Machine Learning Support for Constructing Newt SBGN Maps

Personal Background

- Name: Umut Utku ERŞAHİNCE
- Location: Turkey, Ankara
- Email: utku.ersahince@ug.bilkent.edu.tr
- Mobile: +905383783495
- Linkedin
- Github

I am Umut Utku Erşahince, a second-year computer science major in Bilkent University. I became interested and involved in programming during the COVID lockdown era, which corresponds to high school for me. My high school didn't have a very good program for getting interested students into programming, so I wasn't introduced to it through school. Instead, I was introduced to it by a friend, who taught me programming. We developed some projects together, which piqued my interest. As a result, I decided to study computer science at Bilkent University. At university, I saw the work that was being done in the i-vis research lab and I was interested. I got in touch with Professor Uğur Doğrusöz to get more information about how I might participate. We discussed this project over email and also face to face. Hence, this proposal.

Education

- Bilkent University, Computer Engineering, Undergraduate (2nd year)
- CGPA: 3.96 / 4.00 (currently ranked 1st out of 175 in cohort)

Experience / Skills

Relevant Courses:

- Algorithms and Programming I & II
- Fundamental Structures of Computer Science I & II

- Introduction to Modern Biology

Projects

- pool-for-physicists-only
    - This was a project for the Algorithms and Programming II course where we implemented a physically accurate pool game.
    - I led the team during the development process. I also did research into the physics of the game and implemented the physics engine.
    - I continued developing the game after the course was over and added network features so we were able to play over the internet.
- ConwayGoL
    - This is an implementation of the famous Game of Life. I made it with one of my friends.
    - Besides the base functionality, we made serializers, converters between famous formats and cells with colors that mix.
    - Project was made in C++.
- Various Other Projects
    - I am currently working on two projects, CSML and SDL_Components. The former is an implementation of the client-server model in C. The latter is attempting to implement java swing style components using C++ and SDL2. I will use these two projects in an ongoing project implementing the game of go.
    - Loldle Guessr is a bot that plays the internet game loldle (classic mode). It uses concepts from information theory and it is almost equivalent to the best possible bot due to the simplicity of the game.
    - I also do a little bit of kaggle. I studied some ML theory myself and implemented classical neural networks (and a little more) for digit recognition.
    - I also started learning web development. I already have a project on it using cytospace.js, which is the graph visualization framework used by newt, and I am looking to improve my skills in this area.

Open Source

- I usually put projects I am working on my GitHub page. I also have 1-2 pull requests on small projects but none merged. The projects that these PR's are on are largely abandoned.

Exposure to Bioinformatics

- I took a biology course in my freshman year which I enjoyed. Our instructor mentioned bioinformatics. I also heard about it elsewhere. I am familiar with the usual biological terminology but I am not very familiar with bioinformatics. I would like to learn more.

What I Want to Learn This Summer

First of all, I would like to improve my skills in web development, which includes my skills in ReactJS, Type/Javascript… I would also like to get into open-source development. The primary skill that I want to improve is getting familiar with and writing code in a large codebase that I didn't write/am not familiar with beforehand. I see this project as a fit for me in those regards.

Project Proposal

If this project proposal goes through, I will be working on this idea. This project requires me to work on implementing additional features on Newt. Newt is a software to visualize and edit biological pathways in various formats including the standard notation of Systems Biology Graphical Notation (SBGN).

In order to get a better idea of what I wanted to do, I decided to clone the Newt GitHub repository into my locale. On the README.md included in the project, I was instructed to install the dependencies of the package by "npm install", which is a pretty standard thing to do. However, I ran into some issues pretty quickly. To illustrate, here is a picture of my terminal output:

```
npm ERR! code ERESOLVE
npm ERR! ERESOLVE unable to resolve dependency tree
npm ERR!
npm ERR! While resolving: undefined@undefined
npm ERR! Found: cytoscape@3.2.16
npm ERR! node_modules/cytoscape
npm ERR!   cytoscape@"github:iVis-at-Bilkent/cytoscape.js#master" from the root project
npm ERR!
npm ERR! Could not resolve dependency:
npm ERR! peer cytoscape@"^3.3.0" from cytoscape-edge-editing@2.0.1
npm ERR! node_modules/cytoscape-edge-editing
npm ERR!   cytoscape-edge-editing@"2.0.1" from the root project
npm ERR!
npm ERR! Fix the upstream dependency conflict, or retry
npm ERR! this command with --force or --legacy-peer-deps
npm ERR! to accept an incorrect (and potentially broken) dependency resolution.
```

