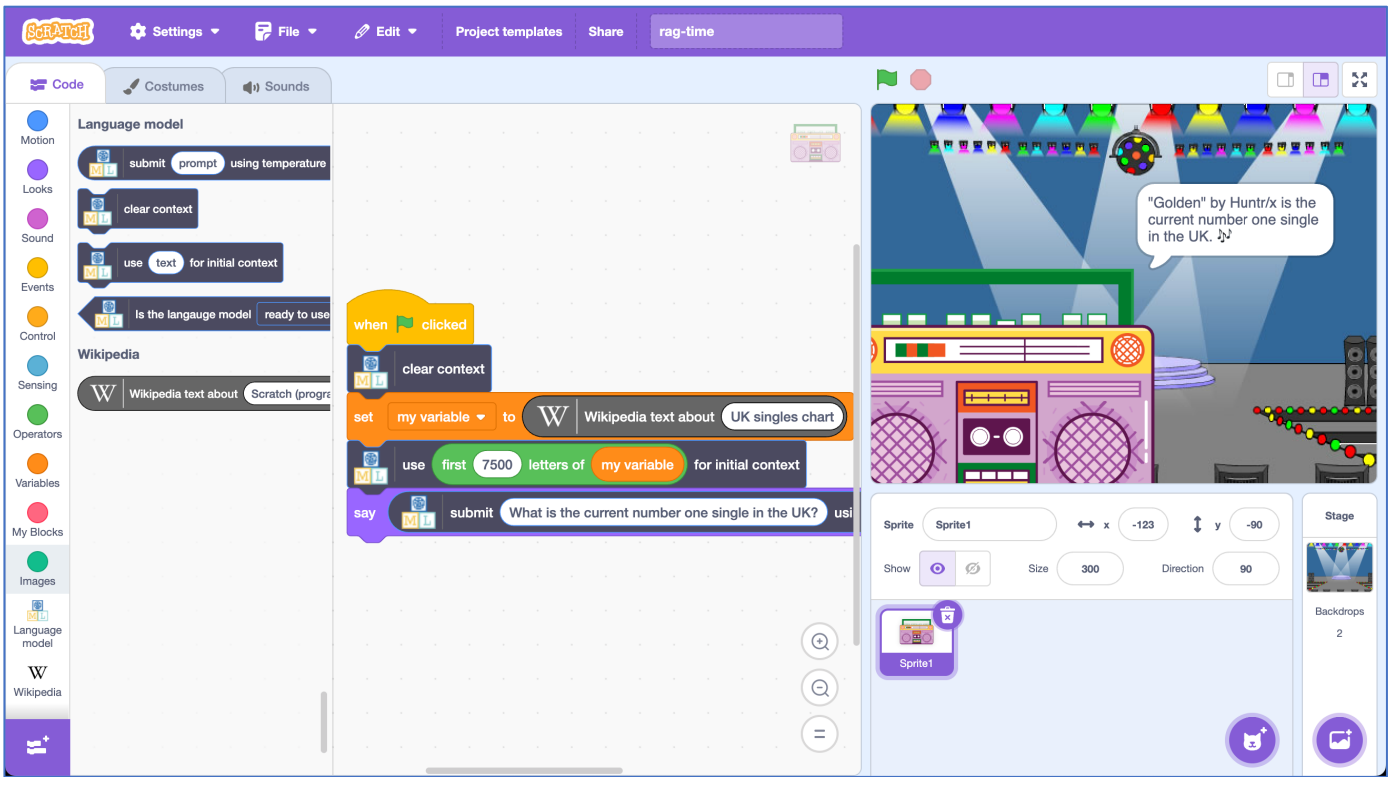
RAG-time

In this project you will use a language model to create a project that will display up-to-date information from the Internet, whenever it runs.

You will use a technique called “RAG” using the same sort of artificial intelligence technology that powers tools like ChatGPT.

This project assumes you already understand the basics of how language models work.

If you haven’t already completed the “Language Models” worksheet, you will understand this project better if you do that worksheet first.



If you are under the age of 13, please only use a small language model with supervision from a trusted adult.

Generative AI can sometimes generate text that isn't nice or appropriate.

1. Go to https://machinelearningforkids.co.uk/

2. Click on "**Get started**"

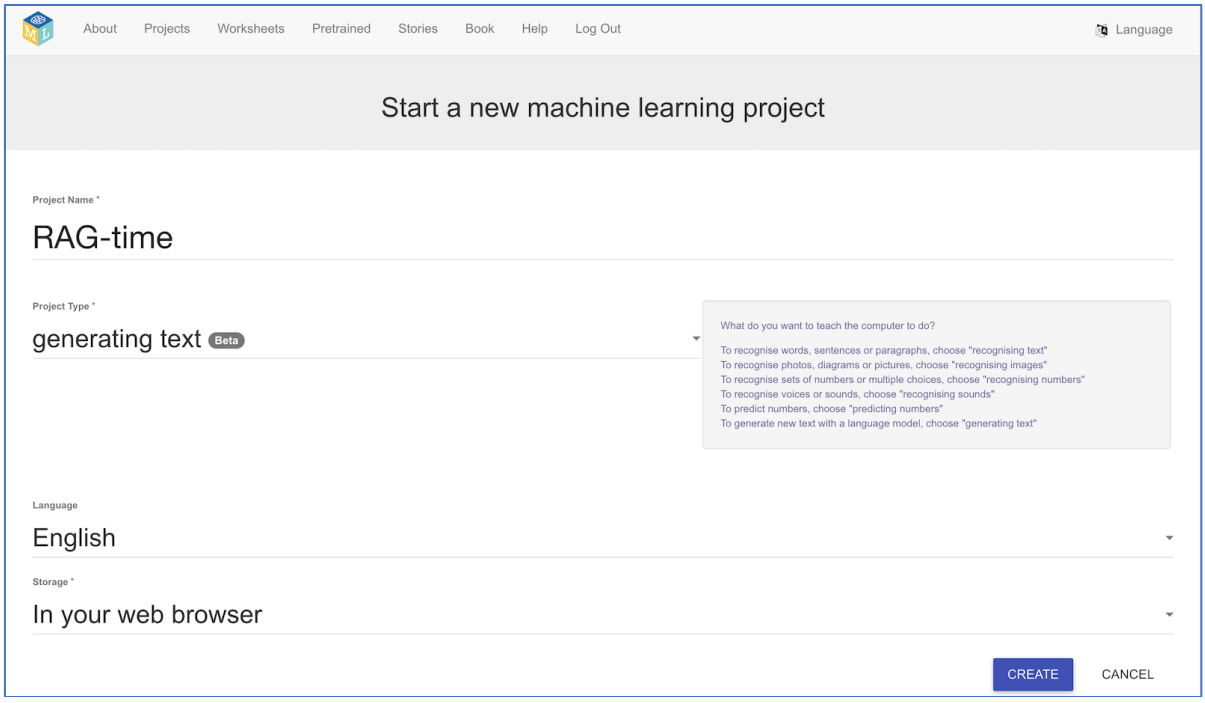
3. Click on "**Log In**" and type in your username and password

If you can't remember your username or password, ask your teacher or group leader to reset it for you.

