# Module 4: Finding Existing Models

Abdullah Arif (LinkedIn)

# Module 4 Contents

1. Why should I learn AI
2. Basic understanding
3. How can I interact with these models
   a. Chat interfaces
   b. LLM APIs
   c. Finding models
4. Where can I find data?
5. Prompt Engineering
6. Tips on choosing a model

# Why should I spend my time learning AI

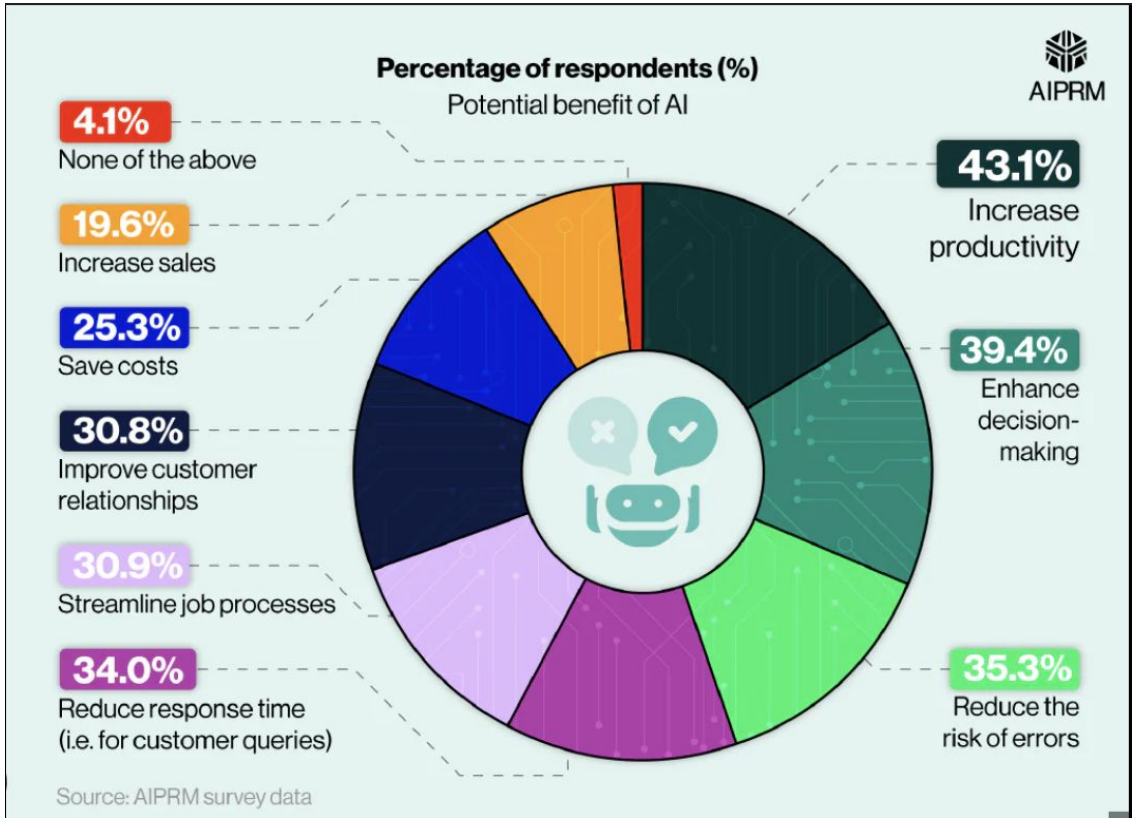
AI will have an estimated 21% net increase on the United States GDP by 2030

Research estimates AI will create 97 million jobs

97% of business owners believe ChatGPT will help their business

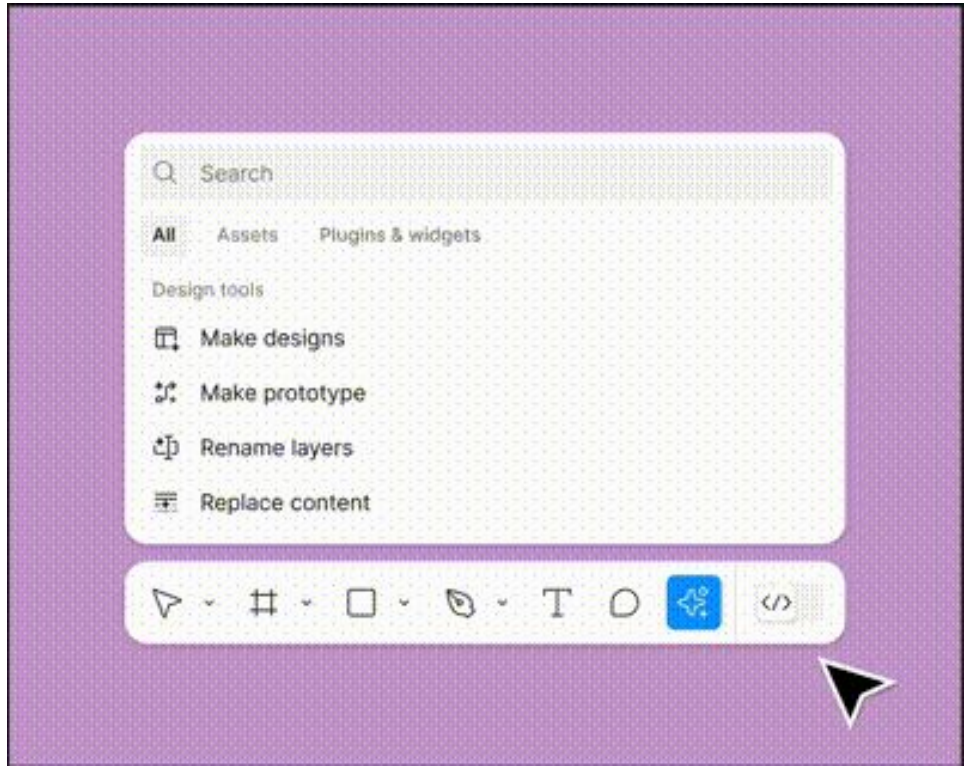
A 2023 report by McKinsey found that up to 30% of tasks in 60% of occupations could be automated.

