
PROGRAMMING CONVERSATIONS WITH CHATBOTS

BLAŽ ZUPAN, LJUBLJANA, 2026

LECTURE NOTES FOR THE UNIVERSITY OF LJUBLJANA'S COURSE ON
CHATGPT FOR RESEARCHERS



In this lecture, we move beyond individual prompts and focus on designing whole conversations with chatbots. By defining rules, constraints, templates, and interaction patterns that persist across multiple turns, we effectively “program” how the chatbot behaves over time. Rather than writing traditional code, we specify conversational logic in natural language: how questions are refined, how subproblems are generated and verified, how outputs are structured, and how results are incrementally transformed. This approach treats the dialogue itself as a programmable process, where careful prompt design allows us to control, guide, and compose chatbot behavior in a predictable and reusable way.

The notes below start with a few useful prompt techniques, but just a note: the main part of the lecture was on data gathering, analysis of complex texts, and structured presentations, including drawing of the graphs, from the information we have extracted.

QUESTION REFINEMENT PROMPT

We can engage language models to help us write better prompts. Here is an example initial prompt:

Whenever I ask a question, suggest a better question and ask me if I want to use it instead.

Or a more specific one:

Whenever I ask a question about molecular biology, suggest a better version emphasizing machine learning and AI. Ask me for the first question to refine.

Or a bit longer one where we also define the structure of LLM's response:

Whenever I ask a question about molecular biology, please suggest a better version of a question that involves some computational aspect and a specific related algorithm. Present this revised question and provide an answer (Q: <revised question>**, **A**: <answer>). Be brief; answer with a single sentence. Now, ask me for the first question to refine.**

Here is a brief example of the chat that followed our last prompt above:

Sure, what's your initial question about molecular biology that you'd like to refine into a computational aspect?

How do I compare two DNA sequences?

Q: How can I align two DNA sequences using the Needleman-Wunsch algorithm?

A: Use the Needleman-Wunsch algorithm to perform global sequence alignment, which scores matches, mismatches, and gaps to find the optimal alignment between two DNA sequences.

What's the next question you'd like to refine?

Can I infer an evolutionary tree given a set of gene sequences from various organisms?

Q: How can I construct an evolutionary tree using the Maximum Likelihood method based on a set of gene sequences from various organisms?

A: Utilize the Maximum Likelihood method to infer an evolutionary tree, which calculates the tree that maximizes the probability of observing the given gene sequences under a specified model of sequence evolution.

What other question would you like to refine next?

Here, we have successfully combined several prompting techniques we have already reviewed: programming, that is, instructing LLMs from the start on, question refinement, and constraining the output of the LLM to a specific, pred-defined structure.

COGNITIVE VERIFIER PATTERN

Here, we will ask LLM to help us break our question or problem into a set of subquestions, or subproblems. Our hypothesis is that once we provide an answer to these subquestions, LLM can become more precise and can provide us with a better answer. Here is an example of such a conversation, with an all-important initial prompt that specifies the type of chat we would like to have.

When you are asked a question, follow these rules: (1) generate several additional questions, one by one, that would help you to answer the questions more accurately; (2) after you get my answers, combine them with the individual questions to produce the final answer to the overall question. Number the questions. Ask max three subquestions.

Sure, I can follow those rules! Could you please ask your main question so we can begin the process?

Can I use machine learning to analyze single-cell gene expression data?

Great question! Let's break it down to get a comprehensive answer:

1. What are the characteristics of single-cell gene expression data that make it suitable or challenging for machine learning analysis?
2. What are the typical preprocessing steps required before applying machine learning to single-cell gene expression data?
3. What are some specific machine learning techniques commonly used for analyzing single-cell gene expression data?

Once I have your answers to these questions, I can give you a detailed and accurate response about using machine learning for analyzing single-cell gene expression data.

1) large number of features and data instances. 2) normalization and batch-effect correction. 3) dimensionality reduction, t-SNE visualization.

Great, thank you for the answers! Now, let's combine them with the original question:

Main question: Can I use machine learning to analyze single-cell gene expression data?