4. Click on "**Projects**" on the top menu bar

5. Click the **"+ Add a new project**" button.

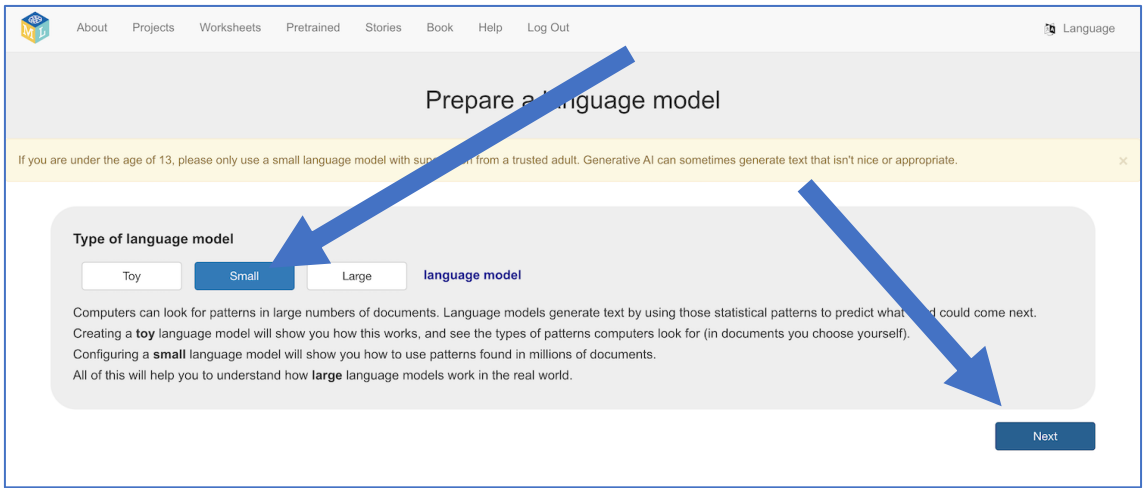
6. Name your project "**RAG-time**" set it to learn how to generate text.



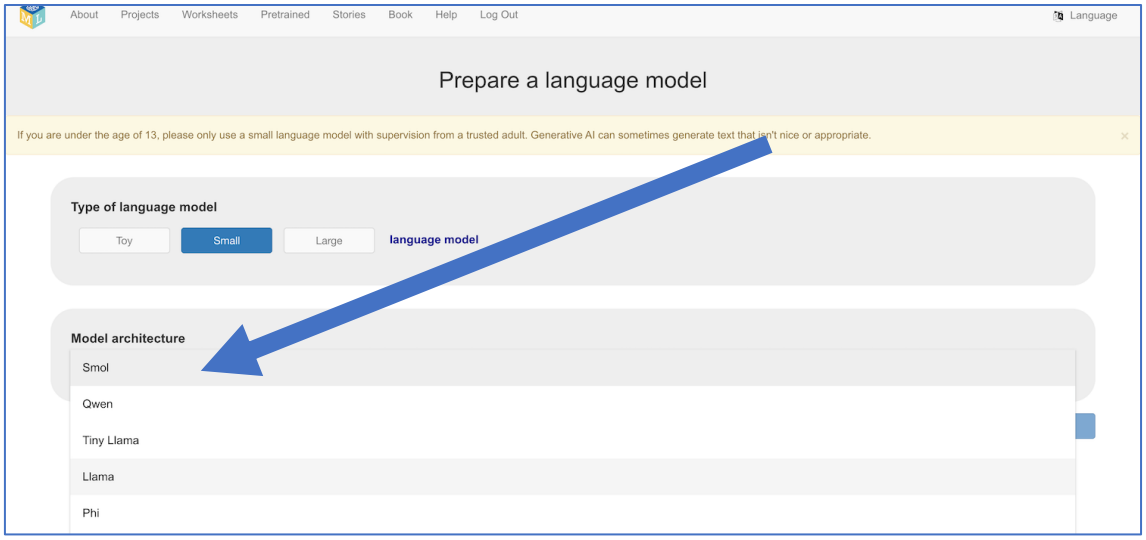
7. Click **Create**

8. You should see your new project in the projects list. Click on it.

9. Click on **Small**, and then click **Next**



10. Choose a model architecture



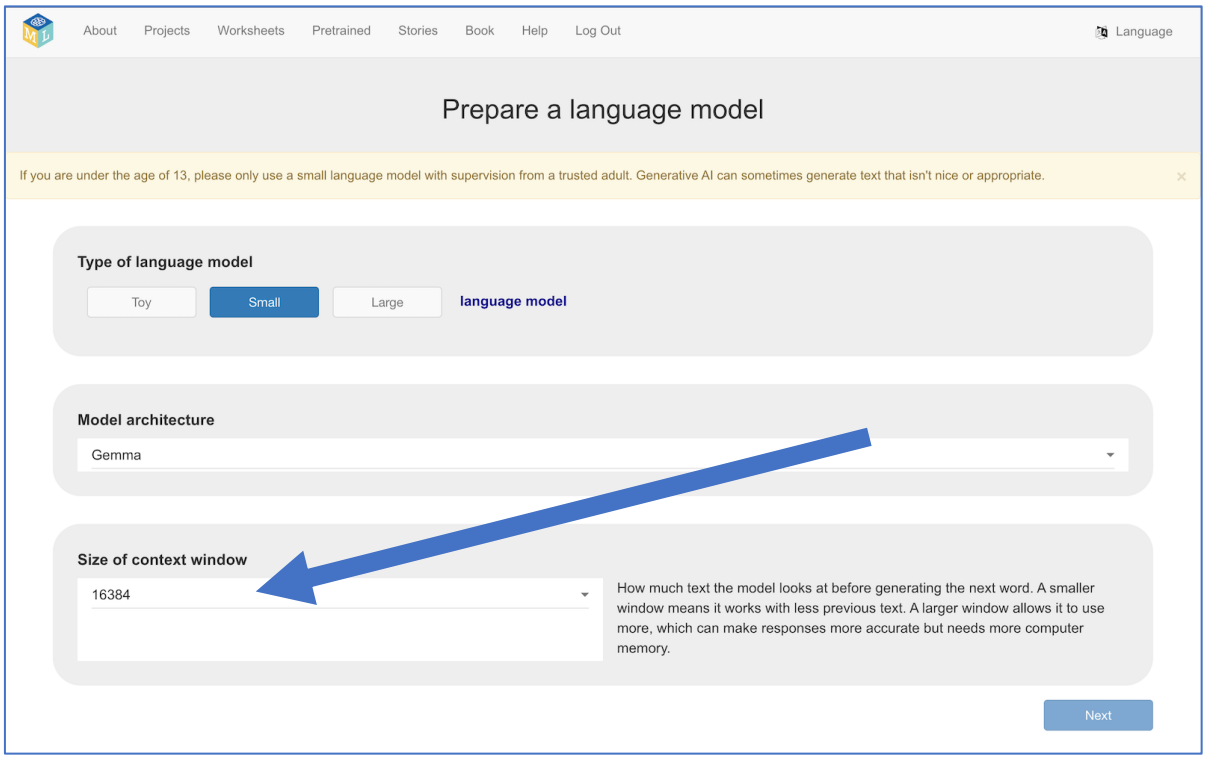
*This is a complex project. It will benefit from the **largest model you can use**.*

***But** larger models:*
** take longer to download*
** need more storage space on your computer*
** need a faster and more powerful computer to run*

Ask your teacher or group leader if you are not sure how large a model you can pick.

11. Click **Download**

12. Choose a context window size



*This is a complex project. It will benefit from the **largest context window you can use**.*

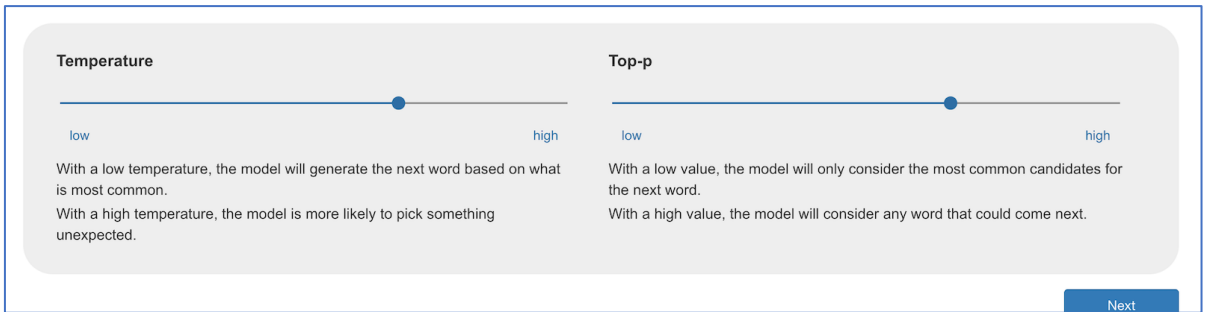
***But** larger context windows need a faster and more powerful computer to run*

Ask your teacher or group leader if you are not sure how large a window to pick.

