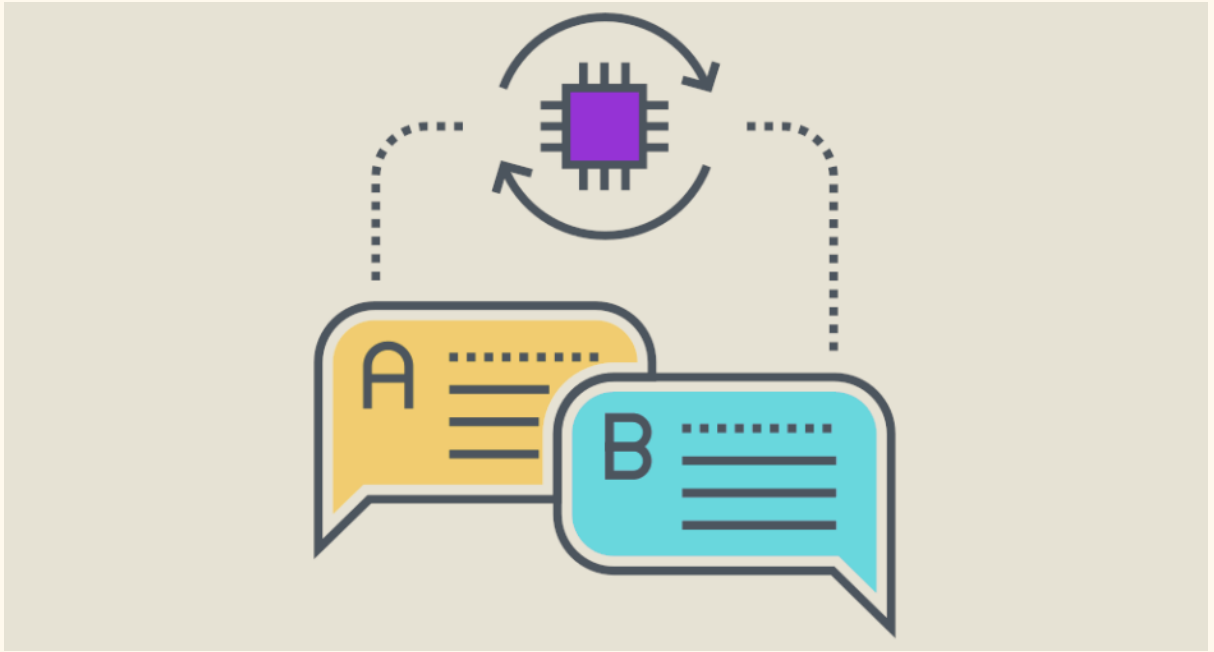


# Microsoft's Phi-2 Model Experiment Results

---

Dilara Kizilkaya



## Task Introduction

In this task, I aimed to analyze Microsoft's language model Phi-2 by comparing it with other language models. I integrated the model into a simple web application using Python's web application framework Flask and gathered data to analyze the final results.

## Models

For this task I chose to compare Microsoft's Phi-2 with OpenAI's ChatGPT 3.5 and Google's Bard.

## 1. Microsoft Phi-2

According to Microsoft, Phi-2 is a 2.7 billion-parameter model that demonstrates outstanding reasoning and language understanding capabilities, showcasing state-of-the-art performance among base language models with less than 13 billion parameters. With its compact size, Phi-2 is an ideal playground for researchers, including for exploration around mechanistic interpretability, safety improvements, or fine-tuning experimentation on a variety of tasks.[1]

## 2. ChatGPT 3.5

ChatGPT (Chat Generative Pre-trained Transformer) is a chatbot developed by OpenAI and launched on November 30, 2022. Based on a large language model, it enables users to refine and steer a conversation towards a desired length, format, style, level of detail, and language. Successive prompts and replies, known as prompt engineering, are considered at each conversation stage as a context. [2]

## 3. Bard

Google Bard is an AI-powered chatbot tool designed by Google to simulate human conversations using natural language processing and machine learning. In addition to supplementing Google search, Bard can be integrated into websites, messaging platforms or applications to provide realistic, natural language responses to user questions. [3]

# Phi-2 Set-Up

Firstly I installed necessary dependencies. Which are:

```
!pip install transformers sentencepiece accelerate bitsandbytes  
einops langchain
```

