



# DATA 8005: Advanced Natural Language Processing

Fall 2023

# Logistics

- Location: Learning Commons CPD 2.25
- Meetings: Friday 1:30 - 4:20 pm
- Instructor: Tao Yu (<https://taoyds.github.io/>)
- Office hours: Wednesday 4 - 5 pm @IDS

# Logistics

Course website: <https://taoyds.github.io/courses/data8005>

- We will maintain the website for schedule, lecture slides, reading lists, grading policies, etc
- Only submit your reports on Moodle.

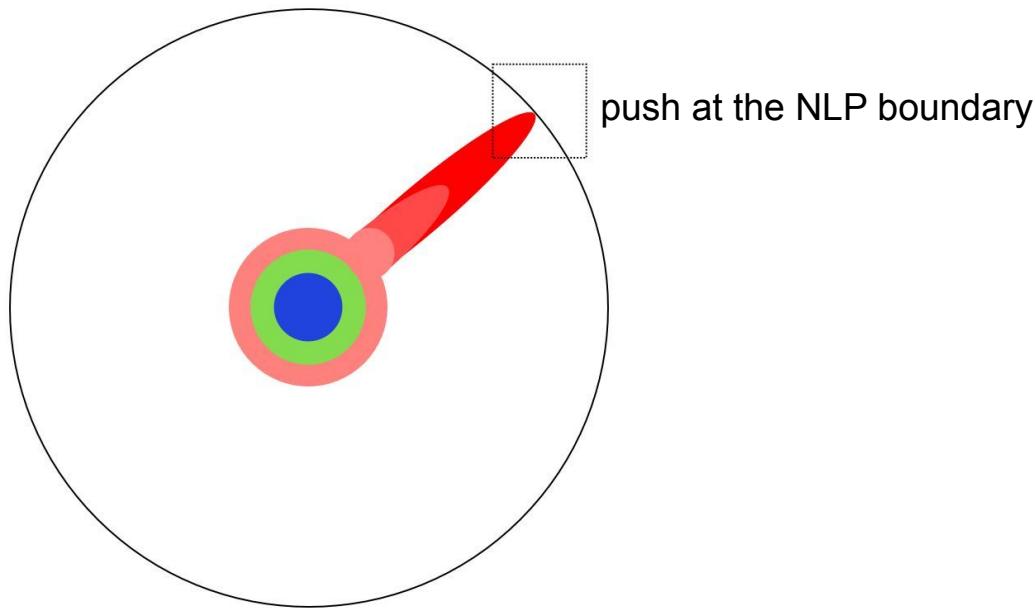
# Logistics

Slack: [https://join.slack.com/t/slack-bdp9993/shared\\_invite/zt-23739aa6d-uW\\_jp~Yh\\_CGbEma1fH007w](https://join.slack.com/t/slack-bdp9993/shared_invite/zt-23739aa6d-uW_jp~Yh_CGbEma1fH007w)

- We will use Slack as the primary mode of communication. DM me on Slack instead of emails.
  - Provide feedback on your lectures
  - Answer lecture-related, project questions
  - More importantly, encourage you ask questions, share random thoughts, highlight interesting papers, brag about cool finding there.
- Join Slack via the invitation link above.