13. Click **Next**

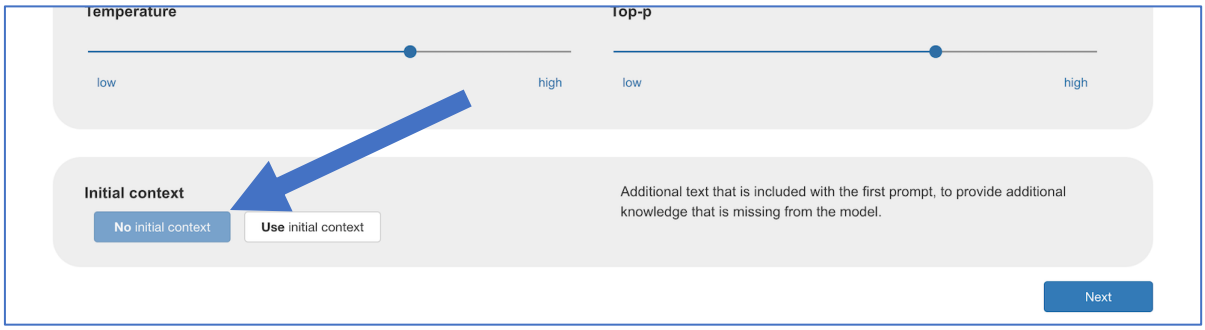
14. Set **temperature** and **Top-p** values

Values somewhere between half-way and full are a good place to start



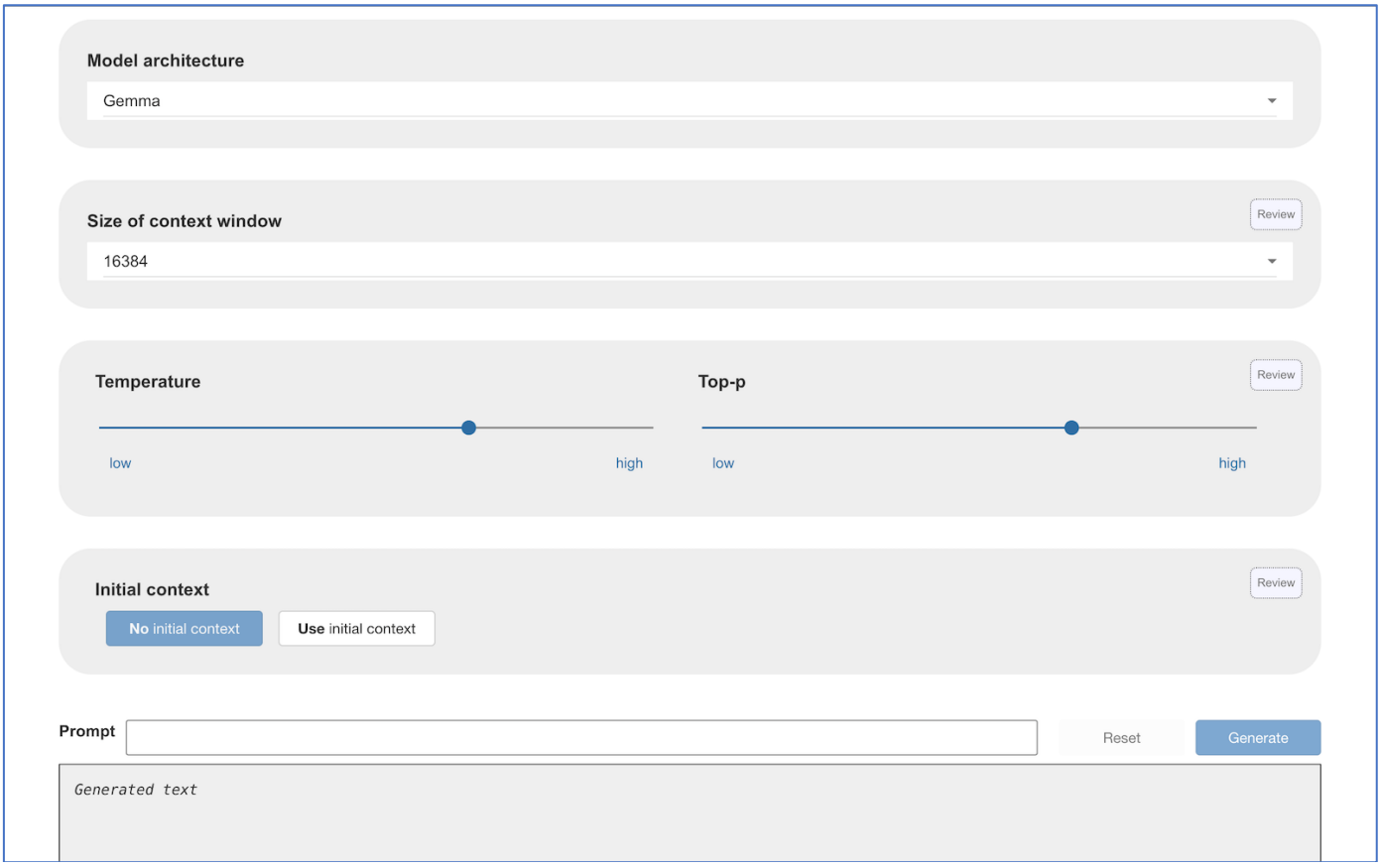
15. Click **Next**

16. Select **No initial context**



17. Click **Next**

Now that you have a small language model ready to use, give it a try by putting a few simple questions in the **Prompt** box and click **Generate** to check that everything is working.



Click on the **Review** buttons if you would like to change any of the decisions that you made.

18. Ask a general knowledge question that everyone knows

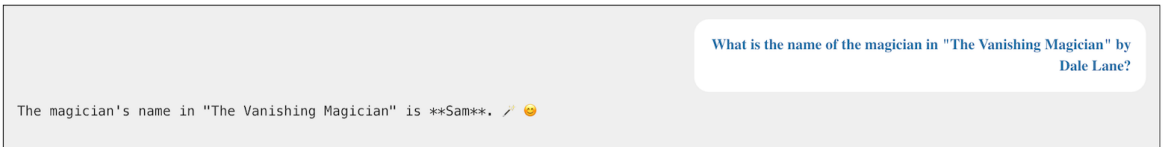
For example, "What city is the Statue of Liberty in?"

The model will hopefully give a correct answer.