**Moral of the story:** AI is here and spending your time mastering the use of AI will help you unlock the benefits it brings and be a meaningful part of this revolution.



# What can I do with AI?

- The opportunities are limitless
- Here are some things people have built:
  - A new search engine (Perplexity)
  - A musician (Suno.ai)
  - Targeted pesticide spraying (Blue River Technologies)
  - AI UI/UX designer (Figma's AI tools)
  - Custom Emojis (part of the upcoming Apple Intelligence)
  - Writing Assistant (Grammarly AI)
  - A workout/meal planner (Fitness AI)
  - Employee Shift scheduler (7shifts)
  - Meeting Notes and Summarizer (Parrot AI)
  - AI Exam Generator (Quizgecko)
  - AI Job Interviewer (talently.ai)
  - AI fashion models (botika.io)
  - Vacation Itinerary Planning (Kayak's AI assistant)
  - Homework :)
  - I also use Chat-GPT to play games with me when I'm bored



# Understanding Machine Learning Models and Large Language Models (LLMs)

**Machine Learning Models:**

- Algorithms that learn patterns from data to make predictions or decisions.
- Types:
  - Supervised Learning: Learns from labeled data (e.g., predicting house prices).
  - Unsupervised Learning: Finds patterns in unlabeled data (e.g., clustering customers).
  - Reinforcement Learning: Learns by trial and error (e.g., game playing AI).
  - Deep Learning: Identifies patterns in data using neural networks

Large Language Models (LLMs):

- A type of deep learning model designed to understand and generate human language.
- Examples: GPT-4, BERT.
- Capabilities:
  - Text Generation: Writing essays, stories, code.
  - Language Translation: Translating between languages.
  - Chatbots: Engaging in conversations with users.
- Training: Trained on vast amounts of text data to understand context, grammar, and nuances of language.

# How can I interact with these models?

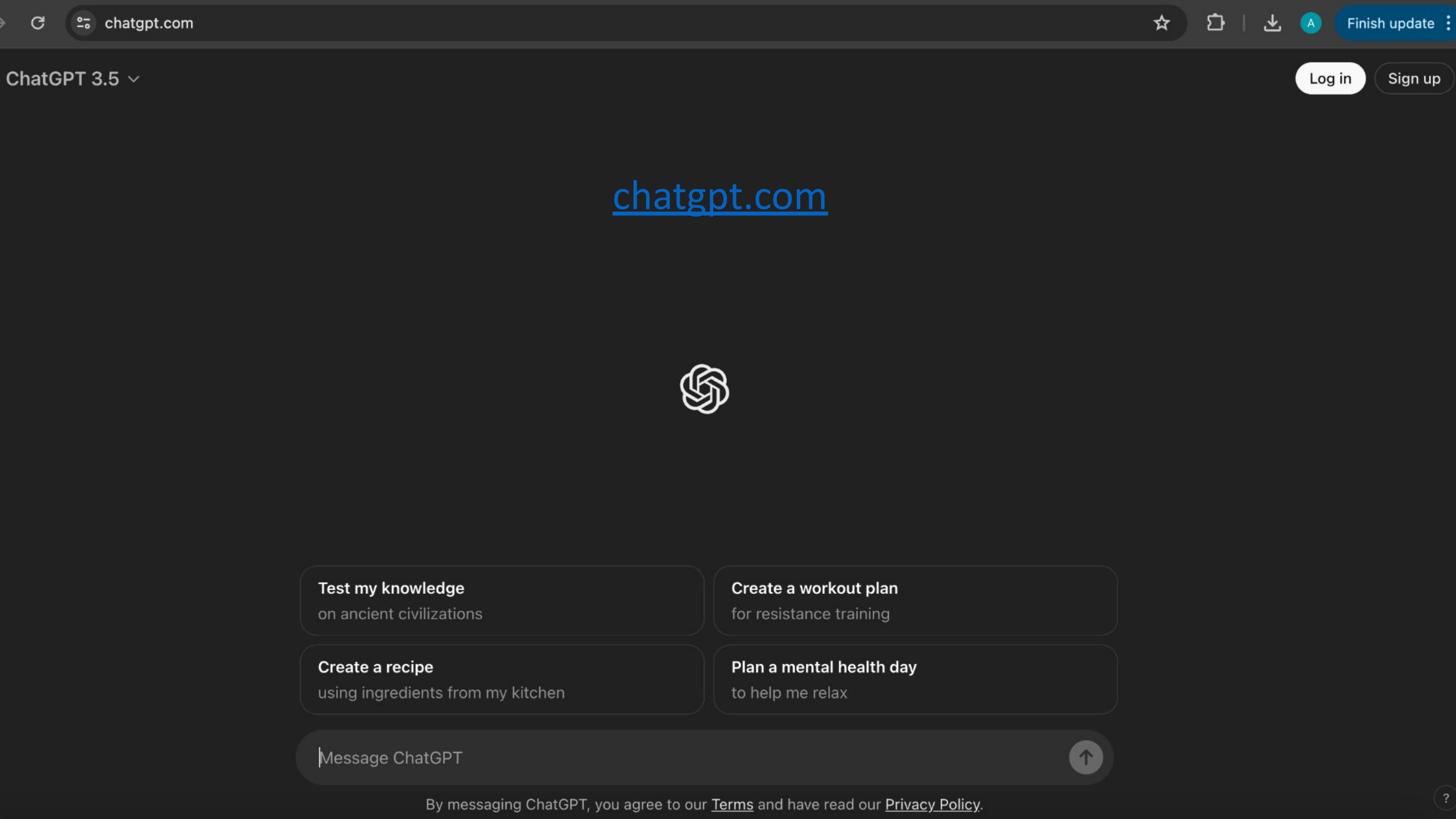
- Chat Interfaces
  - ChatGPT
  - Google Gemini
  - Claude