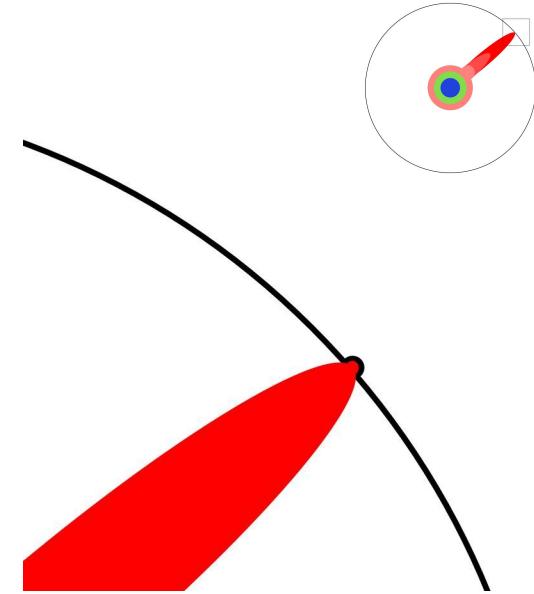
# Course structure



**This is an advanced graduate course**

## Course structure

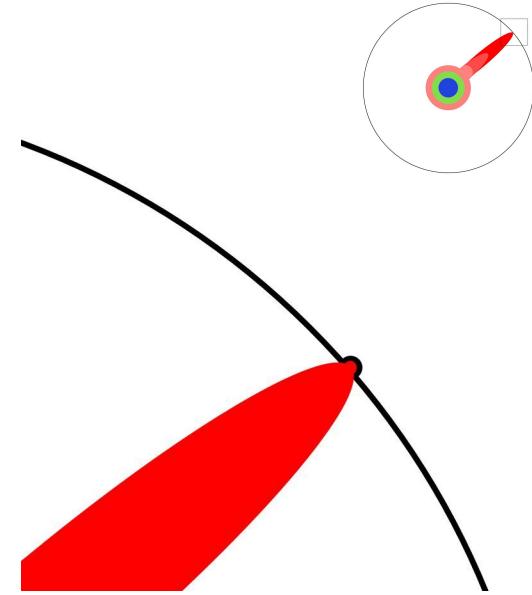
- We will be teaching and discussing state-of-the-art papers about large language models.
- Mostly presentations and discussions.
- You are expected to come to the class regularly and participate in discussion.



**This is an advanced graduate course**

## Course prerequisites

- Familiarity with deep learning and machine learning
- Familiarity with Python programming
- Familiarity with basic NLP tasks
- NLP course background is recommended.



**This is an advanced graduate course**

# Course goals

- Learn more about current state-of-the-art NLP research
- Practice oral and written NLP research presentation skills
- Practice different parts of NLP research cycle including understanding prior literature, brainstorming ideas, choosing a research problem, executing a project.



# Components and grading

- **In-class presentation: 30%**
  - Survey and review papers/blogs on a topic and present it in the class.
- **Class participation: 25%**
  - Read papers before the class and comments on Slack (answer, ask, or add >2 high-quality questions/comments, suggest related papers)
  - In-class group discussions or idea pitches
  - Feedback on presentations from your classmates
- **Final project: 45%**

## In-class presentation

- **Two** students work together and deliver a 60-minute presentation
- Your goal is to **educate others** in the class about the topic
  - Cover at least the required paper(s)
  - Search for latest related papers (citations)
  - Survey more related work/blogs to add background and context
- Presentation meeting
  - Send me your draft slides on Slack **before** my office hours
  - Use Google slides and add slide numbers
  - Provide comments and give suggestions on Slack
- Everyone else: prepare and ask questions!

# What is a good presentation?

- Add background and context and survey more related work/blogs
  - What this topic is about
  - Why this topic is important
  - The most important problems and recent methods on this topic
- Paper presentation
  - Most papers were published in the past 1-2 years (if not a few months)
  - Highlight the biggest takeaways
  - Why this paper is important, what problem it trying to solve
  - Pay attention to technical details, but don't too much
  - Future work?
- More or less like a short tutorial on this topic

# How to survey a new NLP research topic?

- Related papers
  - Prior work: Related Work section
  - Latest work: papers that cite the assigned papers
- Related recent tutorials at NLP conferences (in 1-2 years)
- Good survey papers/blogs
- Many NLPers use Twitter
- We can select these papers together
- Other related courses



# How to survey a new NLP research topic?

- Other related courses
  - Regular NLP courses
    - [CS224N: Natural Language Processing with Deep Learning](#)
    - [CS388: Natural Language Processing](#)
    - [Advanced NLP](#)
  - Seminar-based advanced NLP courses
    - [COS 597G: Understanding Large Language Models](#)
    - [COS 597F: Embodied Language Understanding](#)
    - [CS25: Transformers United V2](#)
    - [CSE 599: Exploration on Language, Knowledge, and Reasoning](#)
    - [Understanding the World Through Code](#)
    - [CS324: Large Language Models, winter 2023](#)

## Before class

- Read the assigned papers before the class
  - **Very likely** you will need to read up on some prior literature to fully understand the paper(s) we discuss!
- Pre-lecture questions
  - Send your comments on **# week#** Slack channel before the class (answer, ask, or add >2 high-quality questions/comments about one topic each week)
- Come prepared with points of discussion and ideas to contribute in class