19. Click **Reset**

20. Ask a question that relies on knowledge that has never been online

For example, I asked a question about a story I wrote

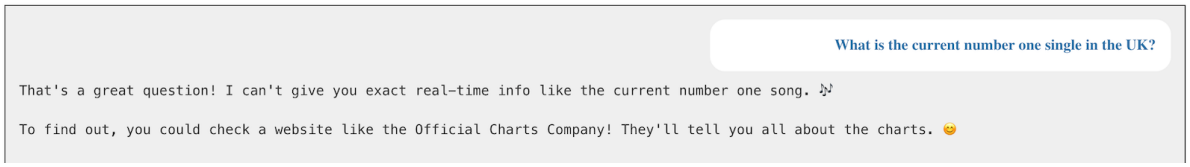


The model might say that it doesn't know.
Or it may "hallucinate" (make up an incorrect answer)

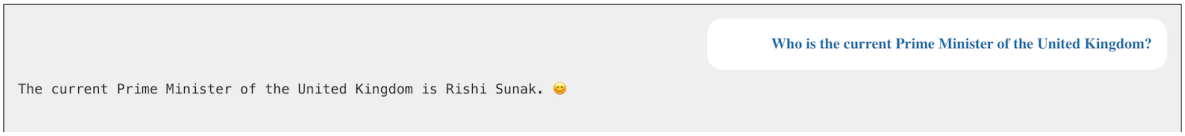
21. Click **Reset**

22. Ask a question that relies on knowledge of something recent

For example, I asked a question about the music charts



For example, I asked a question about election results



The model might say that it doesn't know.
It may give out of date information.
Or it may "hallucinate" (make up an incorrect answer)

What is happening?

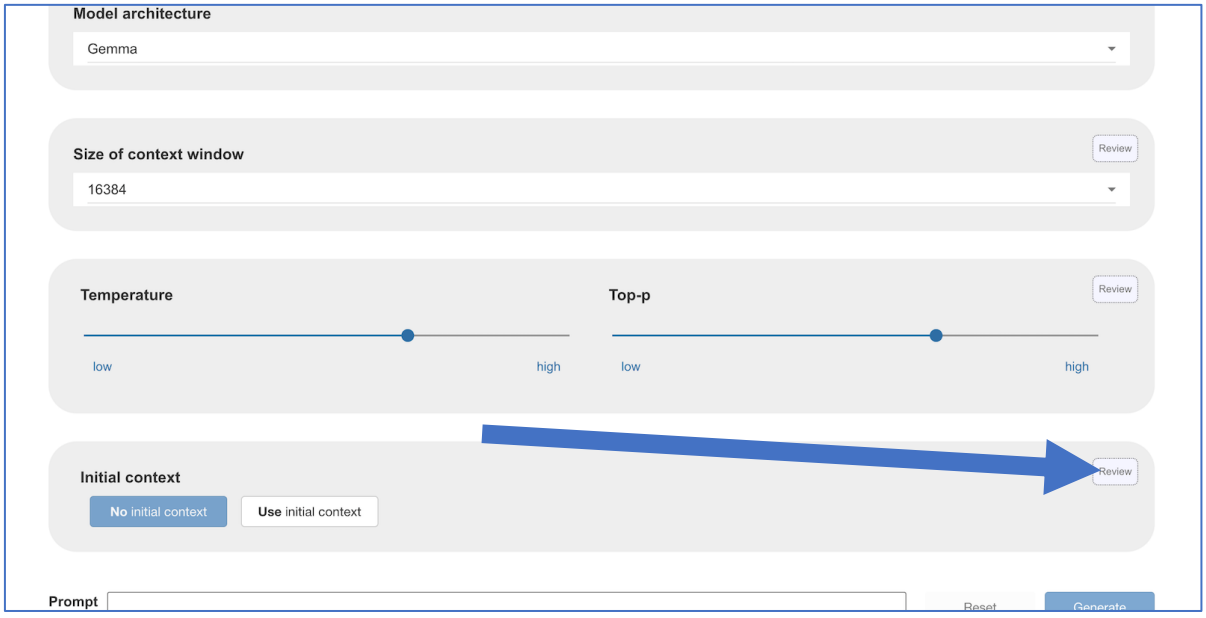
Language models generate text using the information that was used to train them.

Training these models takes months, so it cannot be done very frequently. The model you are using might have been trained up to 6 – 12 months ago. It is not possible for it to contain knowledge about things that have happened since then.

If something is private and cannot be guessed from public information, it cannot contain knowledge about those things, whenever they happened.

In this project, you will learn how we can deal with these issues.

23. Click on **Review** for the **Initial context**



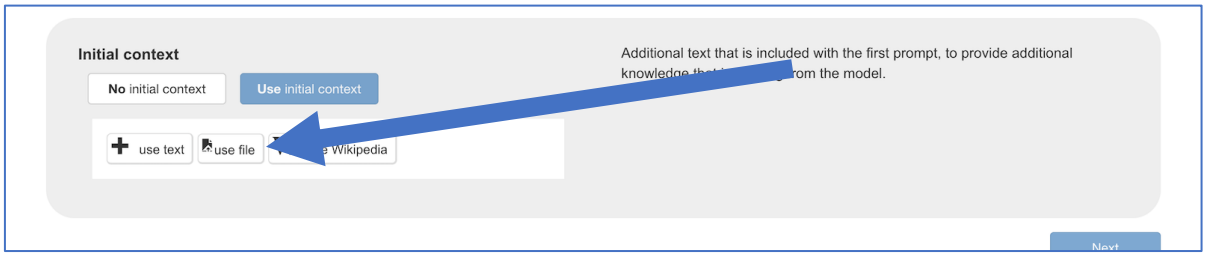
24. Click on **Use initial context**

25. Collect text that contains knowledge to answer your first question.

I have the story that I wrote – "The Vanishing Magician"
If you don't have anything like that, write a few lines now.
It doesn't need to be long.

26. Add your text as the initial context

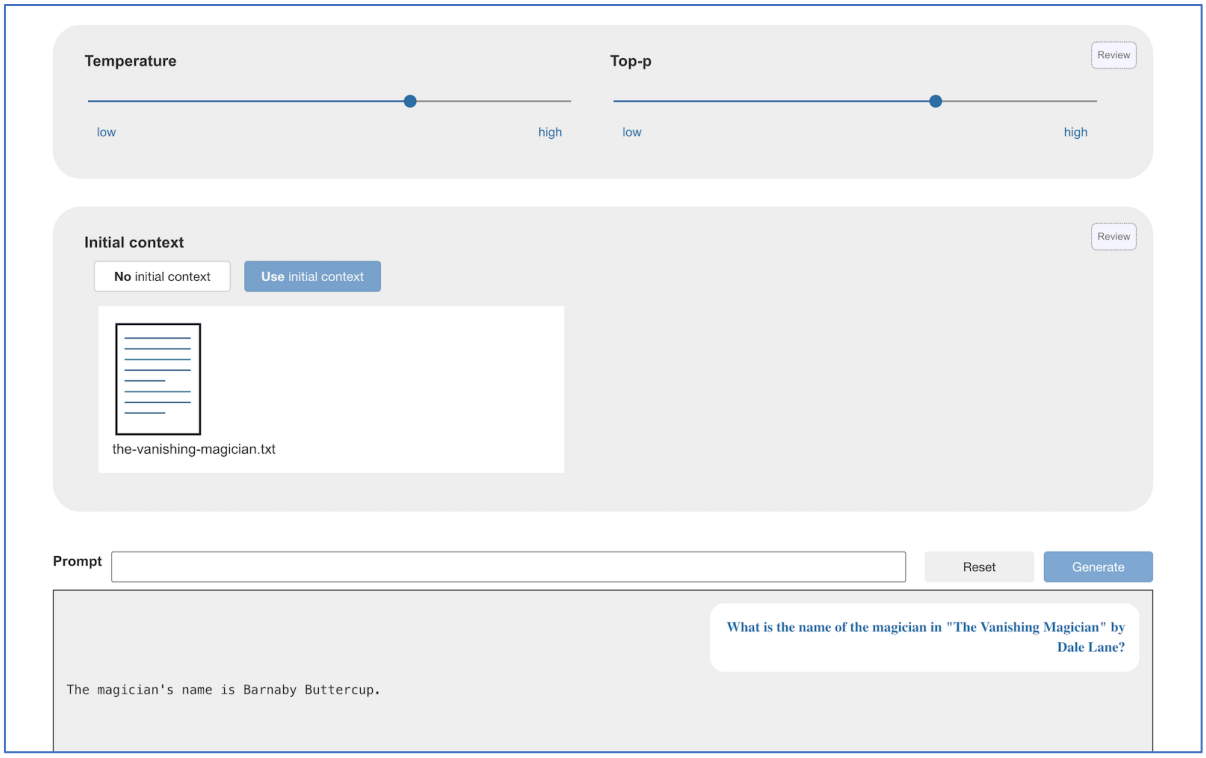
*If you have it in a text (txt) file, click on **use file***
*Otherwise, click on **use text** and copy/paste it in*



27. Click on **Next**

28. Ask your first question again

I asked my first question again, about the main character of my story.
But this time, the text for my story was included in the initial context.