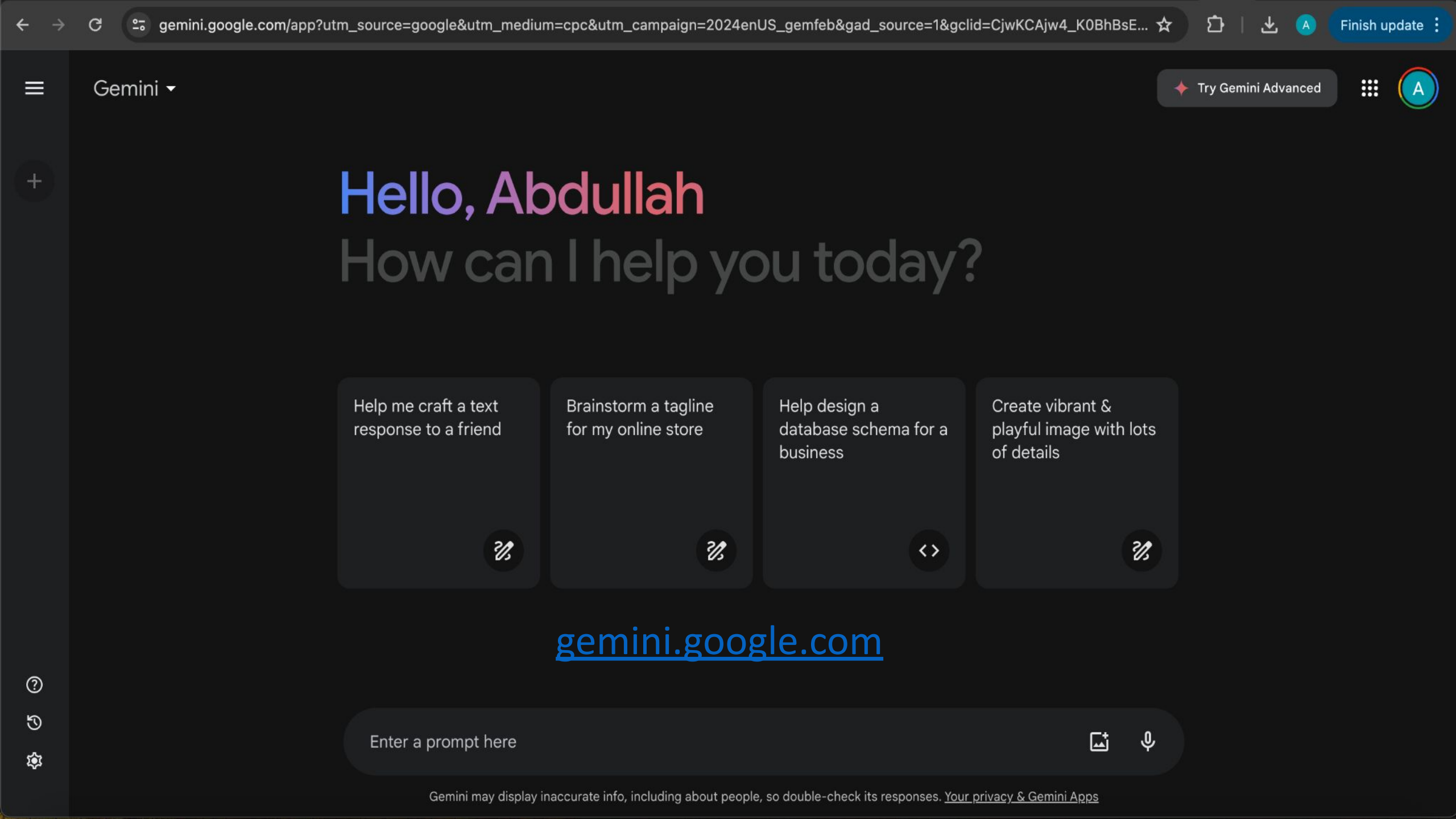
- LLM APIs
  - GPT (3.5, 4, 4 turbo, 4-o, 4-o turbo)
  - Gemini
  - Mistral
  - Whisper