This is a dependency collision. Basically, cytoscape-edge-editing version 2.0.1, required, as a dependency, cytoscape.js version 3.3.0 or higher. In my opinion, none of these are good signs. cytoscape.js version 3.3.0 was released in 2018, 6 years ago. The current release is 3.28.1. cytoscape-edge-editing version 2.0.1 was also released 4 years ago, the most current release being 4.0.0, released 2 years ago. Upon further investigation, I realized that the Newt project does not have, as its dependency, the original cytoscape.js project but rather this fork of it (from here on out, I will refer to this project as the fork). Upon reviewing the package.json file in the fork, I could see that this project was indeed a fork of cytoscape.js version 3.2.16.

From what I understand, there must be some sort of maintenance work done on this codebase to enable a smoother future development experience. Although I understand that this part is not directly relevant to the original issue I linked at the beginning, I would like to incorporate this into my work. My reasons are twofold. First of all, before beginning the intended project, I would like to set up optimal working conditions for myself. I feel like doing this sort of work would give me a great understanding of all the dependency projects. Second, as I already remarked, it would make future development more convenient. The following couple of paragraphs will include some ideas I have about this.

First of all, it is important to realize the full extent of the dependency projects affected by this conflict. Let me first of all mention a relatively easy conflict. sbgnviz v. 6.0.6 requires jquery v.^2.2.4, but the base project has a requirement of jquery v. 3.3.1. This should be an easy fix however, so I will not discuss it further. From what I could collect, following four dependencies appear to be affected by this conflict: cytoscape-edge-editing v. 2.0.1, cytoscape-expand-collapse v. 4.0.0, cytoscape-grid-guide v. 2.3.2, cytoscape-undo-redo v. 1.3.3. These are all relatively small projects that extend Cytoscape.js' base functionality in some way. They can be found under one GitHub organization. I have had some opportunities to try some of them out myself in one of my projects. All of these projects require Cytoscape.js v. 3.3.0 or higher. This means that the fork must be upgraded to at least v. 3.3.0. This is easier said than done, however. The fork of the Cytoscape.js project, unlike these other smaller projects, is a project of considerable size. Its default branch on GitHub, unstable, is currently 2760 commits ahead, 5051 commits behind the original project.

