After that the whole list was saved to the main project folder in csv format for further LLM processing.

```
Please provide your response in valid JSON with the following fields:
{{
    "compound_name": "",
    "structure": "",
    "identifier": "",
    "evidence_description": "",
    "evidence_type": "",
    "disease_targeted": "",
    "mechanism_of_action": "",
    "references": "",
    "manual_validation": ""
}}
```

Figure 4

Once the promt was composed, it was specified that the model return a response in JSON format to easily write the information to the database. To assist the model in structuring the data, an example response ("Example Response") was provided to help guide the model in generating the output. This approach greatly simplified the integration of the analysis results with MongoDB and reduced the risk of data processing errors.

Also, if the same compound name occurs more than once, it is recorded in a separate section of the database. When

```
**Example Response:**
{{
    "compound_name": "Aspirin",
    "structure": "C9H8O4",
    "identifier": "CAS 50-78-2",
    "evidence_description": "Aspirin significantly improves neuronal survival by inhibiting COX enzymes.",
    "evidence_type": "in-vivo",
    "disease_targeted": "Neuronal injury",
    "mechanism_of_action": "COX inhibition",
    "references": "DOI:10.1234/abcd",
    "manual_validation": "Validated"
}}

**Important:** Only provide the JSON response as shown above. Do not include any additional text or explanations.
```

Figure 5

the same compound is mentioned again, new information is added to the existing record to avoid duplication and to ensure centralized storage of all data related to this element.

Once all articles have been processed and the components from them have been entered into the database, the LLM traverses all documents in MongoDB and calculates the confidence score for each compound name.

The logic for calculating confidence score will be presented below:

```python
if num_sources > 5 and len(unique_mechanisms) == 1:
    confidence_score = "High"
elif num_sources > 2 and len(unique_mechanisms) <= 2:
    confidence_score = "Medium"
else:
    confidence_score = "Low"
```

Figure 6

This stage allows an objective assessment of the quality of the extracted information based on the number of references and the variability of the provided informationd. This approach provides additional confidence in the relevance of the extracted data and allows researchers to quickly find the most reliable information.

```
compound_name : "Lactate"
compounds_info : Array (3)
  0: Object
  1: Object
    file_name : "11605911.xml"
    title : "Lactate metabolism and histone lactylation in the central nervous syst…"
    text_block : "pointed out that MCT activity was irrelevant with lactic acid producti…"
    structure : ""
    identifier : ""
    evidence_description : "Lactate has neuroprotective effects in ischemic stroke both in vivo an…"
    evidence_type : "in-vivo and in-vitro"
    disease_targeted : "Ischemic stroke"
    mechanism_of_action : "Downregulation of ATPADP ratio, reduction of ROS production in mitocho…"
    references : "DOI: Not provided"
```

Figure 7

6 Web interface

To easily search for items in the database, I created a web interface using the **Django** framework. Django was chosen for the following reasons:

- **Experience:** I have worked with Django before and have experience using it, which has made the development process much faster.

- **Easy to create web applications:** Writing a simple web application with Django is effortless thanks to built-in tools and accessible documentation.
- **Comfortable database integration:** Django allows you to quickly and conveniently connect your database, which is especially important when working with MongoDB.

6.1 Development steps

Creating the project and application:

I first created a new project in Django, and then in it a new app with a simple name of `my_app`.

Connecting the database:

A `mongo.py` file was created in the application directory, which is responsible for connecting to the MongoDB database. This made it convenient to retrieve and process data for display in the web interface.

Implementation of logic in `views.py`:

In the `views.py` file, three main functions were implemented:

- **First function:** retrieves all data from the database while hiding their unique identifiers (IDs).
- **Second function:** allows searching the database for information entered by the user in the search bar, such as the compound name.
- **Third function:** processes the user's request and sends the result to an HTML page.

Creating the HTML template:

An HTML file was created in the application directory. Using basic HTML and CSS, a page was designed with a search bar and a "Find" button. The user can enter the compound name into the search bar, press the button, and the system will send a query to the database and display the results.