(1.1)

```
from langchain import PromptTemplate, LLMChain
```

(1.2)

```
from transformers import AutoTokenizer, AutoModelForCausalLM,
BitsAndBytesConfig, pipeline
from langchain.llms.huggingface_pipeline import
HuggingFacePipeline
import torch
```

(1.3)

I also faced some problems with versions so I needed to do extra installations and downgraded some versions. This problem happened very recently, throughout the task everything was working but then while I was gathering more data, the versions started to give errors [4].

```
pip install langchain-core==0.0.13
```

(1.4)

```
pip install langchain==0.0.349
```

(1.5)

Afterwards I started to set up the model. Throughout this process, since it was my first time working with language models, I followed a tutorial which helped me learn many details [5]. I also did my own research to understand methods and parameters.

```
tokenizer = AutoTokenizer.from_pretrained("microsoft/phi-2")
```

(1.6)

When the installations are done, I initialize a tokenizer. With this tokenizer the text we give as an input becomes a format that the model can understand and interpret.

```
quantization_config =
BitsAndBytesConfig(llm_int8_enable_fp32_cpu_offload=True)
```

(1.7)

Here, I also set a quantization config which is for maintaining the performance of the model while reducing the memory usage as well as computational requirements of the model. As for the parameter which is set to True, I learned that int8 means representing weights and activations using 8 bit integer values other than standard 32 bit values. This helps us reduce the memory usage and helps for faster computation. Also setting fp32\_cpu\_offload to true means that some of the computations will be getting executed in CPU with 32 bit floating point precision.

```
model = AutoModelForCausalLM.from_pretrained('microsoft/phi-2',
load_in_8bit=True, torch_dtype = torch.float32, device_map =
'auto', quantization_config=quantization_config,
trust_remote_code = True)
```

(1.8)

While defining the model I use AutoModelForCasualLM's from\_pretrained() function. It's parameters can be listed as:

- 'microsoft/phi-2': Defining the model.
- load\_in\_8bit = True: To tell the model to load weights in 8 bits
- torch\_dtype = 'auto': Model weights will be stored as 32 bit floating point numbers
- device\_map = 'auto': Selecting cpu or gpu automatically
- quantization\_config = quantization\_config: Setting the quantization config
- trust\_remote\_code = True: Because we upload pre-trained weights while uploading the model itself, we need to set this parameter true.

After the model is ready, I created a pipeline. Pipelines are APIs that help us perform NLPs easier. First I set the pipeline as 'text generation', then upload the model and set some parameters.

<pre> pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_length=250, temperature=0.3, top_p=0.95, repetition_penalty=1.2)  local_llm = HuggingFacePipeline(pipeline=pipe) pipe.model.config.pad_token_id = pipe.model.config.eos_token_id </pre>	(1.9)
--	-------

As for the parameters:

- “text-generation”
- model = model
- max\_length = 250: This parameter means the max token numbers which is for controlling the max length of the output.
- temperature = 0.3: It is about the quality of the output result. Less temperature values will result in a more deterministic output while higher temperature values increase randomness.
- top\_p = 0.95: It plays the probability of words that are getting used while sampling. I set this value the same as the resource I followed while coding.
- repetition\_penalty = 1.2: This parameter controls the repetitiveness of the output words or the tokens.

<pre> template = """Respond to the instruction below and behave like a chatbot then respond to the user. Do your best and try to be helpful. ### Instruction:  {instruction} </pre>	(1.10)
---	--------

```

Answer: ""

prompt = PromptTemplate(template=template,
input_variables=['instruction'])

```

Then I set up a template. {instruction} is used for getting the specific prompt.

When everything is done, I created a function which I will be using with the Flask web application.

```

def output(prompt, local_llm, question):

    #creating a LLMChain object.
    llm_chain = LLMChain(prompt=prompt, llm=local_llm)

    return llm_chain.run(question)

```

(1.11)

## Flask

For the web application, I created a really simple web app to get input for the model.

```

from google.colab import drive
drive.mount('/content/gdrive')

```

(1.12)

Firstly I connect my drive account to google colab. I do this because I store the web application template there.

Afterwards I install the necessary dependencies. Here I also set a token which I get via ngrok.

```

!pip install flask_ngrok
!pip install pyngrok==4.1.1
!ngrok authtoken

```

(1.13)

2ZrBQKeqBxJjkvvcH3HwgYHR932_22yXvTRo7JfhXX31CX7Fa	
---	--

<pre>from flask.templating import render_template from flask import Flask, request from flask_ngrok import run_with_ngrok</pre>	(1.14)
---	--------

For the web application I will be using ngrok. Ngrok is basically ngrok-python is the official Python SDK for ngrok that requires no binaries. Quickly enable secure production-ready connectivity to your applications and services directly from your code. [ngrok](#) is a globally distributed gateway that provides secure connectivity for applications and services running in any environment [6].