Answers to subquestions:

1. What are the characteristics of single-cell gene expression data that make it suitable or challenging for machine learning analysis?
 - Single-cell gene expression data typically have a large number of features and data instances, which can pose challenges for traditional machine learning algorithms due to scalability issues and high-dimensional data.
2. What are the typical preprocessing steps required before applying machine learning to single-cell gene expression data?
 - Typical preprocessing steps include normalization of gene expression values to account for differences in sequencing depth and gene expression levels, as well as batch-effect correction to remove technical variations introduced during sample preparation and sequencing.
3. What are some specific machine learning techniques commonly used for analyzing single-cell gene expression data?
 - Specific techniques include dimensionality reduction methods such as principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) for visualizing high-dimensional data and identifying clusters of cells with similar gene expression profiles.

Final answer: Yes, machine learning can be effectively used to analyze single-cell gene expression data. However, it requires addressing challenges such as handling a large number of features and instances, performing necessary preprocessing steps like normalization and batch-effect correction, and employing specific techniques such as

dimensionality reduction and t-SNE visualization to extract meaningful insights from the data.

Can you modify the initial prompt for the ChatGPT to be a bit less verbose? Can you modify it so that besides the question, we can also get an explanation of why answering this question will provide additional insight and help in composing the final answer? Can you compose a prompt to request that questions and their motivations be printed out in a structured way?

SEVERAL USEFUL PROMPT PATTERNS

By “prompt patterns” we here refer to the partial structure of our prompt to additionally specify how the output should look. Here are some functional patterns that can constrain the type of the output:

always answer with yes/no

always limit the answers to one sentence

always structure the answer to include a title and a one-paragraph summary

always answer in Slovene language

always answer with mathematical equations only, no additional explanation

Patterns give us control over the output. Test the last prompt above, and, say, ask something technical, like

How do I determine the reaction speed of my chemical reaction?

What is principal component analysis?

Patterns in prompts allow us to instruct LLM to use the output of a specific type. You can, of course, invent any such patterns, but it is good to know that “always limit,” “always answer,” and “always structure the answer” are good phrases to start your prompt that constrain all future answers by ChatGPT in the current chat. Besides those above, there are, of course, other very interesting patterns, the persona pattern we will introduce next being one of them.

ITERATIVE DATA GATHERING

While the particular utility of the chatbots we describe here is also referred to as “chatbot programming”, by “programming” we do not mean conventional computer programming, but rather the incremental instruction of a large language model through natural-language prompts that constrain, transform, and refine its output. Consider the following sequence of prompts, issued one at a time (chatbot’s responses are not shown).

Whenever you generate output, turn it into a table with columns of property names and rows with examples. Answer with only a few words or preferably just with numbers.

What kind of city is Ljubljana, in terms of size, population, and number of theatres?

Does it have a subway?

Add Kranj and Koper.

Place cities in rows, and properties in columns (transpose current table).

Compare Maribor, Zagreb and Vienna.

Include all EU capitals.

This sequence demonstrates progressive refinement (iterative prompting) combined with a persistent output-format constraint and table-transformation patterns, allowing the user to gradually expand, compare, and restructure the model's output without having to restate the full task.

Here's one of the intermediate answers:

City	Size (km ²)	Population	Number of Theatres	Subway
Maribor	147.5	112,065	5	No
Zagreb	641.4	806,341	20	Yes
Vienna	414.6	1,931,830	50	Yes

Be careful; answers are not necessarily correct: Zagreb has a tram, not a subway. But otherwise, it looks rather impressive, right? Using GPT-4o, when writing this text the latest LLM available through ChatGPT, you can also export the resulting data into an Excel file.

Export the data to Excel file.

About tables: LLMs generate text. The data table, as shown above, is also a sequence of characters and strings. We can copy the table (by clicking on a small copy icon in ChatGPT's interface). Pasting the copied text to some text editor (your instructor uses Sublime, for instance) reveals the text generated. Or we can simply ask ChatGPT what format it uses for the table:

What is the format that you have used to generate the table? Is this markdown?