Setting up routes:

All pages and functions were defined in the `urls.py` file to connect them to the main application. This ensured the web interface was displayed on the main page and that search queries worked correctly.

Launching and testing:

After the development was completed, the application was launched locally to test its functionality. The testing process included verifying that the search feature worked correctly, ensuring the data was displayed properly on the webpage, and checking the interaction between the application and the database. Additional checks were performed to make sure the application could handle user input without errors and return the correct results.

As a result, the application offers a simple and user-friendly interface for working with the database. It allows users to easily search for information about compounds by entering their names into the search bar, making the process efficient and straightforward

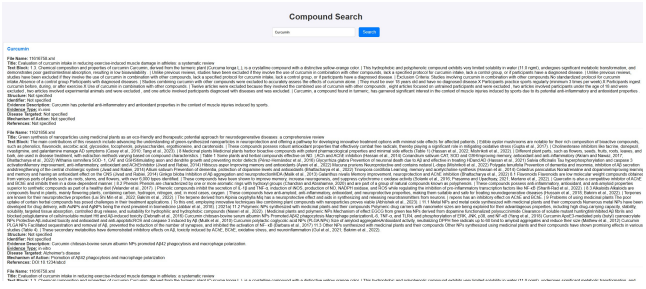

Figure 8

7 Testing and Validation

After the project was completed, it was important to check if the information displayed in the database was accurate and matched the real scientific articles. To do this, I manually selected three compounds from the database and compared the extracted data with the information provided in the original publications. This step was necessary to ensure that the data extraction and structuring processes were working correctly.

7.1 Check 1: Kynurenic Acid

The first article I reviewed was about **Kynurenic acid**. In the database, it is listed as being involved in both *in-vivo* and *in-vitro* experiments. The references provided for this compound in the database are *Mor et al., 2021; Zhang et al., 2019; Platten et al., 2019 [7].*

To verify this, I manually reviewed the references. The article *"Tryptophan metabolism as a common therapeutic target in cancer, neurodegeneration and beyond"* clearly states that Kynurenic acid has effects in both *in-vivo* (studies on living organisms) and *in-vitro* (laboratory experiments) environments. It also mentions that these effects were studied in models of Parkinson's disease (PD). Based on this information, I can confirm that the data in the database for Kynurenic acid is accurate.

7.2 Check 2: Mirogabalin

The second compound I reviewed was **Mirogabalin**. In the database, it is classified as being related to *in-vitro* experiments, and two studies are cited to support this classification.

- The first study mentions: *"Higher binding affinity to 2-1 and 2-2 subunits shown on Mirogabalin."* This suggests that the compound belongs to the *in-vitro* type.
- The second study states more clearly: *"In vitro evidence shows higher binding affinity to 2-1 and 2-2 [8]."* This confirms that Mirogabalin was indeed studied in *in-vitro* experiments.

Based on this, I can confidently say that the information in the database for Mirogabalin is accurate and supported by reliable references.

7.3 Check 3: Vitamin E

The third compound I reviewed was **Vitamin E**. The article related to this compound includes the following statement: *"Treatment of vitamin E significantly increased neuronal survival [9]."* While the study does not directly mention whether the experiments were *in-vivo* or *in-vitro*, the context strongly suggests that it was an *in-vivo* study. This conclusion is based on the fact that such experiments are commonly performed on living organisms when studying the effects on neuronal survival. Although the information is not explicitly stated, the database's classification seems reasonable and supported by indirect evidence.

7.4 Additional Testing

After reviewing these three compounds, I wanted to expand the testing to ensure the overall reliability of the database. I randomly selected 10 more compounds from the database for validation:

- For five of these compounds, the type (*in-vivo* or *in-vitro*) was clearly stated in the referenced studies.
- For four compounds, the type was not directly mentioned but could be confirmed through indirect evidence.
- For one compound, the referenced study was not closely related to the classification, making it difficult to draw clear conclusions.