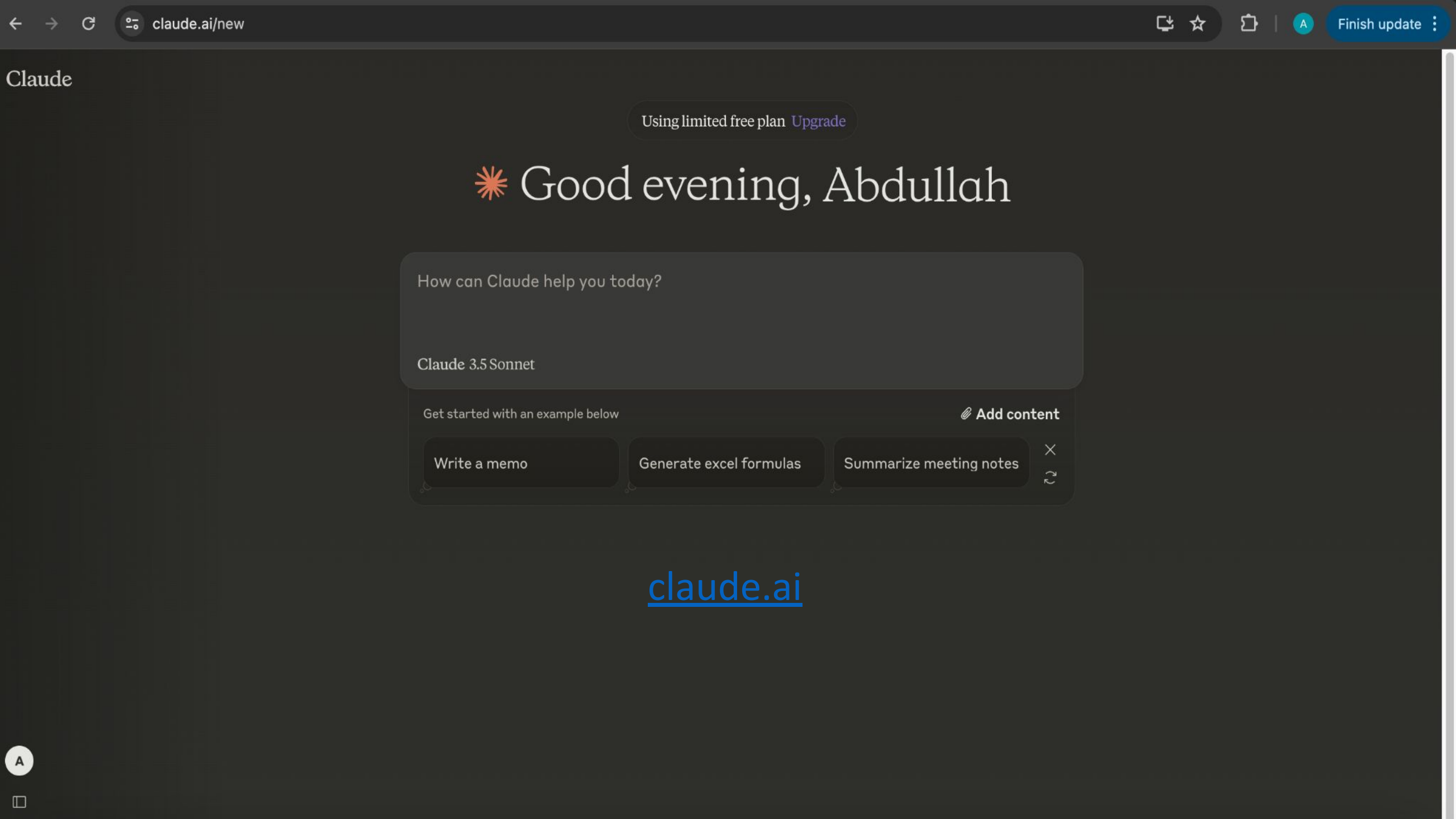
- Specialized models
  - YOLO
  - Google T5-base
  - Whisper

# Chat Interfaces

- Simplest way to interact with LLMs (Large Language Models)
- These interfaces exist for most of the big LLMs available today e.g.
  - ChatGPT
  - Gemini
  - Claude
  - Groq
- Most of these services have a free plan that limits usage (certain number of prompts per day) or limits the models available (ChatGPT 3.5 vs. 4)
- These chat interfaces are a great way to play around with LLMs, work on prompting skills and getting everyday tasks done like writing emails, documentation, study questions, etc.

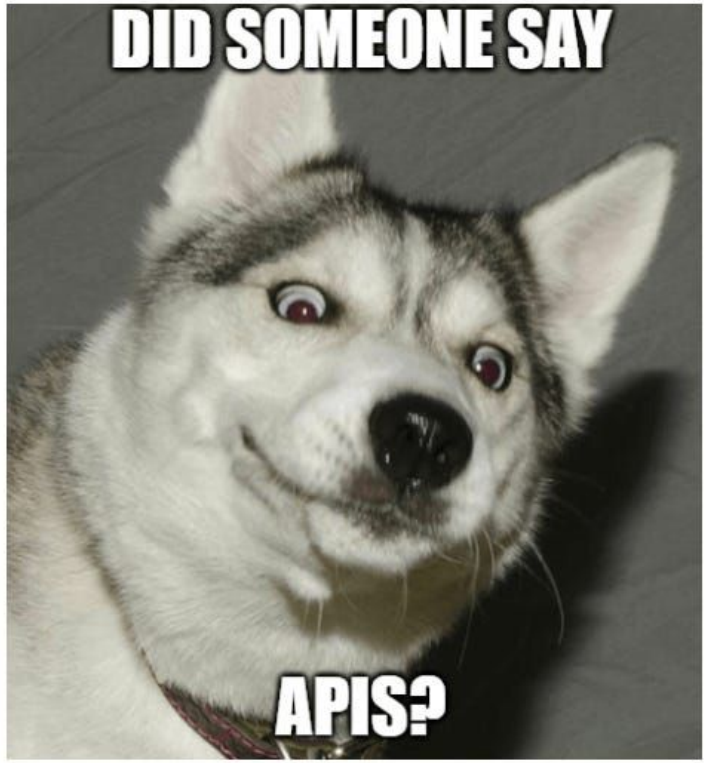

chatgpt.com
Finish update
ChatGPT 3.5
Log in
Sign up
chatgpt.com
Test my knowledge
on ancient civilizations
Create a workout plan
for resistance training
Create a recipe
using ingredients from my kitchen
Plan a mental health day
to help me relax
Message ChatGPT
By messaging ChatGPT, you agree to our Terms and have read our Privacy Policy.