Yes, the format I used to generate the table is Markdown. Markdown is a lightweight markup language with plain text formatting syntax, often used for formatting readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

Here is the Markdown code for the table:

markdown

Copy code

City	Size (km ²)	Population	Number of Theatres	Subway
Maribor	147.5	112,065	5	No
Zagreb	641.4	806,341	20	Yes
Vienna	414.6	1,931,830	50	Yes

You can use this format to generate tables in any Markdown-supported environment.

You can learn more about Markdown and its simple syntax by asking ChatGPT about it. Here, it suffices to say that even when ChatGPT displays tables, this is rendering of a text in the Markdown syntax, and everything is still in the realm of LLM generating a string of characters. Similarly, for example, ChatGPT renders text in bold (Markdown syntax uses two asterisks and with them embodies the part of the text that should be rendered in **bold**) or can display itemized lists or even nicely set mathematical equations (using LaTeX syntax). Not that we will discuss much about this here, but to know, again: LLMs generate text that is nicely rendered by ChatGPT.

Could you please show me an example of a complicated equation without any extra explanation?

Sure, here is an example of a complicated equation:

$$\sum_{k=1}^{\infty} \left(\int_0^1 \frac{\sin(kx)}{x} dx \right)^2 = \frac{\pi^2}{6} (\ln(2\pi) - \gamma)^2 + \frac{\pi^4}{72}$$

Hm, how was this rendered?

This equation was rendered using LaTeX, a typesetting system that's widely used for mathematical and scientific documents. LaTeX allows for the precise formatting of mathematical notation. Here's how the LaTeX code for the equation looks:

latex

Copy code

```
\[
\sum_{k=1}^{\infty} \left( \int_0^1 \frac{\sin(kx)}{x} dx \right)^2 = \frac{\pi^2}{6} (\ln(2\pi) - \gamma)^2 + \frac{\pi^4}{72}
\]
```

TEMPLATE PATTERN

Prompt templates are predefined structures or patterns to guide the LLMs text generation. These templates can help in generating more consistent and targeted outputs. In practice, they can be used either in a single step or staged across multiple prompts, where the template and formatting constraints are defined first, and the content to be generated is provided later,

enabling tighter control over both structure and process. Here is an example of a staged prompt:

I am going to give you a template for your output. CAPITALIZED WORDS are my placeholders. Fill in my placeholders with your output. Preserve the overall formatting of my template. My template is: **Question: QUESTION **Answer:** ANSWER Do not do anything at this stage. I will give you the data to format in the next prompt. Provided the data, you will create five questions using my template. Ok?**

Sure, I understand! Just let me know when you're ready to proceed with the data.

The data: Julian's Assange's wife has told of her elation that the WikiLeaks founder has been released from Belmarsh prison in London and will soon be a "free man" under a deal in which he will plead guilty to violating US espionage law. Speaking from Australia, where she flew on Sunday to prepare her family's new life, Stella Assange, a human rights lawyer, said she had not told the couple's two young sons, Gabriel and Max, about their father's release after five years in jail for fear of the information leaking. She said: "All I told them was that there was a big surprise. And, on the morning that we left, I told them where we're heading to the airport, and we got on the plane, and I told them that we were going to visit our family, their cousin, their grandfather and so on. "And they still don't know. We've been very careful, because obviously, no one can stop a five- and a seven-year-old from, you know, shouting it from the rooftops at any given moment. Because of the sensitivity around the judge having to sign off the deal, we've been very careful, just gradually, incrementally telling them information."

Assange, 52, was woken at 2am in his cell in Belmarsh on Monday before being handcuffed and transported to Stansted airport to take a chartered jet via Bangkok to the island of Saipan in the Northern Mariana Islands, which is US territory.

Now please create three questions using my template.

Question: Where did Julian Assange travel after his release from Belmarsh prison?

Answer: Julian Assange was transported to Stansted airport in London and then took a chartered jet via Bangkok to the island of Saipan in the Northern Mariana Islands, which is US territory.

Question: Why did Stella Assange refrain from informing her sons about Julian Assange's release immediately?

Answer: Stella Assange refrained from immediately informing her sons about Julian Assange's release due to the sensitivity around the judge signing off on the deal. She was concerned that their young age might lead them to inadvertently disclose the information.

Question: How long had Julian Assange been in jail before his release?