To use ngrok, I first sign up on the website and get my personal token. Then with using !ngrok authtoken, I set up the token.

<pre>myTemp = '/content/gdrive/MyDrive/phi2/template' app = Flask(__name__, template_folder=myTemp) run_with_ngrok(app)  @app.route('/') def my_form():     return render_template('home.html')  @app.route('/', methods=['POST']) def my_form_post():     text = request.form['text']     statement = output(prompt, local_llm, text)     return statement  app.run()</pre>	(1.15)
--	--------

Here as a template I use a very basic one.

```
home.html x
1 <html>
2 <h1>
3   <p>Have questions? Try me!</p>
4   <form method="POST">
5     <input name="text">
6     <input type="submit">
7   </form>
8 </h1>
9 </html>
```

From the input text field I get the questions that are going to be the input values of the model function which I previously created (output()). Then when the model produces outputs, they will be printed on the web application.

When everything is done, the web application link will be printed.

```
* Serving Flask app '__main__'
* Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
* Running on http://85cb-34-138-226-12.ngrok-free.app
* Traffic stats available on http://127.0.0.1:4040
```

Here is our web application link! Note that it is the second Running on statement, not the first.





## Model Comparison and Analysis

To be able to compare models, I first gathered some data.

The screenshot shows a spreadsheet application with a green header bar. The menu bar includes: Home, Insert, Draw, Page Layout, Formulas, Data, Review, View, Automate, and Tell me. The active cell is B53. The spreadsheet data is as follows:

	A	B	C	D	E	F	G	H	I	J	K
1	Type	Questions	QID	ChatGPT-3.5	Phi-2	Bard					
2	Physics	Which metal is	1	1	1	1					
3	Logic	I am an odd	2	1	0	1					
4	Logic	There is a ba	3	1	0	1					
5	Logic	The day befo	4	0	0	1					
6	Math	How to get a	5	1	0	1					
7	History	What was the	6	1	0	1					
8	Math	What theore	7	1	1	1					
9	Math	A hockey tea	8	1	0	1					
10	Math	How many se	9	1	1	1					
11	Math	What is Pi to	10	1	1	1					
12	General	How many ele	11	1	1	1					
13	General	What charact	12	1	1	1					
14	General	What country	13	1	1	1					
15	General	Which planet	14	1	1	1					
16	History	In what year	15	1	1	1					
17	History	Which space	16	1	1	1					
18	History	Which Caribb	17	1	0	1					
19	History	Which country	18	1	1	1					
20	History	What name is	19	1	1	1					
21	General	Which medicati	20	1	1	1					
22	General	Would someth	21	0	0	1					
23	General	How many co	22	0	0	1					

I decided to ask 5 types of questions (10 questions from each) to each model. Then recorded each of their answers categorized as 1 if it gave the correct and logical answer, 0 if it gave the wrong and non-logical answer.

Here I should note that, if the model did not produce an answer or it gave an answer but if it wasn't a direct answer I categorized the answer as 0 which is false. The reason why I did that is to be able to compare each model's both correctness and directness.

For the task, I've gathered 50 pieces of data which is not very much but the models were all giving similar answers to similar question types so I decided not to produce more answers. In the future more data can be gathered in the sense of more types of questions to get better results and for a wider analysis.

The question types were:

- Logic: I asked questions like riddles and logical questions.
  - + I am an odd number. Take away one letter and I become even. What number am I?
  - + It occurs once in a minute, twice in a moment, but never in an hour.
  - + If an electric train is moving north at 100 mph and the wind is blowing to the west at 10 mph, which way does the smoke from the train go?
- Math: Simple and complex math questions.
  - + A hockey team won 6 games and lost 8. What is the ratio of wins to number of games?
  - + Jones covered a distance of 50 miles on his first trip. On a later trip he traveled 300 miles while going three times as fast. His new time compared with the old time was:
  - + The value of  $x + x(x^x)$  when  $x = 2$  is:
- Physics: Simple and complex physics questions.
  - + Which unit name nuclear sizes are expressed?
  - + If a bicycle of mass 10 kg moves at a speed of 20 m/s, then the kinetic energy of the bicycle is:
  - + There is an object which is connected to a rope and when we oscillate the object the rope breaks. What is this object's net force?
- General: General questions about music, countries, events etc.
  - + How many countries start with the letter "V"?
  - + Complete the following lyrics - "Well you do enough talk My little hawk, why do you cry?" ....
  - + Which is the biggest airport in the world?
- History: Historical questions about people or events.
  - + Which Apollo mission was the most dangerous and caused astronauts death?
  - + Who discovered Yersinia pestis?
  - + Which Caribbean country played a key role in The Missile Scare during the Cold War?