## In-class group discussion

- Ask questions or suggest paper improvements
- Answer others' questions
- Add more related work or propose new ideas
- Review and discuss pre-lecture questions
- Also can provide feedback on the presentation: clarity, structure, completeness, slides

## Lecture format

- Two lectures per week (2 x ~8 = 16 lectures)
  - 60-minute per lecture
    - Be prepared for lots of questions (and we encourage questions in the class)
    - Please control your time (rehearsal is very helpful)!
  - 15-minute post-lecture group discussion
    - Divide the class into groups of 4-5 students (depending on seating and enrollment)
- Potential guest lectures

## Final project

- Students complete a research project in teams of 2-3
- More guidelines will be announced soon
- Deadlines
  - Group registration: week 5, Sep 29
  - Project proposal: week 7,
  - Final report paper: 11:59 pm on Dec 15
- In-class presentation: ToD

# Introduce yourself!



# Topics: Large language models (LLMs)

- What are LLMs?
  - Pretraining: from BERT (encoders) to GPT4 (decoders)
- How to use and adapt LLMs?
  - Prompting, in-context learning
  - Instruction tuning
  - Knowledge and reasoning
  - Alignment/RLHF
- How to evaluate and analyze LLMs
  - Data and benchmarking
  - Scaling law
  - Robustness, interpretability, explainability

# Large language models (tentative)

- Other models:
  - Sparse/retrieval- based
  - Code LLMs
  - Multimodal LLMs
- Social impacts
  - Security and privacy
  - Efficiency
  - Ethics