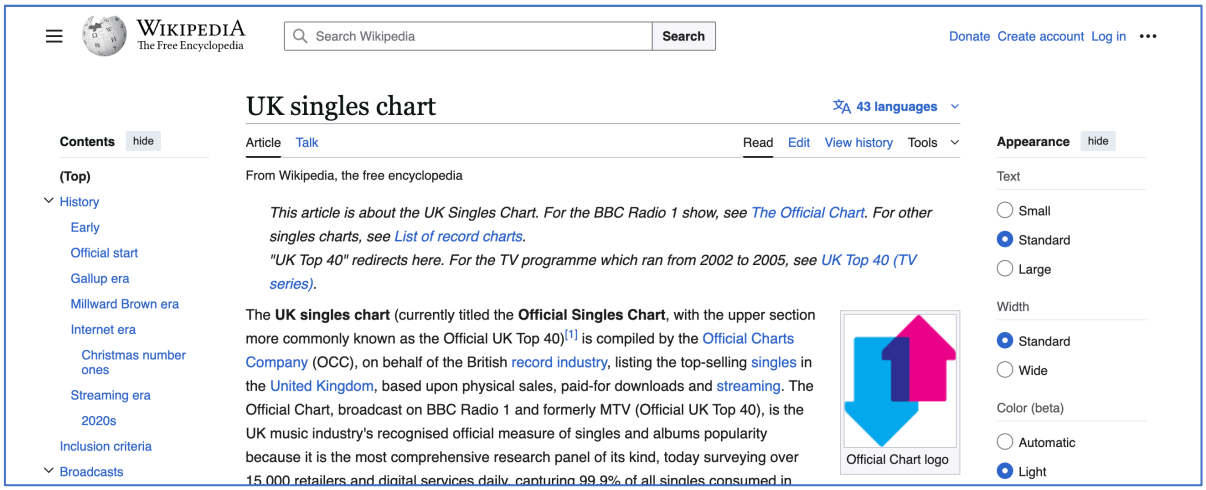


If the answer to the question can be found in the initial context, the model will hopefully have found it and given a correct answer.

If the model does not give you the answer you expected, click on **Reset** and try again. If it still does not give the correct answer, rephrase your question.

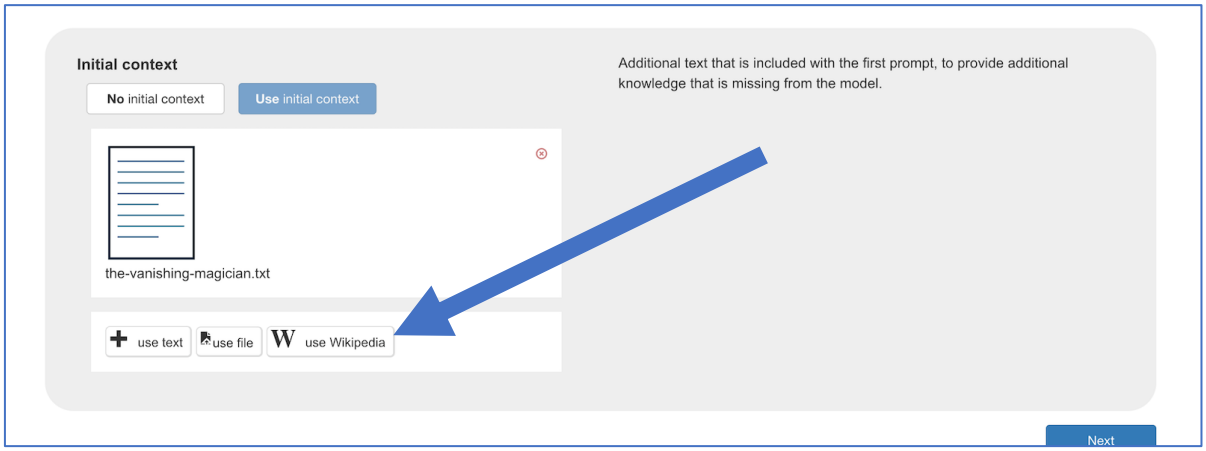
29. In a separate browser window, find a Wikipedia page that has the answer to your second question

*For example, the page "**UK singles chart**" has the answer to my question "What is the current number one single in the UK?"*