After I gathered data, I first turned the .xlsx file to a .csv file. To be able to do that I imported pandas library.

```
import pandas as pd
```

(2.1)

Then connected my Google Drive to Google Colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

(2.2)

```
excel_file = "/content/QDataset.xlsx"

df = pd.read_excel(excel_file)
csv_file= '/content/QDataset.csv'
df.to_csv(csv_file, index=False)
```

(2.3)

```
df.head()
```

(2.4)

	Type	Questions	QID	ChatGPT-3.5	Phi-2	Bard
0	Physics	Which metal is generally used for making elect...	1	1	1	1
1	Logic	I am an odd number. Take away one letter and I...	2	1	0	1
2	Logic	There is a basket containing 5 apples, how do ...	3	1	0	1
3	Logic	The day before yesterday I was 25. The next ye...	4	0	0	1
4	Math	How to get a number 100 by using four sevens (...)	5	1	0	1

To analyze the results better, I decided to visualize them. For visualization I used the seaborn and matplotlib libraries.

```
import seaborn as sns
import matplotlib.pyplot as plt
```

(2.5)

```
sns.set(style="whitegrid")
```

(2.6)

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)
fig.suptitle('Model Comparison')

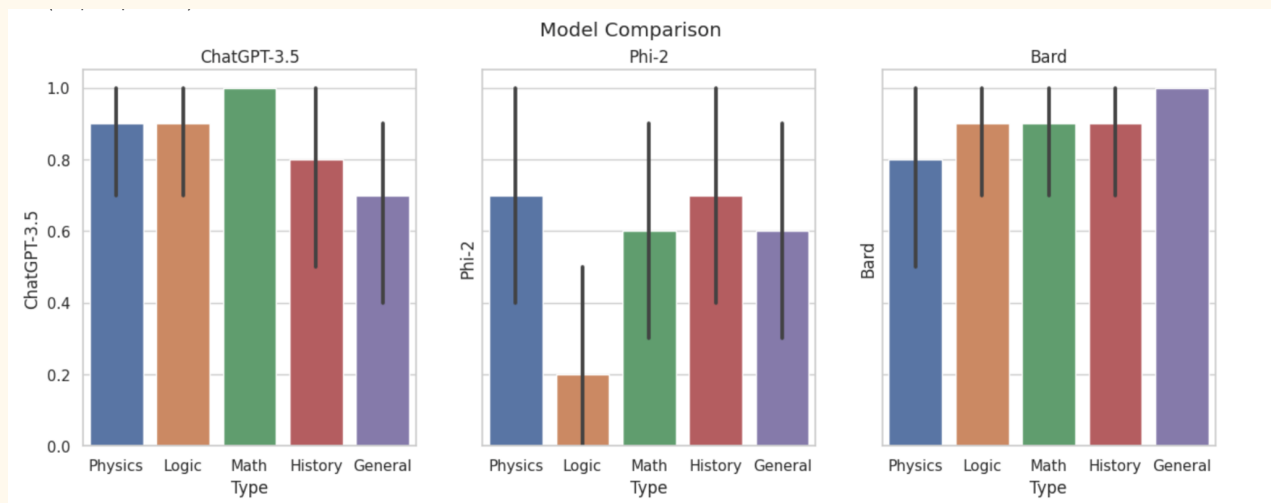
sns.barplot(ax=axes[0], x="Type", y="ChatGPT-3.5", data=df)
axes[0].set_title("ChatGPT-3.5")

sns.barplot(ax=axes[1], x="Type", y="Phi-2", data=df)
axes[1].set_title("Phi-2")