gemini.google.com/app?utm_source=google&utm_medium=cpc&utm_campaign=2024enUS_gemfeb&gad_source=1&gclid=CjwKCAjw4_K0BhBsE...
Finish update
Gemini
Try Gemini Advanced
Hello, Abdullah
How can I help you today?
Help me craft a text response to a friend
Brainstorm a tagline for my online store
Help design a database schema for a business
Create vibrant & playful image with lots of details
gemini.google.com
Enter a prompt here
Gemini may display inaccurate info, including about people, so double-check its responses. Your privacy & Gemini Apps


claude.ai/new
Finish update
Claude
Using limited free plan Upgrade
Good evening, Abdullah
How can Claude help you today?
Claude 3.5 Sonnet
Get started with an example below
Add content
Write a memo
Generate excel formulas
Summarize meeting notes
claude.ai

# What is an API?

- APIs, or Application Programming Interfaces, are like waiters at a restaurant. They take requests from you (the customer), tell the kitchen (the server) what you want, and then bring the response back to you.
- Imagine a Restaurant:
  - You (the customer): Want to order food (ask for data or action).
  - Waiter (API): Takes your order to the kitchen (server).
  - Kitchen (Server): Prepares the food (processes the request).
  - Waiter (API): Brings the food back to you (returns the data).
- Simple Breakdown:
  - You make a request: "What's the weather in New York?"
  - API delivers the request: Takes it to the server.
  - Server processes: Finds the weather data.
  - API brings back: "The weather is sunny!"



# Accessing LLMs through APIs

- Most of the LLMs from the big tech companies have APIs available to interact with their LLMS
- Pros:
  - Easy to setup and use
  - Faster prototyping
  - You do not need expensive equipment like GPUs to run these models locally, everything runs on their servers
  - Access to the most powerful LLMs available (GPT 4-o, Gemini Ultra, etc.)
- Cons:
  - The API usage usually costs money
  - The application can be slower because the data has to go to the companies server and come back.
  - Any application using LLM APIs must be connected to the internet when being used and the speed of the internet connection can impact performance.
- Links
  - Open AI API documentation
  - Google Gemini API Studio
  - Claude by Anthropic Console

# Accessing LLMs through APIs

```
# Import prerequisite libraries
import os
import openai

# Setting the API key
openai.api_key = os.getenv("OPENAI_API_KEY")
# Define the user prompt message
prompt = "Hello!"
# Create a chatbot using ChatCompletion.create() function
completion = openai.ChatCompletion.create(
  # Use GPT 3.5 as the LLM
  model="gpt-3.5-turbo",
  # Pre-define conversation messages for the possible roles
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": prompt}
  ]
)
# Print the returned output from the LLM model
print(completion.choices[0].message)
```

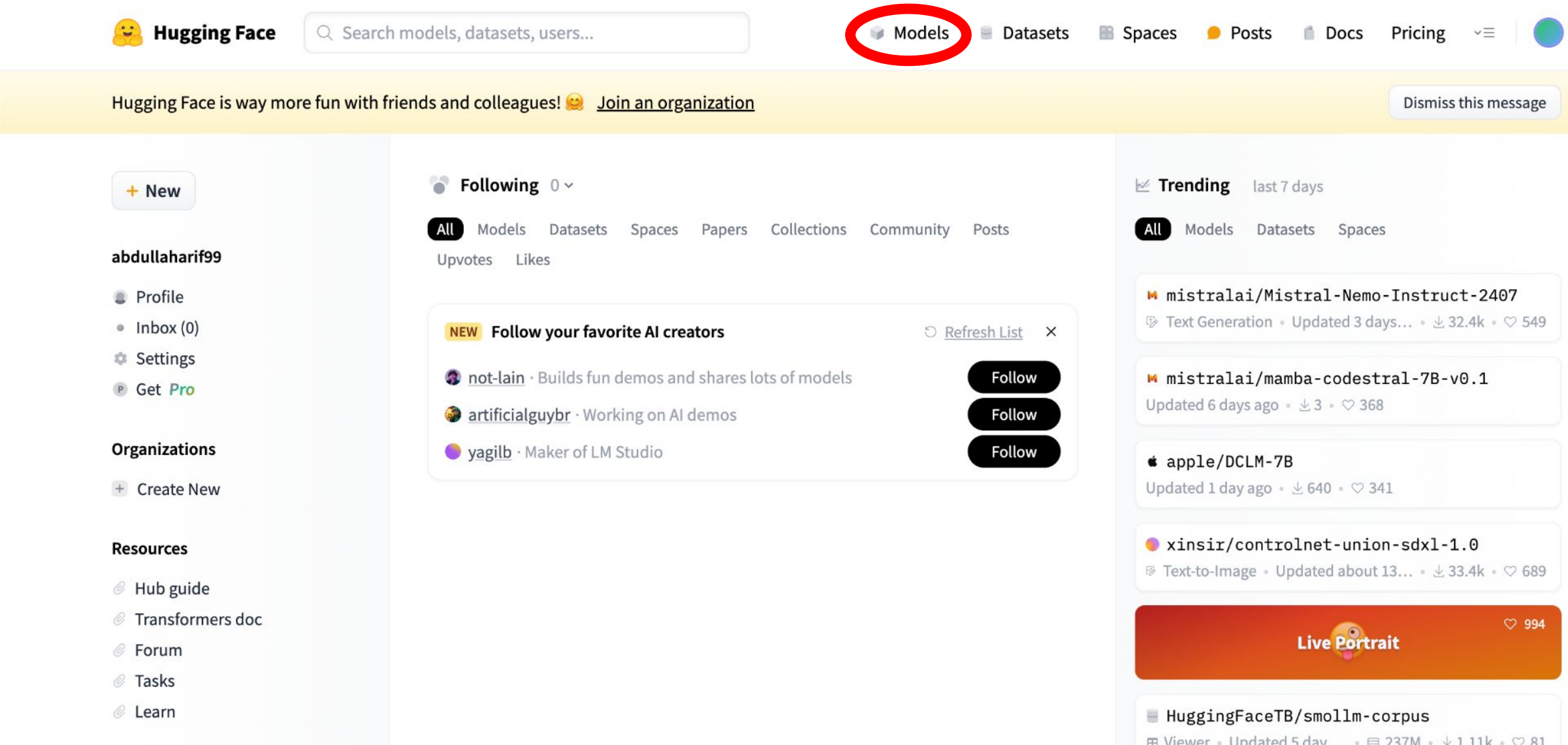
# What are pre-trained specialized models?