Answer: Julian Assange had been in jail for five years before his release from Belmarsh prison in London.

The next example demonstrates a prompt template used in a single step, a so-called single-shot use of a prompt template. Here, the task description and the output template are provided together, allowing the language model to generate structured content immediately that matches the predefined formatting and constraints.

Create a random strength workout with complementary exercises. I am going to provide a template for your output, where CAPITALIZED-WORDS are placeholders, replaces them with content. This is the template: **##: NAME <newline> **set and timing:** **REPS @ SETS, ESTIMATED-TIME-IN-MINUTES** <newline> **muscles:** **MUSCLE GROUPS WORKED** <newline> **difficulty:** **DIFFICULTY SCALE 1-5** <newline> **notes:** **FORM NOTES. No itemization.****

Notice that we have used several tricks. We embedded the text that should be set in bold with ******; we already know this stems from the markdown syntax. Placeholders are in capital letters. We have used the “<newline>” tag to tell LLM where we want line breaks. To avoid the use of bullets, we have added “no itemization” at the end of the prompt. The output:

1: Push-ups

set and timing: 15 @ 3, 10 minutes

muscles: Chest, Shoulders, Triceps

difficulty: 3

notes: Maintain a straight body alignment throughout.

2: Dumbbell Shoulder Press

set and timing: 12 @ 3, 12 minutes

muscles: Shoulders, Triceps

difficulty: 4

notes: Keep core engaged to avoid arching the back.

3: Bent-over Dumbbell Rows

set and timing: 12 @ 3, 10 minutes

muscles: Upper Back, Biceps

difficulty: 3

notes: Focus on pulling elbows back and squeezing shoulder blades together.

4: Bodyweight Squats

set and timing: 20 @ 3, 8 minutes

muscles: Quadriceps, Glutes, Hamstrings

difficulty: 2

notes: Keep knees tracking over toes and maintain a neutral spine.

5: Plank

set and timing: 30 seconds @ 3, 6 minutes

muscles: Core, Shoulders, Back

difficulty: 2

notes: Brace your core and avoid sagging hips.

6: Dumbbell Bicep Curls

set and timing: 12 @ 3, 8 minutes

muscles: Biceps, Forearms

difficulty: 2

notes: Control the movement to fully contract the biceps.

7: Glute Bridges

set and timing: 15 @ 3, 8 minutes

muscles: Glutes, Hamstrings, Lower Back

difficulty: 2

notes: Squeeze your glutes at the top of the movement.

Cool-down: Finish with stretching exercises for each major muscle group, holding each stretch for 20-30 seconds.

This workout targets multiple muscle groups and provides a balanced strength training session. Adjust weights and repetitions based on your fitness level and goals.

This is already great, right? Consider an AI that, together with text like the one above, would provide a sketch of the exercise or an instructional video. All this is probably inevitable; the AI technology of today is still in its infancy. Until then, you can further refine the workout plan using some more prompting and providing constraints to the list:

Group the exercises to the warmup, main part, and cooldown.

Construct the exercise plan to fit within half-an-hour.

Emphasis should be on legs and feet.

Make exercises lighter.

Or anything similar. Try it out!

TEMPLATE-DRIVEN KNOWLEDGE EXTRACTION

Let us combine the template from the previous section with further analysis of the data. Here we combine an output schema template, an information-extraction template, and formatting constraints to control both what is extracted and how it is presented. Suppose we are interested in people involved in the assassination of Archduke Franz Ferdinand in Sarajevo, but would like to instruct an LLM to provide the output in a specific form:

I am providing a template for your output. CAPITALIZED words are placeholders.

Preserve the overall formatting. My template:

Bio: <NAME>

****Summary:** <ONE SENTENCE SUMMARY>**

****Born:** <YEAR OF BIRTH>, <PLACE>**

****Major impact:** <ONE SHORT PARAGRAPH OF MAJOR ROLE OF THIS PERSON>**

Extract the names of all the people mentioned in the following text and report on them using the template.