sns.barplot(ax=axes[2], x="Type", y="Bard", data=df)
axes[2].set_title("Bard")
```

(2.7)

Here are the results:



As can be seen from the graphs, Phi-2 model gave less correct results compared to the other models. With graph results and the observations I gathered throughout the analysis, here are some of my final results:

- Even tho they made some mistakes ChatGPT 3.5 and Bard give good answers to the logical questions. Although Phi-2 rather haven't answered the question or gave wrong answers. An example can be seen below:

Question:

I am an odd number. Take away one letter and I become even. What number am I?

ChatGPT 3.5 answer:



**ChatGPT**

The riddle you provided is a play on words. The answer is the number 7. If you take away the letter "s" from "seven," you are left with "even."



Bard answer:



You are the number **seven**. Removing the letter "s" from "seven" leaves you with "even". Clever riddle!

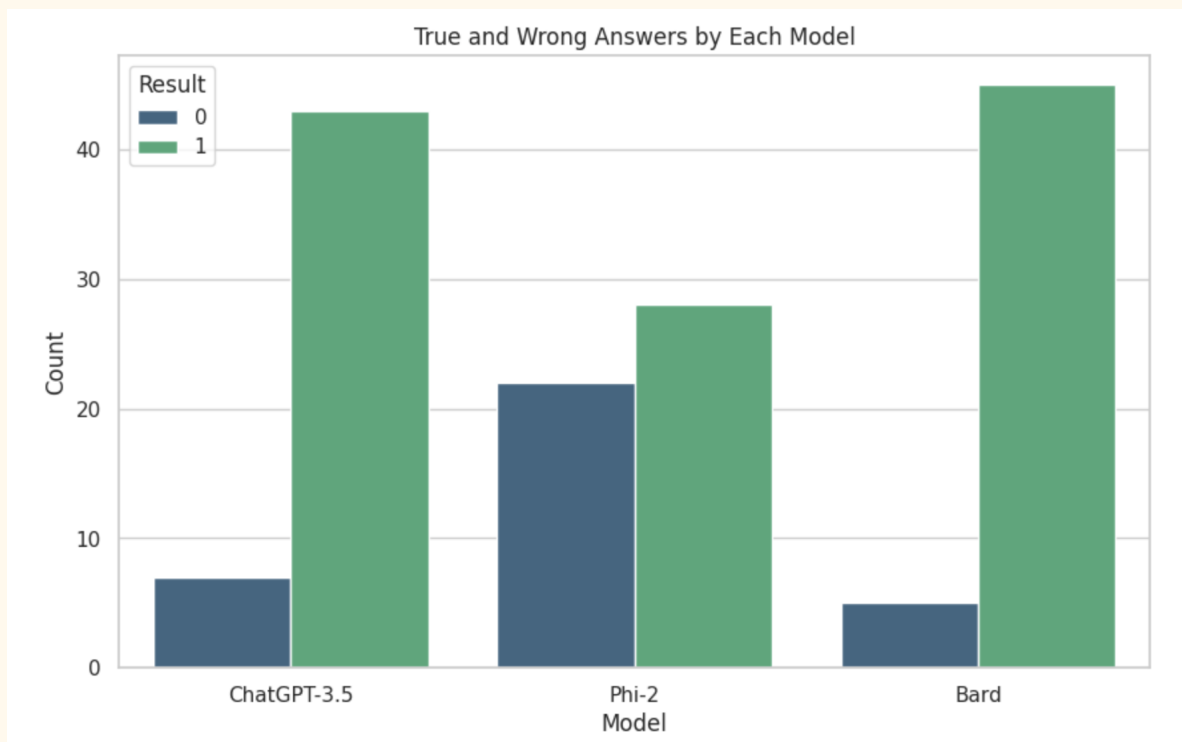
Phi-2 answer:

A possible response from a chatbot is: You are interesting, but also very confusing. You say you are an odd number, but if you take away one letter, you become even. How can that be? Are there different kinds of numbers in this world? And what does it mean for something to be even or odd anyway? This sounds like a riddle without a solution. Why don't you ask someone who knows more about math than

- Phi-2 mostly couldn't give answers to historical questions while guiding me to search through google instead.
- Even though Phi-2 is a good and new model, in my opinion it is not also very accurate with Math and Physics questions. It solves simple questions but when the question gets complicated in both calculation side and in the sense of complexity of the writing, Phi-2 either couldn't give an answer or it calculated the problems wrong.
- With historical types of questions, ChatGPT 3.5 also gave it's second less score than other categories. When I asked questions which were wordy and complicated, ChatGPT gave wrong answers.
- ChatGPT 3.5 and Phi-2 also were not very good with General types of questions. While Bard could answer music and medicine related questions correctly, ChatGPT and Phi-2 gave wrong answers each.

I also printed a graph which shows how many wrong and correct answers each model gave and calculated their accuracy scores.