- These are smaller models that are trained and fine-tuned to perform specific tasks. For example:
  - whisper - speech to text
  - Google T5 base - text-to-text translation
  - YOLO - Object detection
  - Depth-Anything-V2-Small - to measure depth in pictures
  - faces_age_detection - to classify pictures of faces into age group (Young, Middle, old)
- These specialized models are far more accurate at accomplishing specific tasks
- These models can be further fine tuned using your own data to increase their accuracy

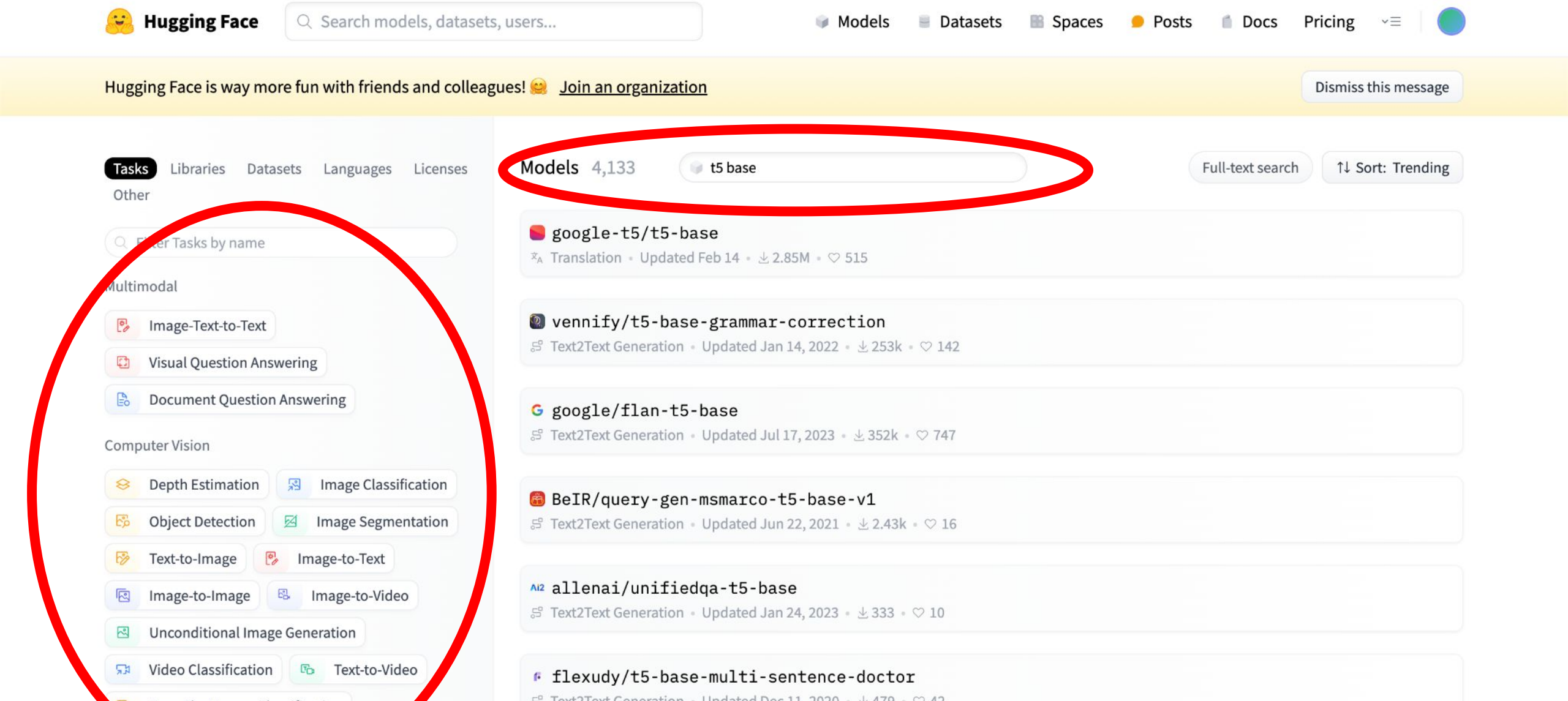
# Where can I find these specialized models?

- Many of these specialized models are open sourced which means that they are freely available on the internet for anyone to download, use and modify.
- There are specific platforms that are built to upload pre trained models with instructions on how to use them. For example:
  - Hugging Face
  - Kaggle
  - TensorFlow Hub
  - Model Zoo
- Hugging Face is the most popular source for pre-trained models and has easy to use instructions for most models
- Most of these websites also have communities and forums for each model so you can ask questions in there as well.