<<<In the summer of 1914, the sovereigns of Europe were woven together by treaties, alliances, as well as secret agreements. The Triple Alliance (1882) encompassed the German Empire, Austria, and Italy.[34]

On 28 June 1914, Archduke Franz Ferdinand of Austria, heir presumptive to Emperor Franz Joseph I of Austria, visited Sarajevo, the capital of the recently annexed Bosnia and Herzegovina. Cvjetko Popović, Gavrilo Princip, Nedeljko Čabrinović, Trifko Grabež, and Vaso Čubrilović (Bosnian Serbs) and Muhamed Mehmedbašić (from the Bosniaks community),[35] from the movement known as Young Bosnia, took up positions along the route taken by the Archduke's motorcade, to assassinate him. Supplied with arms by extremists within the Serbian Black Hand intelligence organisation, they hoped his death would free Bosnia from Austrian rule, although there was little agreement on what would replace it.[36]

Nedeljko Čabrinović threw a grenade at the Archduke's car and injured two of his aides, who were taken to hospital while the convoy carried on. The other assassins were also unsuccessful but, an hour later, as Ferdinand was returning from visiting the injured officers, his car took a wrong turn into a street where Gavrilo Princip was standing. He fired two pistol shots, fatally wounding Ferdinand and his wife Sophie.[37] Although Emperor Franz Joseph was shocked by the incident, political and personal differences meant the two men were not close; allegedly, his first reported comment was "A higher power has re-established the order which I, alas, could not preserve".[38]>>>

Bio: Archduke Franz Ferdinand of Austria

Summary: Archduke Franz Ferdinand of Austria, heir presumptive to Emperor Franz Joseph I, was assassinated in 1914, an event that triggered World War I.

Born: 1863, Graz, Austria

Major impact: Archduke Franz Ferdinand's assassination in Sarajevo by Gavrilo Princip of the Young Bosnia movement on June 28, 1914, sparked a series of events leading to the outbreak of World War I. His death destabilized Europe, setting off a chain reaction of alliances and conflicts that culminated in a global war.

Bio: Gavrilo Princip

Summary: Gavrilo Princip was a Bosnian Serb nationalist who assassinated Archduke Franz Ferdinand of Austria, sparking World War I.

Born: 1894, Obljaj, Bosnia and Herzegovina

Major impact: Gavrilo Princip's assassination of Archduke Franz Ferdinand of Austria on June 28, 1914, was a catalyst for World War I. As a member of the Young Bosnia movement,

Princip's actions were driven by a desire for Bosnian independence from Austrian rule, although his act led to widespread and catastrophic consequences.

Bio: Nedeljko Čabrinović

Summary: Nedeljko Čabrinović was one of the assassins involved in the plot to kill Archduke Franz Ferdinand of Austria.

Born: 1895, Sarajevo, Bosnia and Herzegovina

Major impact: Nedeljko Čabrinović attempted to assassinate Archduke Franz Ferdinand of Austria on June 28, 1914, by throwing a grenade at his car. Although unsuccessful in killing the Archduke, Čabrinović's actions contributed to the tensions that led to the outbreak of World War I.

...

The list includes few more people. Amazing, right? In prompt engineering, templates play a crucial role in shaping the way queries or prompts are constructed to elicit specific responses from language models like GPT-3. Templates essentially serve as structured formats or frameworks into which variables or placeholders can be inserted. They can improve standardization and consistency, reproducibility, assist in targeted query generation. Templates can be adapted and scaled across different applications and domains. Once a template structure is defined, it can be reused with minor adjustments for similar tasks or contexts, which saves time and effort in prompt design.

DRAWING INTERACTION GRAPHS: SARAJEVO ASSASSINATION

This part is a bit tricky. Graph drawing is currently supported in both Copilot Chat and ChatGPT, but unless you provide specific and detailed instructions, the resulting graphs can be (extremely) messy and unhelpful. Of course we expected this to be improved in the coming months. But before that, and if you're a computer scientist—or simply someone who enjoys creating or exploring graphs—you can write a well-structured prompt with clear instructions. This prompt, if put together right, can then be translated into a Python script that produces clean and meaningful visualizations. (I did say this section was tricky!)