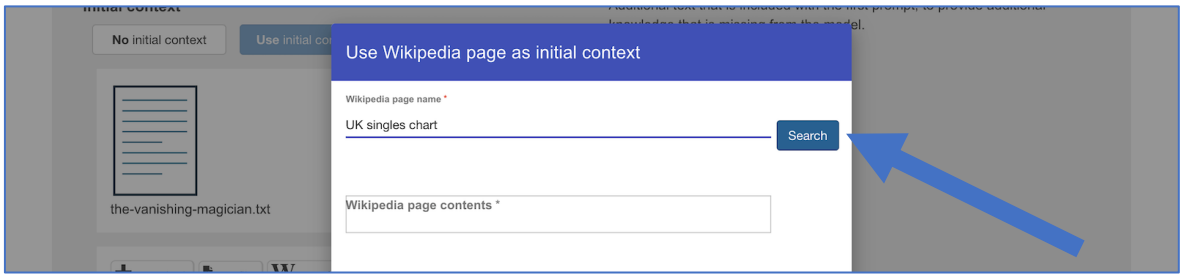


30. Click on **Review** for the **Initial context** again

31. Click on **use Wikipedia**

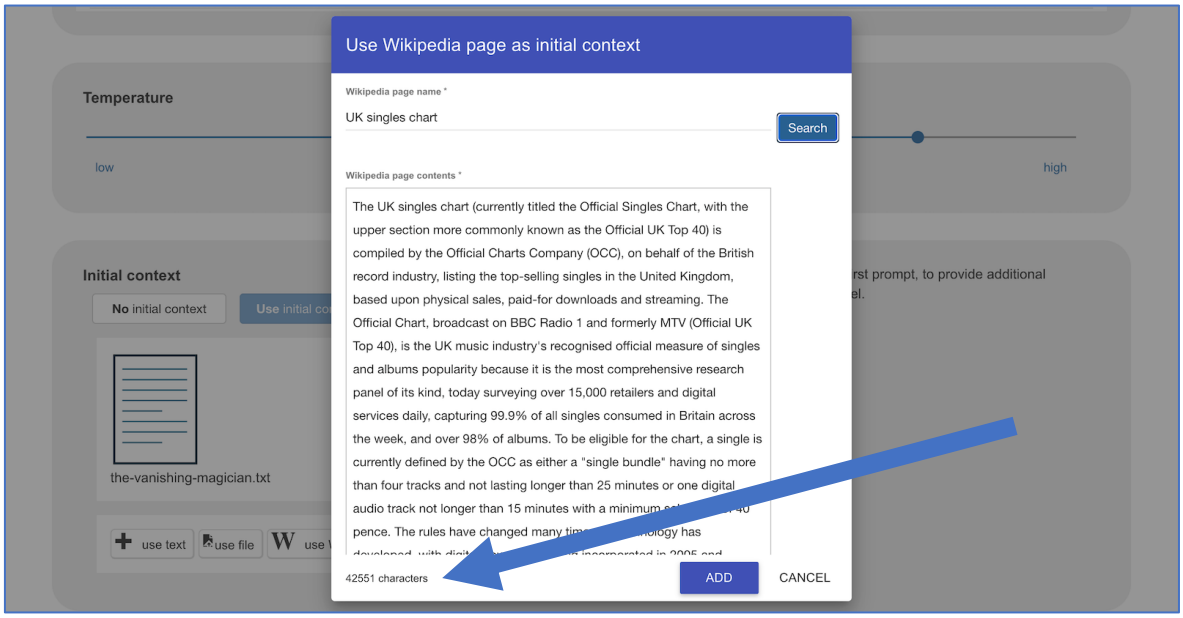


32. Enter the name of the Wikipedia page you found. Click on **Search**



33. Look at the size of the Wikipedia page

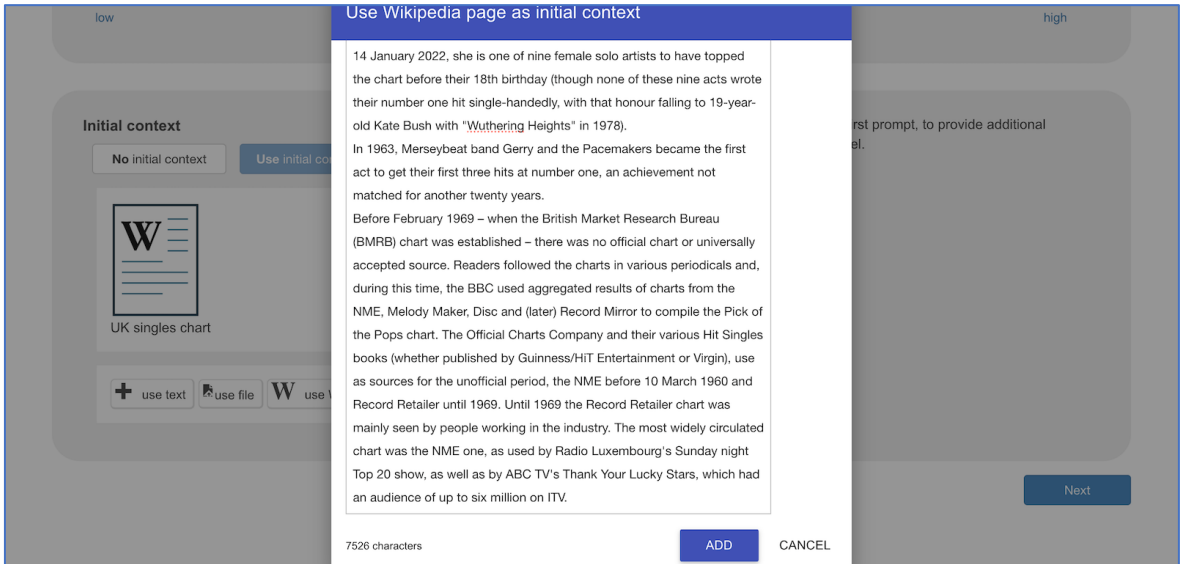
If it is very large, your computer will need a lot of memory to keep this whole page in the context while it answers your question.



34. Delete the end of the page until it is smaller than 10000 characters

*Delete **from the bottom** of the page – keep what is at the start*

Be careful not to delete the answer the model needs!



35. Make a note of approximately how many characters you kept

For the "UK singles chart" page, I kept about the first 7500 characters

36. Click on **Add**

37. Click on **Next**

38. Ask your second question again

I asked my question about the song at the top of the UK charts again. But this time, recent chart information was included in the initial context.

If the answer to the question can be found in the initial context, the model will hopefully have found it and given a correct answer.

If the model does not give you the answer you expected, click on **Reset** and try again. Rephrasing your question may also help.

39. In a separate browser window, find a Wikipedia page with the answer to your last question

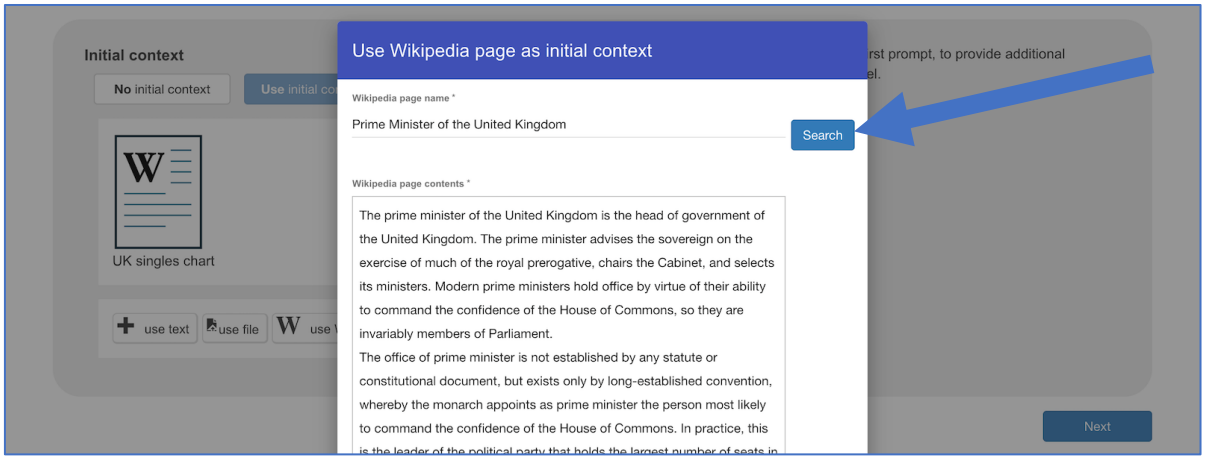
*For example, the page "**Prime Minister of the United Kingdom**" has the answer to "Who is the current Prime Minister of the United Kingdom?"*

40. Click on **Review** for the **Initial context** again

41. Click on **use Wikipedia**

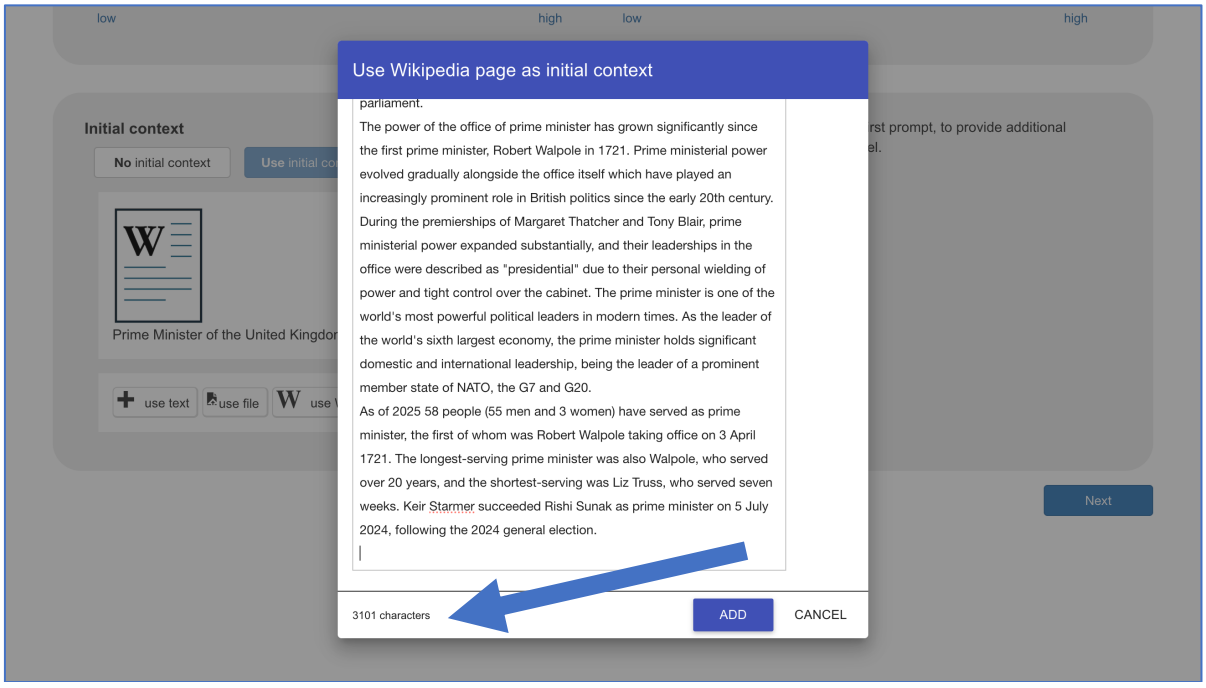
42. Enter the title for your new Wikipedia page