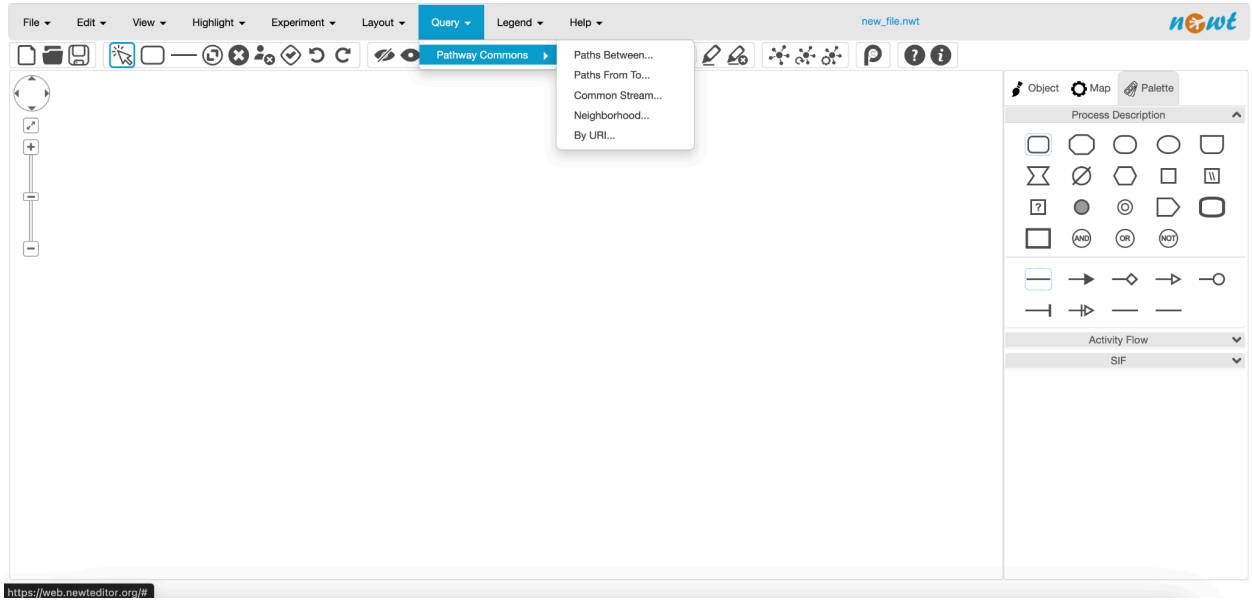
Upgrading this project in a safe way is not an easy task. To determine my strategy on how to approach this, I needed a way of visualizing the current branches that are active on the GitHub page of the project.

These are screenshots from the most recent commits into the forked cytoscape.js. Some particularly interesting branches for me are cyan and dark blue branches. These include changes that did exactly what I wanted, namely upgrading the Cytoscape.js version. I will certainly be looking into the content of these commits during the course of this project to see how it is done. As a small note, upgrade-cy has been merged into the unstable branch on the 9th of October update but it is not reflected here for some reason. I even found the PR on GitHub, which mentions the concerns I have. It also reflects the reason why Newt is not reliant on the unstable branch of the fork.

In my estimation, this maintenance project seems doable and it appears people who have worked on this project before had similar needs in the past. The following is a task breakdown of the things I plan to do in order to make this happen.

- **Task 1**: Perform Maintenance on Cytoscape.js fork to make dependencies work
    - **Subtask 1:** Look into the ways that this project extends the functionality provided by the default cytoscape.js to better understand how the newt project uses these additional functionalities. This will also help later with Task 2.
    - **Subtask 2:** Look into how the cyan and dark blue branches did the upgrades in the past to get a better understanding of what needs to happen in order for these changes to occur in a safe way. Looking into the changelog of cytoscape.js version 3.3.0 may also come in handy.
    - **Subtask 3:** Implement these changes to upgrade the unstable branch to a version >= 3.3.0 in order for the dependencies to work correctly. Also test whether it works with the newt project.

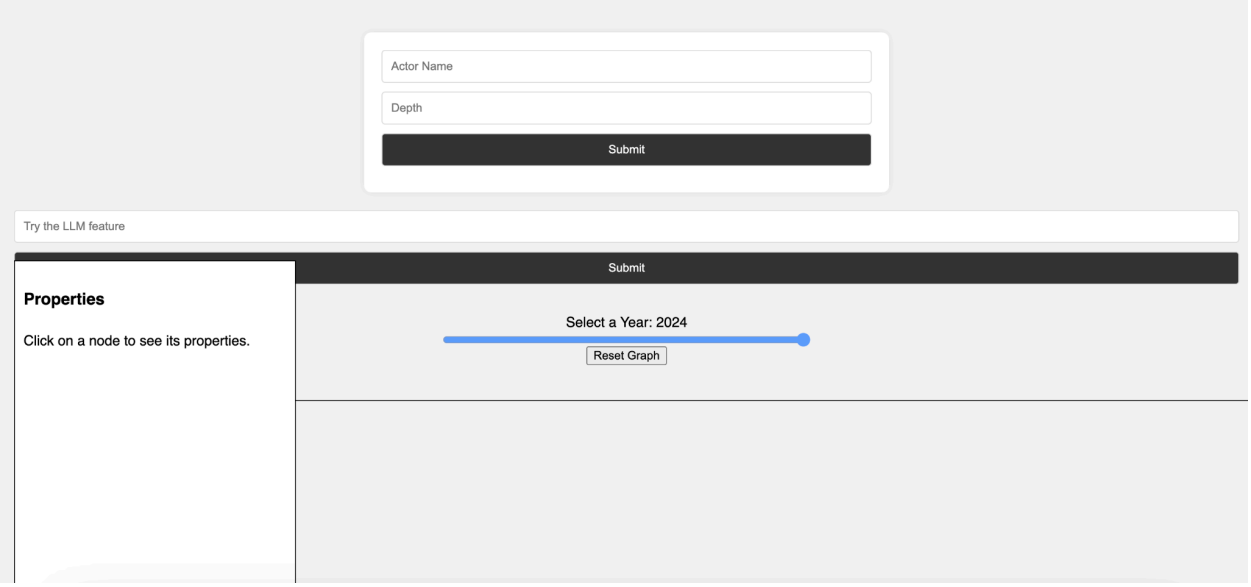
Now, after Task 1 is done, I am assuming I will have access to the project in my locale without any issues. I should be able to get a web page like this:



What this tab does is send a query to the [Pathway Commons] database. Each of these tabs will query the user for slightly different information, performing some checks on the information submitted, and sending an API request to the Pathway Commons website, with the format of SBGN. After the API successfully returns, the results are interpreted and added to the graph. The backend for this functionality is provided by a library called [Chise.js]. If the API request fails, we get a generic failed request screen.

One way for this project to go is to add another tab under this “Query->Pathway Commons” dropdown which is shown in the figure above. Let’s say we name this tab “AI Query”. Upon clicking, this tab will prompt the user for a string which explains the kind of pathway the user is looking to query. Then using an AI agent, like ChatGPT, we will aim to convert this string to a valid Pathway Commons API request. Then, using pretty much the same methods as those other query types already implemented in the code, we can get the updates shown to the user in the visual interface.

Let me explain what I am talking about here with a similar project that uses ChatGPT to convert explanatory strings into valid database requests. The project can be found here. This project uses the neo4j movie database to show visualizations about movies using cytoscape.js. Here is how the website looks upon opening:



As one can see, there is an LLM feature which is nothing but a ChatGPT API connection. In the code, we define for ChatGPT a **role prompt**, which defines what we want ChatGPT to do. This string is a very detailed description of what exactly we want ChatGPT to do, what we want the format of its output to be, what functionalities the database provides, what the things to pay attention to in the input, etc. See the picture below for a section of the role prompt in this example project.

```
Relationship Types:
ACTED_IN: A relationship from a Person node to a Movie node indicating that the person has acted in that particular movie.
DIRECTED: A relationship from a Person node to a Movie node indicating that the person has directed the movie.
PRODUCED: A relationship from a Person node to a Movie node indicating that the person has produced the movie.
WROTE: A relationship from a Person node to a Movie node indicating that the person wrote the movie.
REVIEWED: A relationship that could be from a Person node to a Movie node indicating that the person reviewed the movie.
FOLLOWS: A relationship that could be between two Person nodes indicating that one follows the other, perhaps on social media or within a professional context.

Property Keys:
For Person nodes:
born: The birthdate or birth year of the person.
name: The full name of the person.
For Movie nodes:
title: The title of the movie.
released: The release date or release year of the movie.
rating: The rating of the movie, which could be an IMDb rating, Rotten Tomatoes score, or similar.
tagline: A memorable phrase or sentence that encapsulates the essence of the movie.
summary: A brief description or synopsis of the movie.
For edges:
roles: Likely a property associated with the ACTED_IN relationship, detailing the character(s) played by the person in the movie.
Feedback from earlier runs will be given if the query is not correct. If the query is correct, the feedback will be empty.`;
```