The example below was actually inspired by a question from the audience during our last lecture. We were discussing the people involved in the Sarajevo Assassination, and at the end of the lecture there was a question whether it was possible to prompt for the interactions between those individuals. Not only is it possible—we can also generate a neat sociogram. Here's how:

Please read:
https://en.wikipedia.org/wiki/Assassination_of_Archduke_Franz_Ferdinand. Questions will follow. Ok?

Got it. Ready when you are.

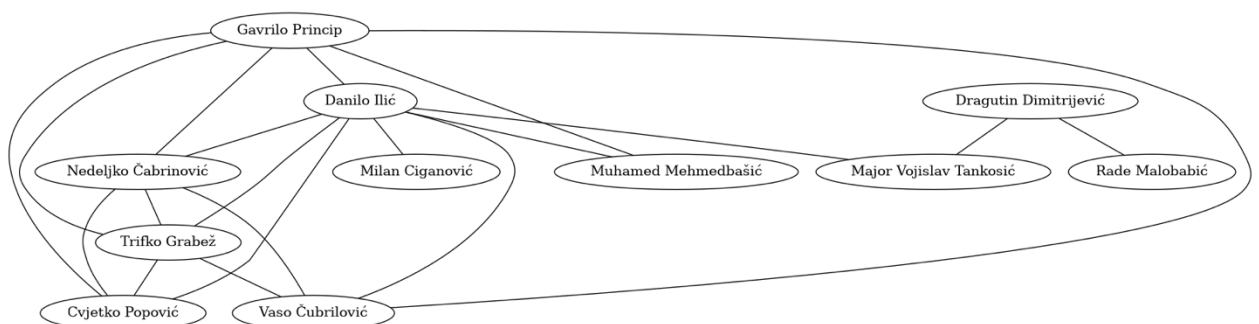
Find pairs of people who were in contact with each other prior to the assassination. Use the form: person-person. Search for at least 20 such interactions.

Here are at least 20 pairs of people who were in contact with each other prior to the assassination of Archduke Franz Ferdinand, based on the Wikipedia article 1:

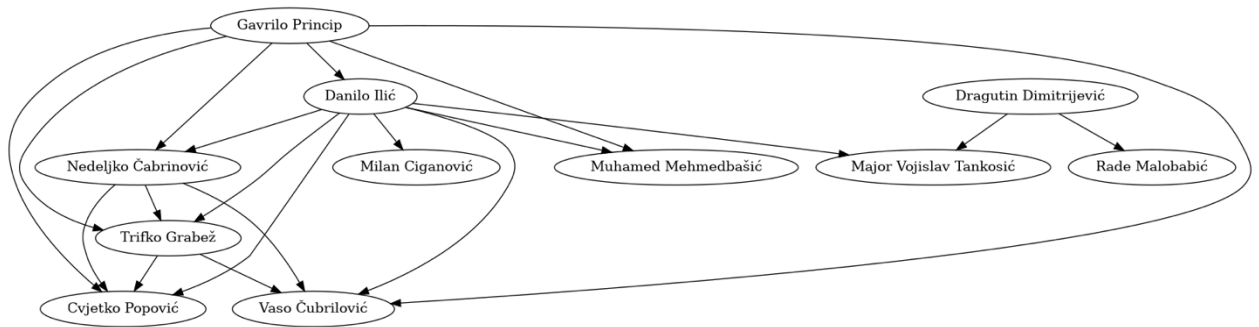
Gavrilo Princip – Danilo Ilić
 Danilo Ilić – Nedeljko Čabrinović
 Danilo Ilić – Trifko Grabež
 Danilo Ilić – Vaso Čubrilović
 Danilo Ilić – Cvjetko Popović
 Danilo Ilić – Muhamed Mehmedbašić
 Danilo Ilić – Milan Ciganović
 Danilo Ilić – Major Vojislav Tankosić
 Gavrilo Princip – Trifko Grabež
 Gavrilo Princip – Nedeljko Čabrinović
 Gavrilo Princip – Vaso Čubrilović
 Gavrilo Princip – Cvjetko Popović
 Gavrilo Princip – Muhamed Mehmedbašić
 Nedeljko Čabrinović – Trifko Grabež
 Nedeljko Čabrinović – Vaso Čubrilović
 Nedeljko Čabrinović – Cvjetko Popović
 Trifko Grabež – Vaso Čubrilović
 Trifko Grabež – Cvjetko Popović
 Dragutin Dimitrijević – Major Vojislav Tankosić
 Dragutin Dimitrijević – Rade Malobabić

Construct an interaction graph where people are represented as nodes, and the edges represent their interactions. Render this graph with a Python script, use Graphviz, save the result in an image file, and show it here.