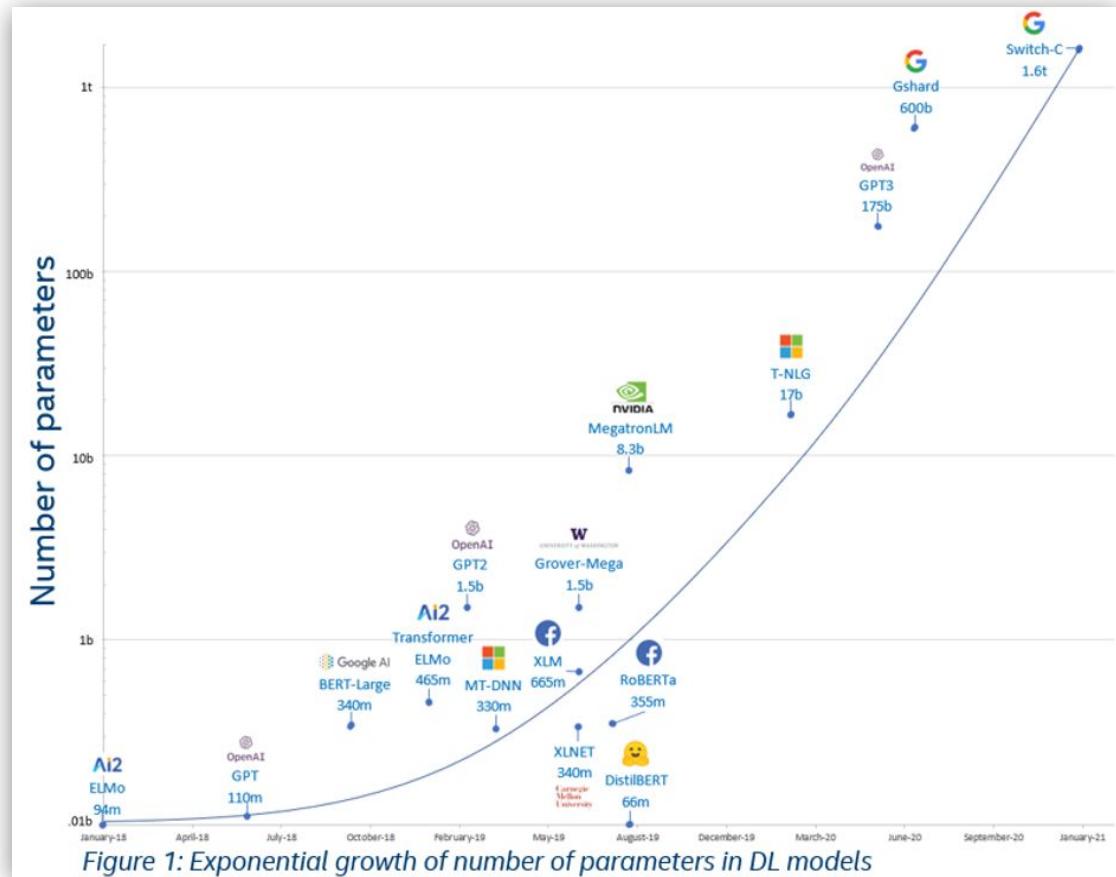
## LM agents

- Language and interaction
- Language grounding
- Reasoning and planning
- Tool use
- Multi/human-agent communication
- Personalization, memory, skill learning
- Robotics and embodied interaction
- .....

# Large language models

# Scaling language models

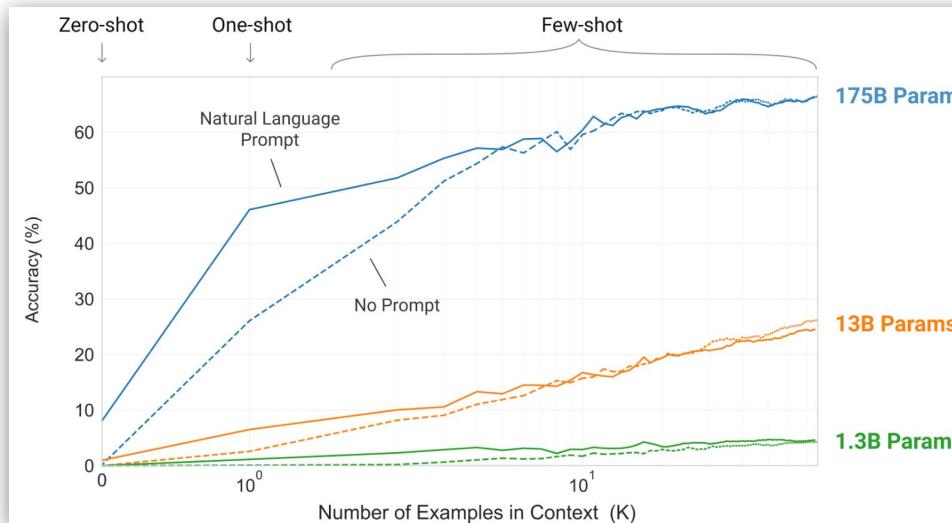
- Language models are getting bigger and bigger...
- GPT-3 is almost **500 times** the size of the largest BERT model.



# Scaling laws in performance

## Scaling GPT models shows clear scaling laws in performance

- We still don't know how far we can go in scaling...
  - We don't know how large GPT4 is.
  - Challenge: efficient distributional training on thousands of GPUs/TPUs...



# A single language model for all NLP applications

Showing good or even SOTA performance on most of the tasks with a single model checkpoint!

The screenshot shows the 'Examples' section of the OpenAI website. At the top, there are navigation links: 'Overview', 'Documentation', and 'Examples'. Below the navigation is a search bar with placeholder text 'Search...' and a dropdown menu labeled 'All categories'. The main area is titled 'Examples' and contains the sub-instruction 'Explore what's possible with some example applications'. There are two columns of cards, each representing a different NLP task. The first column includes: Q&A, Grammar correction, Summarize for a 2nd grader, Natural language to OpenAI API, Text to command, English to other languages, Natural language to Stripe API, SQL translate, Parse unstructured data, Classification, Python to natural language, Movie to Emoji, Calculate Time Complexity, Translate programming languages, Advanced tweet classifier, Explain code, and Keywords. The second column includes: TL;DR summarization, Python bug fixer, Spreadsheet creator, JavaScript helper chatbot, ML/AI language model tutor, Science fiction book list maker, Tweet classifier, Airport code extractor, #SQL request, Extract contact information, JavaScript to Python, Friend chat, Mood to color, Write a Python docstring, Analogy maker, JavaScript one line function, Micro horror story creator, Third-person converter, Notes to summary, VR fitness idea generator, ESRB rating, and Essay outline.