7.5 Overall Conclusion

From this testing, it can be concluded that the information in the database is generally accurate and matches the referenced studies. However, because the sample size was small, it is difficult to make a definitive judgment about the entire database. Further validation with a larger number of entries would be needed to ensure the consistency and accuracy of the data. Still, the manual review has shown that the data extraction process works well in most cases, and the database provides relevant and reliable information for researchers.

8 Future Work

In the future, I plan to increase the size of the database to make it more comprehensive and cover more compounds and related studies. I also plan to add a toxicity section where the toxicity data for each compound will be immediately visible, including toxicity levels, affected organs, and acceptable safety limits. This will provide a more comprehensive view of the properties of each compound for researchers.

There are also improvements to be made to the accuracy of confidence score. In the current version, it checks articles with different data about the same compound and renders a verdict based on their number and lack of contradictions. In the future, it is planned to improve this mechanism by adding more factors such as source reliability, research methodology and publication date to make the evaluation more accurate and reliable.

Improving the web interface is also an important task. It is planned to make the interface more pleasant and user-friendly, add an autocomplete function in the search bar, implement data sorting and filtering, and visualize information through graphs and charts for easy analysis.

In addition, the LLM will be further optimized to improve the speed and accuracy of data processing. The integration of a high-quality NER (Named Entity Recognition) system is planned to better recognize key terms, compound names and relationships in the text. These improvements will make the database more reliable, user-friendly and efficient for researchers.

9 Conclusion

As a result of this Bachelor semester project (Bsp), a user-friendly database has been developed that allows the automatic extraction, processing and structuring of information from scientific articles on neuroprotective and neurotrophic compounds.

The database provides quick access to key information such as types of experiments, mechanisms of action and references to original research, making it a useful tool.

The use of large language models (LLMs), such as GPT-3.5 Turbo, in combination with MongoDB, made it possible to extract key components from scientific studies and organize them in a convenient and understandable way. The developed Django-based web interface provides a simple and intuitive tool for searching and analyzing the data, which greatly improves its accessibility and usability.

The validation performed confirmed that the system extracts and categorizes data correctly in most cases. Nevertheless, further refinements and more extensive validation are needed to maximize the accuracy and reliability of the information.

In the future the project will be supplemented with new functions, the volume of processed articles will increase, the quality of extracted information and web-interface will be improved. All identified minor deficiencies will be eliminated, and the system will become even more useful and efficient for working with scientific data.

References

[1] ChEMBL: a large-scale bioactivity database for drug discovery. https://academic.oup.com/nar/article/40/D1/D1100/2903401 Author: Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies.
[2] Wishart, D. S., et al. "DrugBank 5.0: A major update to the DrugBank database for 2018." Nucleic Acids Research, vol. 46, no. D1, 2018, pp. D1074-D1082. https://doi.org/10.1093/nar/gkx1037
[3] PubMed API https://www.ncbi.nlm.nih.gov/home/develop/api/
[4] PMC Open Access Subset https://pmc.ncbi.nlm.nih.gov/tools/openftlist/
[5] Llama https://www.llama.com/llama2/
[6] GPT-3.5 Turbo (OpenAI) https://openai.com/

[7] Serum neurotransmitter analysis of motor and non-motor symptoms in Parkinson's patients Yichun Fan et al. https://pmc.ncbi.nlm.nih.gov/articles/PMC11625801/

[8] Mirogabalin as a novel calcium channel 2 ligand for the treatment of neuropathic pain: Fei Yang, Yan Wang, Mingjie Zhang, Shengyuan Yu. https://www.frontiersin.org/journals/pharmacology/articles/10.3389/fphar.2024.1491570/full

[9] Association between composite dietary antioxidant index and cognitive function impairment Cong Zhao et al. https://www.frontiersin.org/journals/nutrition/articles/10.3389/fnut.2024.1471981/full