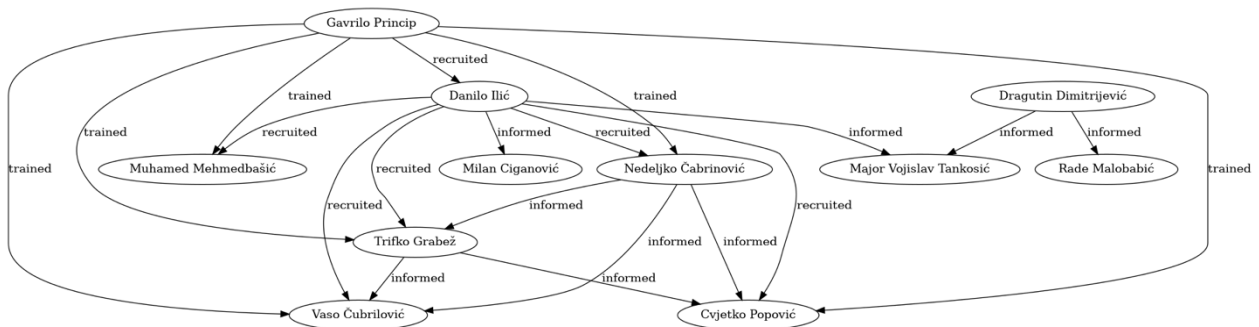
Here is the interaction graph showing connections between individuals involved in the assassination of Archduke Franz Ferdinand:



Edges should be directed. A -> B, where A is the one that initiated the contact.

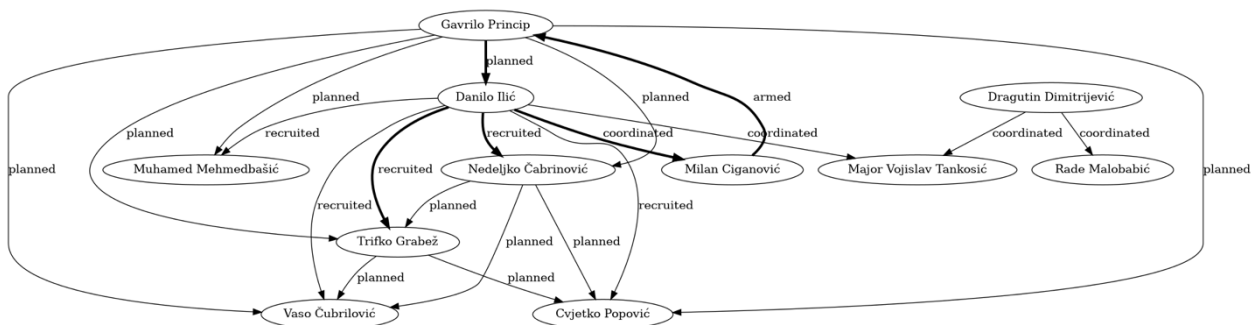


Describe every interaction with one word, and use this as an edge label. Revise the graph accordingly.



Add edges that show the arming of Gavriilo Princip. Denote most important interactions with thick lines.

Here's the enhanced interaction graph with thicker lines highlighting the most important interactions, such as recruitment and arming of Gavriilo Princip:



Disclaimer: the author of this text does not claim anything is right in this graph, though some interactions at least make sense, and others would need to be verified and scrutinized by somebody with knowledge of history. The only purpose of this section was to show that such graphs can be created, and because of the Graphviz package (created in 1991!) they can be very neat.

DRAWING INTERACTION GRAPHS: GENE NETWORKS

Here is one more graphing example, perhaps better fitting to biomedical audience, but using almost exactly the same tricks as above.

Name most important genes in Dictyostelium development.