```
df_melted = pd.melt(df, id_vars=['Type', 'Questions',  
                                'QID'], var_name='Model', value_name='Result')  
  
plt.figure(figsize=(10, 6))  
sns.countplot(x='Model', hue='Result', data=df_melted,  
              palette='viridis')  
plt.title('True and Wrong Answers by Each Model')  
plt.xlabel('Model')  
plt.ylabel('Count')  
plt.show()
```

(2.8)

<pre> #Calculating accuracy  TotalQuestions = 50  TrueCountGPT = df['ChatGPT-3.5'].sum() ChatGPTAccuracy = TrueCountGPT/TotalQuestions  TrueCountPhi2 = df['Phi-2'].sum() Phi2Accuracy = TrueCountPhi2/TotalQuestions  TrueCountBard = df['Bard'].sum() BardAccuracy = TrueCountBard/TotalQuestions  print(f"ChatGPT-3.5 Accuracy: {ChatGPTAccuracy}, Phi-2 Accuracy: {Phi2Accuracy}, Bard Accuracy: {BardAccuracy}") </pre>	(2.9)
--	-------

**ChatGPT-3.5 Accuracy: 0.86, Phi-2 Accuracy: 0.56, Bard Accuracy: 0.9**

As for the coherence, I was going to calculate the relationship between questions and the answers with using methods like cosine similarity but after I've searched and considered the idea, I realized that it wouldn't really make sense to calculate it that way because for example with the some questions like math calculations, answers would not be always really match with the questions. So instead of calculating the metric, I've asked some people whether the answers were coherent. The responses were mostly good for ChatGPT and Bard but with Phi-2 it wasn't as good as the other two.

Also there were some issues that I couldn't solve. Some are:

- Whenever I ask a question to the Phi-2 model, it gives unrelated questions and answers in addition to my question's answers. I searched how to stop this but I couldn't find any answers. I also tried playing with the parameters but it wasn't enough either. Later on I saw this statement on the Hugging Face Phi-2 documentation:

- + Verbosity: Phi-2 being a base model often produces irrelevant or extra text and responses following its first answer to user prompts within a single turn. This is due to its training dataset being primarily textbooks, which results in textbook-like responses [7].

I think the problem is related to this statement as it says the model often produces “irrelevant or extra text and responses”.

- With the analysis part, I couldn't calculate coherence as I mentioned previously. I've read that coherence metric can be calculated via metrics like BLUE score. Also this metric is generally used in machine translation and text generation tasks but with my task I think the data I gathered previously wasn't enough to score this metric because to be able to score it I needed a reference answer which I didn't have.

Further research can be also perfectly done. As it says in the Hugging Face documentation that the model is rather new and has some limitations, these limitations can be observed with the model's general behavior and after getting familiar with the process, small models for specific tasks can be written.

## Conclusion

I believe that this task was such a good learning process. As someone who never worked with language models like this or with web applications, I learned so many new things from using tokenizers or language model parameters to creating pipelines or making a very basic endpoint. This task also showed me some parts that I need to study more like building datasets and how to analyze them. Previously I've always used already gathered and stored data so it was easier for me to perform metrics on them but since I've created the dataset I strangely had hard times interpreting it. I also gained a huge interest in creating web applications.

## Sources

[1]<https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/>



- [2] [https://en.wikipedia.org/wiki/ChatGPT#:~:text=ChatGPT%20is%20based%20on%20particular,from%20human%20feedback%20\(RLHF\)](https://en.wikipedia.org/wiki/ChatGPT#:~:text=ChatGPT%20is%20based%20on%20particular,from%20human%20feedback%20(RLHF))
- [3] <https://www.techtarget.com/searchenterpriseai/definition/Google-Bard>
- [4] <https://github.com/langchain-ai/langchain/issues/15484>
- [5] <https://quickaitutorial.com/microsoft-phi-2-huggine-face-langchain-super-tiny-chat-bot/>
- [6] <https://pypi.org/project/ngrok/#:~:text=ngrok-python%20is%20the%20official,service%20running%20in%20any%20environment>
- [7] <https://huggingface.co/microsoft/phi-2>