# Hugging Face Example



# Hugging Face Example



# Hugging Face Example

google-t5 / **t5-base** ♡ like 515

Translation | Transformers | PyTorch | TensorFlow | JAX | Rust | Safetensors | c4 | 4 languages | t5 | text2text-generation | summarization | text-generation-inference | Inference Endpoints | 8 papers | License: apache-2.0

**Model card** | Files and versions | Community 27 | Train | Deploy | Use this model

Edit model card

## Model Card for T5 Base

model image

### Table of Contents

1. Model Details
2. Uses
3. Bias, Risks, and Limitations
4. Training Details
5. Evaluation
6. Environmental Impact

Downloads last month
2,847,630

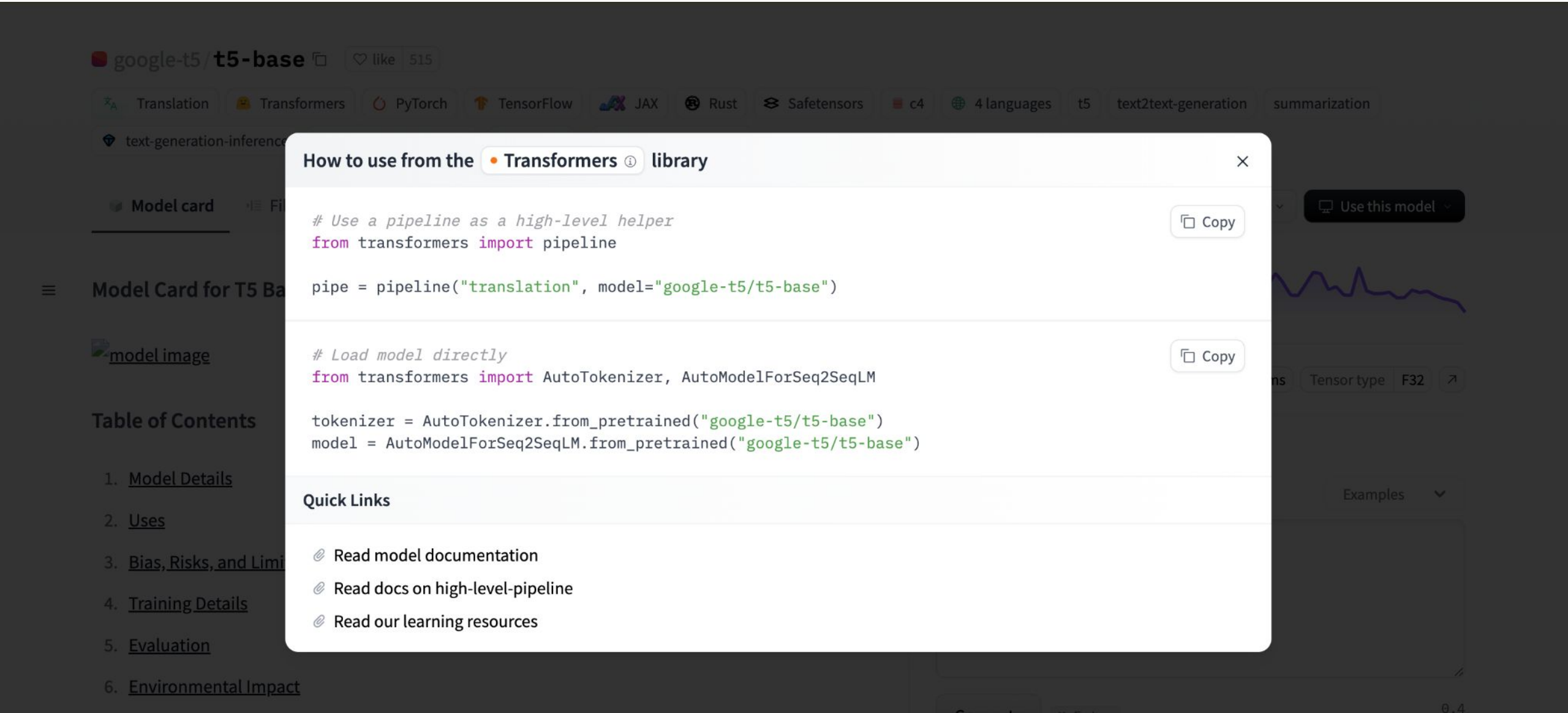
**Safetensors** Model size 223M params | Tensor type F32