Some of the most important genes involved in Dictyostelium discoideum development include:

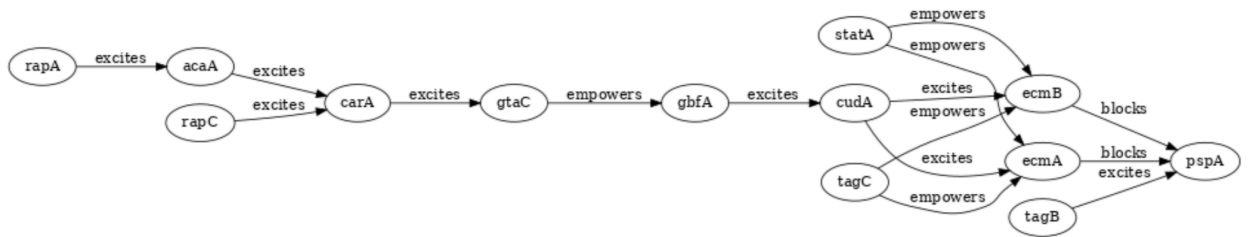
acaA – Encodes adenyl cyclase A, crucial for cAMP signaling during aggregation.
carA – cAMP receptor A, essential for detecting cAMP and initiating aggregation.
gtaC – A transcription factor regulating gene expression during early development.
gbfA – G-box binding factor, another transcription factor controlling developmental genes.
cudA – Required for culmination; expressed in the tip organizer.
statA – A STAT-like transcription factor regulating prestalk cell differentiation.
ecmA / ecmB – Marker genes for prestalk cell subtypes.
tagB / tagC – Involved in cell-type-specific gene regulation during differentiation.
pspA – Prespore-specific gene, a marker of spore cell fate.
rapA / rapC – Regulate aggregation and differentiation via Ras signaling.

These genes play roles in signaling, transcriptional regulation, and cell fate determination throughout the stages from aggregation to fruiting body formation.

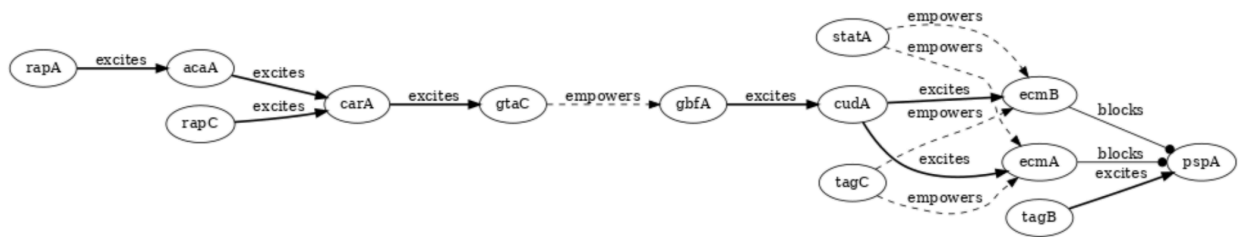
List (epistatic) interactions between these genes. Like: gene -> gene: name of the interaction where "->" means influences (the expression), and where "name of the interaction" is a short name of what the interaction is (say, inhibits, excites, blocks, empowers, use one word only.

acaA -> carA: excites
carA -> gtaC: excites
gtaC -> gbfA: empowers
gbfA -> cudA: excites
cudA -> ecmA: excites
cudA -> ecmB: excites
statA -> ecmA: empowers
statA -> ecmB: empowers
ecmA -> pspA: blocks
ecmB -> pspA: blocks
tagB -> pspA: excites
tagC -> ecmA: empowers
tagC -> ecmB: empowers
rapA -> acaA: excites
rapC -> carA: excites

Now use this list, and construct a graph with genes as nodes and relations as directed edges (instead of arrow, inhibits is a full circle). Annotate the relations. Create the graph using Python and Graphviz, and store the image; this is the result. Render the image (inline).



Use different types of arrows for different relations. Blocks could have a filled circle at the end of the edge instead of the arrow. Excites could be strong arrows and empowers a dashed one.



Depending on the verbosity of the chatbots there is usually some more text associated with such chat, and we have left it out. Copilot Chat would also offer to draw its own graph, and while it is perfectly ok to try, they still need to work on this.