This role prompt very clearly explains what the database that ChatGPT is writing a query for is and all of the properties the objects within can have. As for me, if we went with this approach, I would describe in detail how the Pathway Commons API calls work. Let's look at the Newt source code for some examples.

```
var queryURL = "http://www.pathwaycommons.org/pc2/graph?format=SBGN&kind=NEIGHBORHOOD&limit="
    + self.currentQueryParameters.lengthLimit;
var geneSymbolsArray = geneSymbols.replaceAll("\n", " ").replaceAll("\t", " ").split(" ");
```

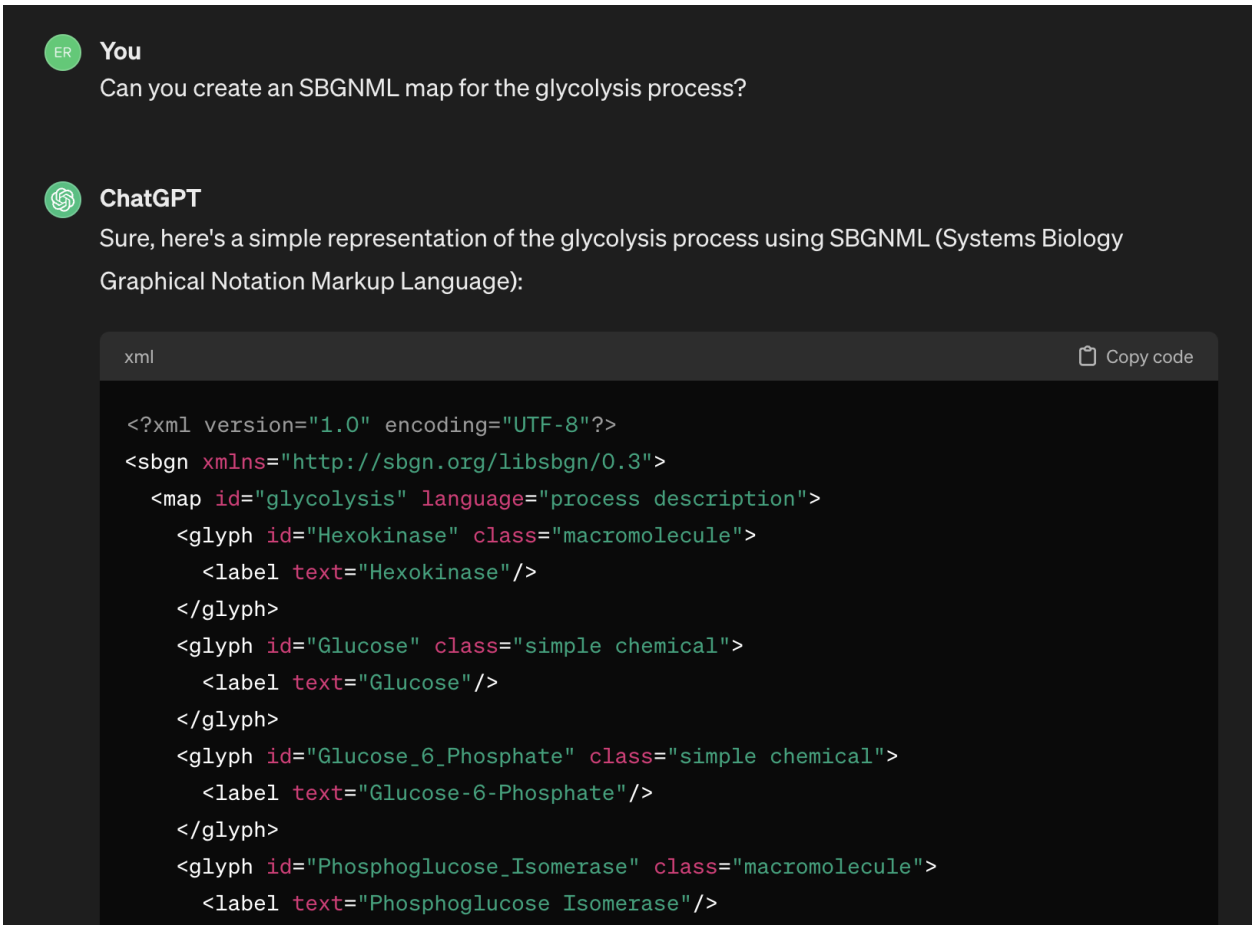
```
var queryURL = "http://www.pathwaycommons.org/pc2/graph?format=SBGN&kind=PATHSBETWEEN&limit="
    + self.currentQueryParameters.lengthLimit;
var geneSymbolsArray = geneSymbols.replaceAll("\n", " ").replaceAll("\t", " ").split(" ");
```

```
var queryURL = "http://www.pathwaycommons.org/pc2/graph?format=SBGN&kind=PATHSFROMTO&limit="
    + self.currentQueryParameters.lengthLimit;
var sourceSymbolsArray = sourceSymbols.replaceAll("\n", " ").replaceAll("\t", " ").split(" ");
var targetSymbolsArray = targetSymbols.replaceAll("\n", " ").replaceAll("\t", " ").split(" ");
```