**Inference API**

Translation | Examples

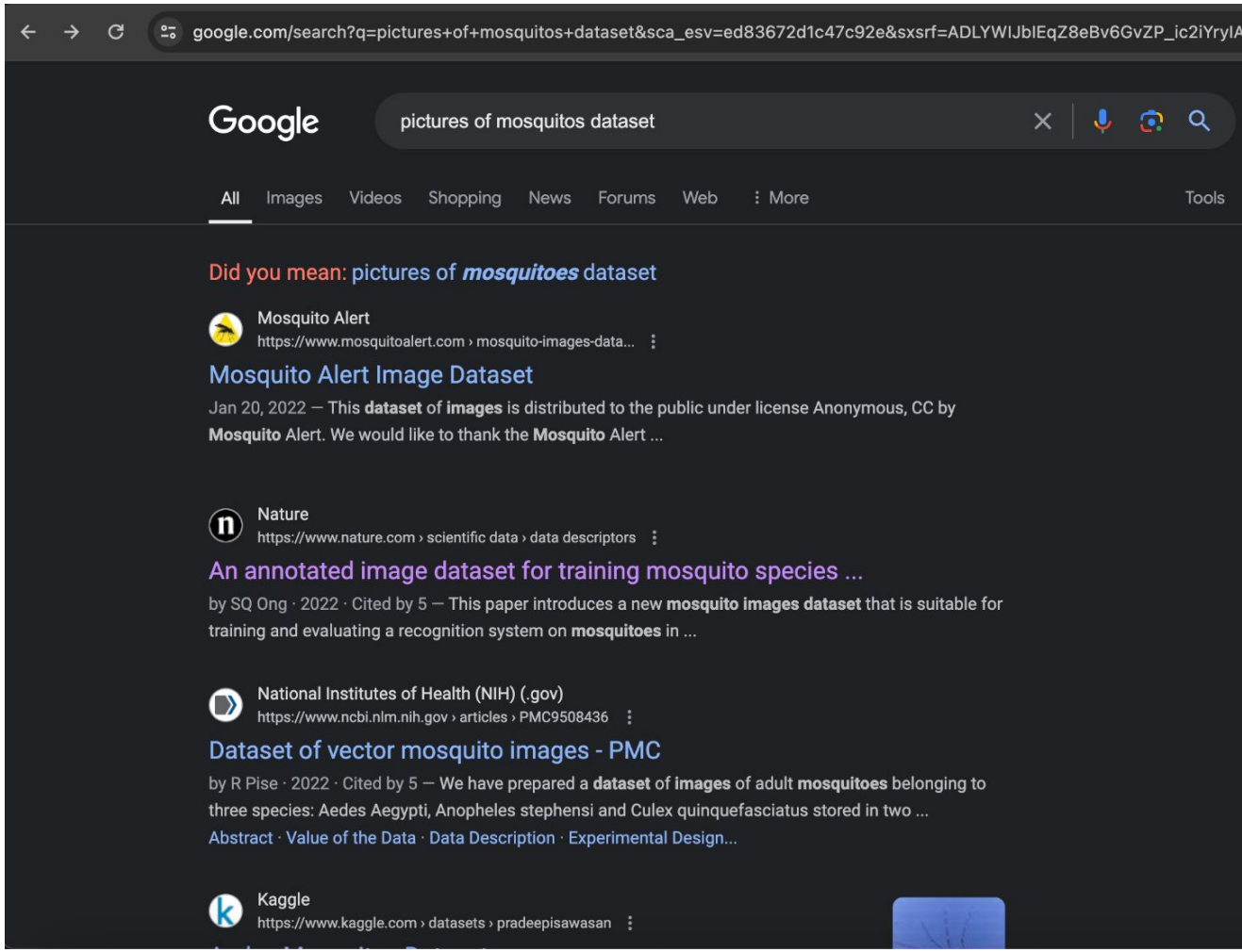
My name is Sarah and I live in London

# Hugging Face Example



# Datasets…it's all about data

- Data is necessary for any machine learning/AI task
- **Where do I get data?**
  - Internet
- Popular sources:
  - kaggle.com
  - huggingface.co
  - https://github.com/awesomedata/awesome-public-datasets
- Github can be a great source of datasets
- Just google it
  - Someone probably has a dataset similar to what you need somewhere on the internet




YOU GET DATA! AND YOU GET DATA!
AND YOU GET DATA!
EVERYBODY GETS DATA!

# Prompt Engineering

**What is Prompt Engineering?**

- The process of designing and refining input prompts to get the desired output from AI models, especially language models like GPT-4.
- Purpose: To improve the accuracy, relevance, and quality of the responses generated by the model.

**Why is it Important?**

- Directs AI Behavior: Helps guide the AI to provide more accurate and useful responses.
- Maximizes Model Potential: Ensures the AI model is used effectively for various tasks.

**Example**

- Basic Prompt: "Tell me about AI."
- Improved Prompt: "Explain the benefits of AI in healthcare, focusing on diagnostics and patient care."

# Prompt Engineering

Tips for Effective Prompt Engineering:

- Be Clear and Specific:
  - State your request clearly.
  - Provide context if needed.
- Use Examples:
  - Show the model examples of the kind of output you expect.
- Iterate and Refine:
  - Test different prompts and refine them based on the results.
- Set Constraints:
  - Limit the scope if necessary (e.g., "List three benefits of AI").
- Ask Direct Questions:
  - Frame prompts as direct questions to get focused answers.
- Use Step-by-Step Instructions:
  - Break down complex requests into simpler steps.

**Great Resource**: https://www.promptingguide.ai

# Prompt Engineering Example

**Act as a Prompt Generator**

I want you to act as a prompt generator. Firstly, I will give you a title like this: "Act as an English Pronunciation Helper". Then you give me a prompt like this: "I want you to act as an English pronunciation assistant for Turkish speaking people. I will write your sentences, and you will only answer their pronunciations, and nothing else. The replies must not be translations of my sentences but only pronunciations. Pronunciations should use Turkish Latin letters for phonetics. Do not write explanations on replies. My first sentence is "how the weather is in Istanbul?"." (You should adapt the sample prompt according to the title I gave. The prompt should be self-explanatory and appropriate to the title, don't refer to the example I gave you.). My first title is "Act as a Code Review Helper" (Give me prompt only)

Resources to check out:

- https://github.com/f/awesome-chatgpt-prompts
- https://github.com/adamkdean/prompts

# Tips for choosing a model

1. Define the Problem:

- What are you trying to solve or achieve? (e.g., image recognition, text analysis, prediction)

2. Identify Data Type:

- What type of data do you have? (e.g., text, images, numbers)

3. Determine Model Requirements:

- What accuracy do you need?
- How fast should the model be?
- Are there any resource constraints? (e.g., computational power, memory)

4. Research Available Models:

- Look for models that are commonly used for your problem type and data.
- Check for pre-trained models (e.g., TensorFlow, PyTorch libraries).

# Tips for choosing a model

5. Evaluate Model Performance:

- Test different models with your data.
- Compare performance metrics (e.g., accuracy, speed).

6. Consider Complexity:

- Simpler models are easier to implement and understand.
- Complex models might perform better but are harder to manage.

7. Check for Support and Community:

- Models with strong community support and documentation are easier to troubleshoot and improve.

8. Make a Decision:

- Choose the model that best meets your needs based on performance, complexity, and support.

# Questions