43. Click on **Search**



44. As before, delete **from the bottom** of the page to reduce the size of the initial context

For my Wikipedia page, keeping the first 3100 characters was enough to still include examples of the correct answer



45. Click on **Add**

46. Click on **Next**

47. Ask your third question again

I asked my question about the current UK prime minister. But this time, recent information about the PM was included in the initial context.

This is called "RAG"

You have been searching for documents with additional information that the language model does not already have. You added these to the context for your question to improve the answers the model can give.

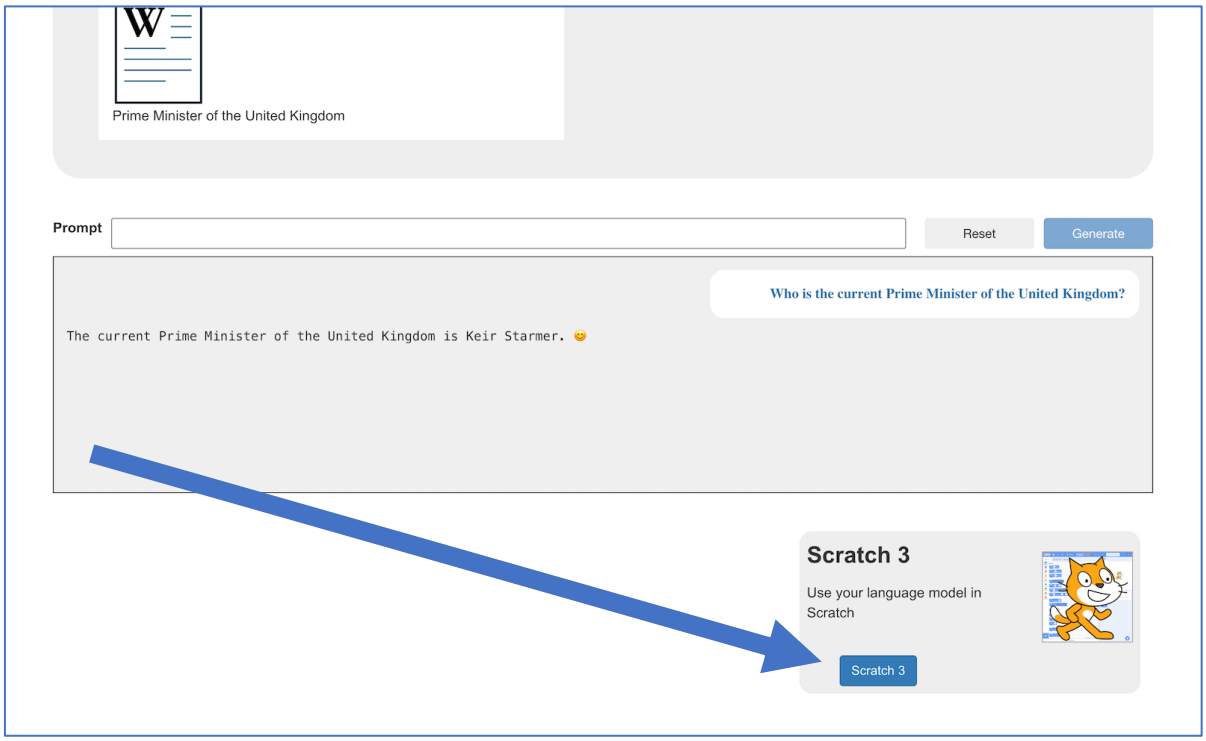
This technique is called "Retrieval Augmented Generation" because we are **retrieving** documents and using these to **augment** (improve) the answers that the language model can **generate**.

We call this **RAG** for short.

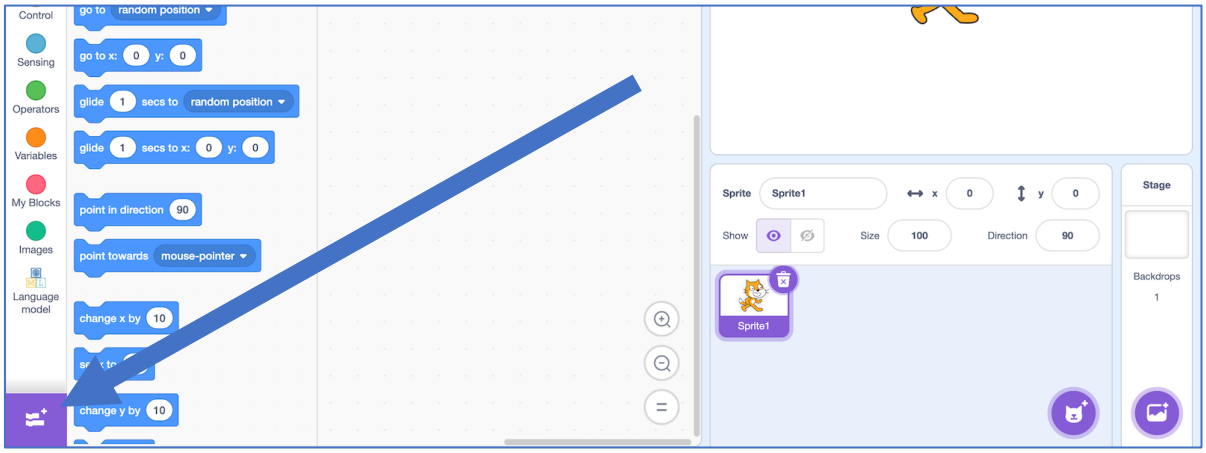
RAG projects often don't use only one document for this – often the top three or top five search results for the question could be included in the initial context to give the model the best chance to find the answer.

You are running your RAG project on your own computer instead of a full set of servers, so you need to cut that down - but the technique is the same.