These are some of the queries that Newt already performs. In all of these, we can see that the Newt backend specifies the format, the kind, and the limit. The information about what these fields (and potentially more) will be supplied in the role prompt I am going to generate. That way, we will maximize our chances of getting a valid and correct API request.

Let me also explain what must happen in the code in order to get this functionality. The backend for this functionality will be implemented at *backbone-views.js*. This file is already where the other options for the "Query" dropdown are implemented. I also need to add an html (and potentially css) template for this code to use and make sure to properly connect it to the various error screens other queries can get. There are potentially small changes required in the *app-menu.js* as well.

There is also one more approach to this project, which is adding a similar “AI Query” tab under the “Edit” dropdown menu. Here, the goal would not be generating an API request for the Pathway Commons database but rather creating SBGNML (SBGN Markup Language) directly. This is also possible and probably more useful. I tried playing around with ChatGPT a little bit to see what it could do. See below:



You

Can you create an SBGNML map for the glycolysis process?

ChatGPT

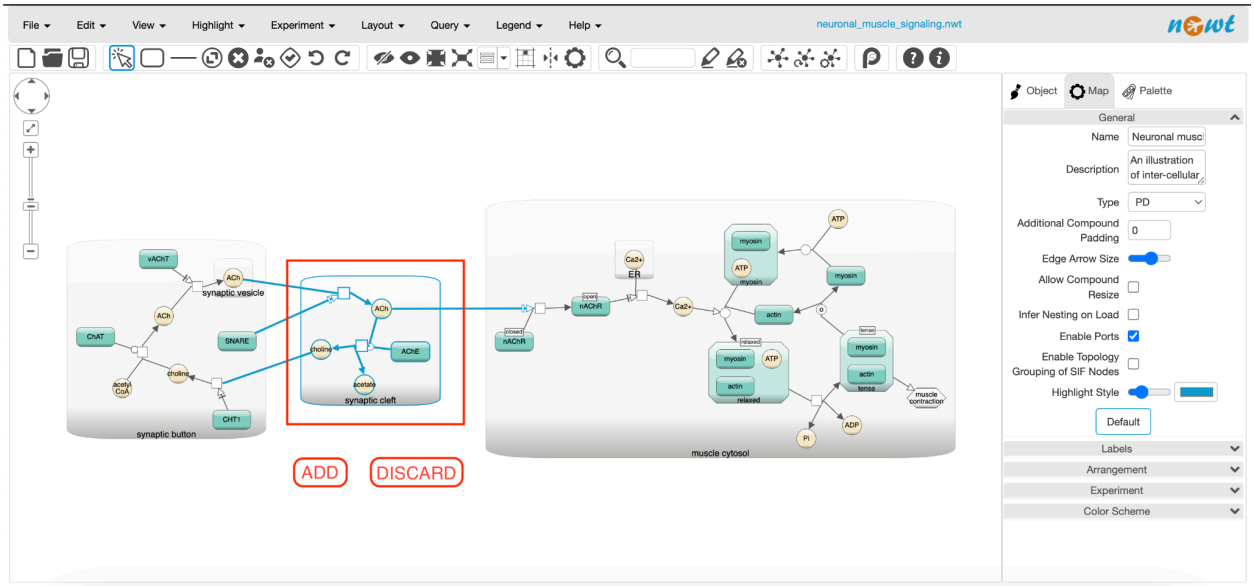
Sure, here's a simple representation of the glycolysis process using SBGNML (Systems Biology Graphical Notation Markup Language):