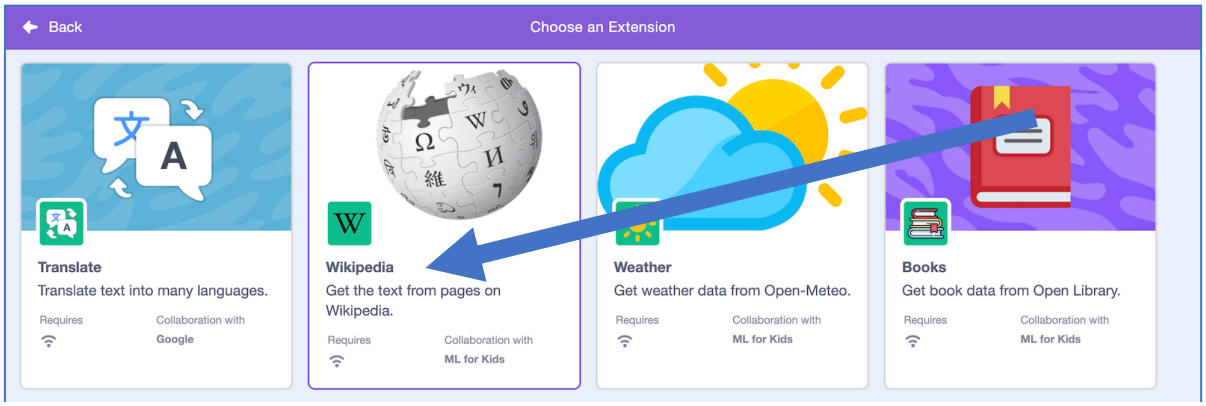
48. Click on **Scratch 3**



49. Open the **Extensions**

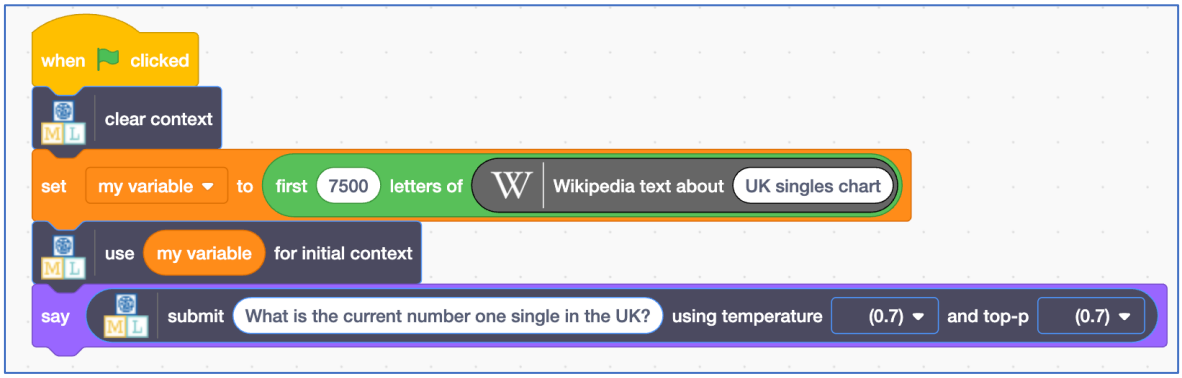


50. Add the **Wikipedia** extension to your project

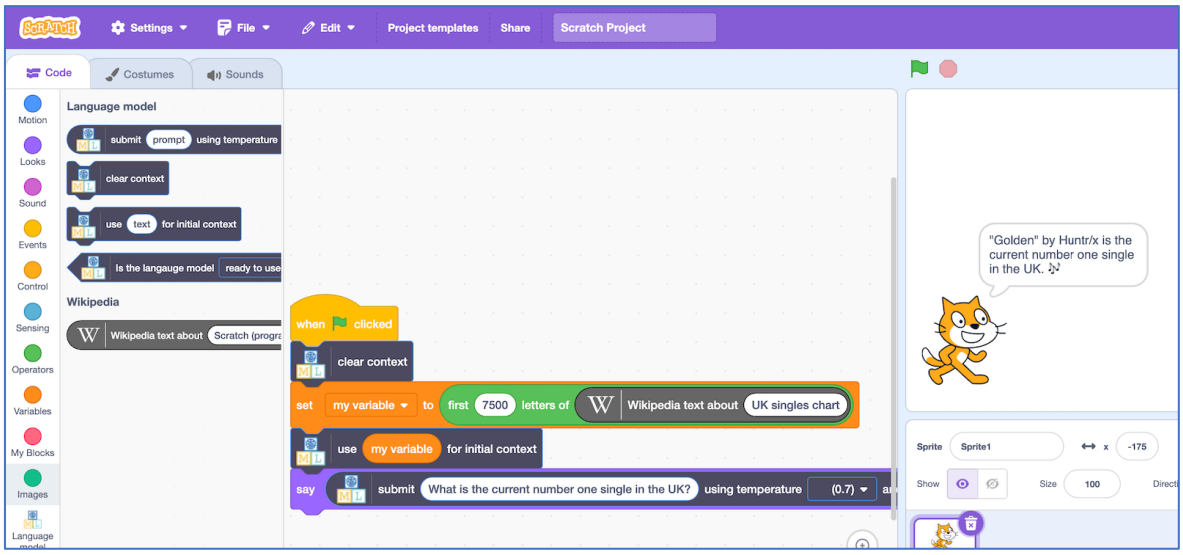


51. Create the following code

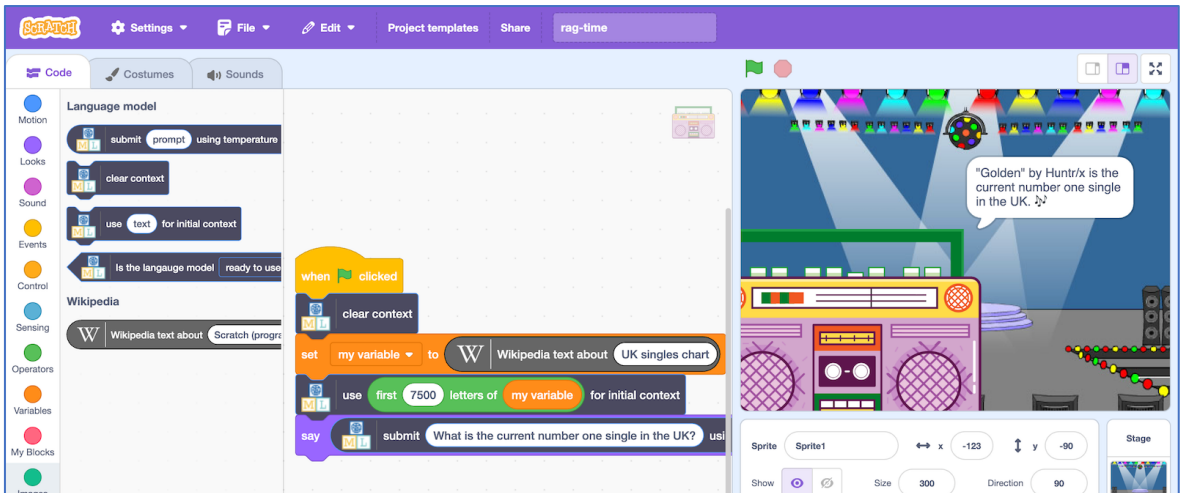
Replace these bits with what you found worked well in your testing:
*"**7500**" – replace this with how much of the Wikipedia page you needed to keep*
*"**UK singles chart**" – replace this with the name of the Wikipedia page you used*
*"**What is the current number one single in the UK?**" – replace with your question*



52. Click the **Green Flag**



53. Try adding sprites, backgrounds, animations, or sounds



What have you done?

You've created a project that blends live text from the Internet with a language model's ability to work with text.

Live data from Wikipedia means whenever you run this project it will have fresh, current data. If you run it next week, it will use different data to what it uses today.

You are using the language model's ability to "understand" natural language to interpret the live data, and display the information you are interested in.

This is a common approach in artificial intelligence projects today, enhancing what is possible with language models by themselves.

If you include context from your private documents, you can create a system that can answer questions about your private content the model alone doesn't have any information about.

If you include context from recent documents, you can create a system that can answer questions about current events that occurred after the model was trained.

This is a common approach, often called "RAG".

A similar approach is described as "Model Context Protocol" (MCP) - which includes agreeing (a **protocol**) on the ways to provide additional **context** to a **model**.

These techniques are essential for creating useful systems with language models.