xml — Copy code

```
<?xml version="1.0" encoding="UTF-8"?>
<sbgn xmlns="http://sbgn.org/libsbgn/0.3">
  <map id="glycolysis" language="process description">
    <glyph id="Hexokinase" class="macromolecule">
      <label text="Hexokinase"/>
    </glyph>
    <glyph id="Glucose" class="simple chemical">
      <label text="Glucose"/>
    </glyph>
    <glyph id="Glucose_6_Phosphate" class="simple chemical">
      <label text="Glucose-6-Phosphate"/>
    </glyph>
    <glyph id="Phosphoglucose_Isomerase" class="macromolecule">
      <label text="Phosphoglucose Isomerase"/>
```

This is obviously not the full string that ChatGPT generated as that would require a lot of space to show. But, barring the fact that it could be inaccurate which I couldn’t assess myself as I am not a professional, it looked like a valid SBGNML map which gave me a lot of hope for this approach. Here, the approach is exactly the same as I described. We would define a role prompt, supply ChatGPT with the role prompt as well as the prompt from the user, let it return this map (and only this map with no text above it ideally, which would be specified in the role prompt), and process this map using the functionality provided by Chise.js (updateGraph() and convertSbgnmlToJson() being two crucial functions) to update the visual interface. The code for this would go exactly to the same files I have outlined in the previous pages.

Let me demonstrate what must happen after we have generated the map. See the picture below:



Let's assume that we initially don't have the section in the middle, circled in red (also highlighted with blue). In our AI generation tool, we give a prompt like "Give an SBGNML map for the processes that happen between the synaptic button and muscle cytosol in the synaptic cleft in the process of neuronal muscle signaling" and generate the map in the highlighted section. We will give the user options to add or discard this section of the map. If the user chooses to add this, we will go ahead and integrate this into the map. If the user chooses to discard this, we will delete it and it will be as if the user never generated this map.

I would like to bring out one potential issue here before I close this proposal. As is well known, ChatGPT API calls are not free. You can see the pricing here. This cost is billed to the account which provides the API key. What we might do is prompt the user for an API key in the dropdown menu where they write their prompt. This seems like an appropriate solution but ultimately the Newt project managers need to provide an answer here as to what to do.

- **Task 2**: Build the AI tool to generate SBGN maps
    - **Subtask 1:** Write and experiment with the role prompt. Test how successful the role prompt is. Look into ways of building an API connection.
    - **Subtask 2:** Implement the API connection in the website.
    - **Subtask 3:** Implement the add / discard functionality.

Schedule

- **May 1 - 26:** Begin reading the code (or at least the relevant parts) of the projects that I am planning to work on.
- **May 28 - June 3:** Work on Subtask 1.1
- **June 4 - 10:** Work on Subtask 1.2
- **June 11 - 17:** Work on Subtask 1.3
- **June 18 - July 8:** Finalize Task 1, potentially start Task 2
- **July 8 - 12:** Midterm review
- **July 13 - 20:** Work on Subtask 2.1
- **July 13 - 27:** Work on Subtask 2.2
- **July 28 - August 3:** Work on Subtask 2.3
- **August 3 - 19:** Finalize Task 2 and finalize the entire project.
- **August 19 - 26:** Final Evaluation Week

Availability

I have no commitments other than GSoC for the summer. That means no full/part time jobs or internships. I will be located in my hometown, Ankara, during the summer and I currently have no vacations scheduled. I will be able to allocate the equivalent of a work day for this project during the weekdays. If need be, I can use some time off the weekends as well. That comes up to about 40 hrs per week.