# Q&A example with GPT-3

 Q&A  
Answers Generation Conversation

Answer questions based on existing knowledge.

**Prompt**

I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".

Q: What is human life expectancy in the United States?  
A: Human life expectancy in the United States is 78 years.

Q: Who was president of the United States in 1955?  
A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?  
A: He belonged to the Republican Party.

Q: What is the square root of banana?  
A: Unknown

Q: How does a telescope work?  
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?  
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?  
A: Unknown

Q: Where is the Valley of Kings?  
A:

## Sample response

The Valley of Kings is located in Luxor, Egypt.

## API request

python ▾ ⌂ Copy

```
1 import os
2 import openai
3
4 openai.api_key = os.getenv("OPENAI_API_KEY")
5
6 response = openai.Completion.create(
7     engine="text-davinci-002",
8     prompt="I am a highly intelligent question answering bot",
9     temperature=0,
10    max_tokens=100,
11    top_p=1,
12    frequency_penalty=0.0,
13    presence_penalty=0.0,
14    stop=["\n"]
15 )
```

# More examples with GPT-3

 **TL;DR summarization**

Transformation Generation

Summarize text by adding a 'tl;dr:' to the end of a text passage. It shows that the API understands how to perform a number of tasks with no instructions.

**Prompt**

A neutron star is the collapsed core of a massive supergiant star, which had a total mass of between 10 and 25 solar masses, possibly more if the star was especially metal-rich.<sup>[1]</sup> Neutron stars are the smallest and densest stellar objects, excluding black holes and hypothetical white holes, quark stars, and strange stars.<sup>[2]</sup> Neutron stars have a radius on the order of 10 kilometres (6.2 mi) and a mass of about 1.4 solar masses.<sup>[3]</sup> They result from the supernova explosion of a massive star, combined with gravitational collapse, that compresses the core past white dwarf star density to that of atomic nuclei.

TL;dr

**Sample response**

A neutron star is the collapsed core of a massive supergiant star. These ultra-dense objects are incredibly fascinating due to their strange properties and their potential for phenomena such as extreme gravitational forces and a strong magnetic field.

 **Interview questions**

Generation

Create interview questions.

**Prompt**

Create a list of 8 questions for my interview with a science fiction author:

**Sample response**

1. What inspired you to write science fiction?
2. What themes or topics do you explore in your stories?
3. What are some of the unique elements that make science fiction stand out from other genres?
4. What challenges do you face when writing science fiction?
5. What are some of the most important aspects of a good science fiction story?
6. What do you think are the biggest challenges facing science fiction today?
7. What do you think the future of science fiction holds for readers and writers?
8. What advice would you give to aspiring science fiction authors?

# Coding Examples

 **SQL request**

Transformation Generation Translation

Create simple SQL queries.

**Prompt**

Create a SQL request to find all users who live in California and have over 1000 credits:

**Sample response**

```
SELECT * FROM users WHERE state='CA' AND credits > 1000;
```

 **Python bug fixer**

Code Generation

There's a number of ways of structuring the prompt for checking for bugs. Here we add a comment suggesting that source code is buggy, and then ask codex to generate a fixed code.

**Prompt**

```
##### Fix bugs in the below function

### Buggy Python
import Random
a = random.randint(1,12)
b = random.randint(1,12)
for i in range(10):
    question = "What is "+a+" x "+b+"? "
    answer = input(question)
    if answer == a*b
        print (Well done!)
    else:
        print("No.")

### Fixed Python
```

**Sample response**

```
import random
a = random.randint(1,12)
b = random.randint(1,12)
for i in range(10):
    question = "What is "+str(a)+" x "+str(b)+"? "
    answer = input(question)
    if answer == str(a*b):
        print ("Well done!")
    else:
```

# What is a language model?

- A probabilistic model that assigns a probability  $P(w_1, \dots, w_m)$  to every finite word sequence  $w_1, \dots, w_m$  (grammatical or not)

$$P(w_1, \dots, w_m) = \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1})$$

**Conditional probability**

$$\begin{aligned} p(w_1, w_2, w_3, \dots, w_N) &= \\ p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \times \dots \times p(w_N|w_1, w_2, \dots, w_{N-1}) & \end{aligned}$$

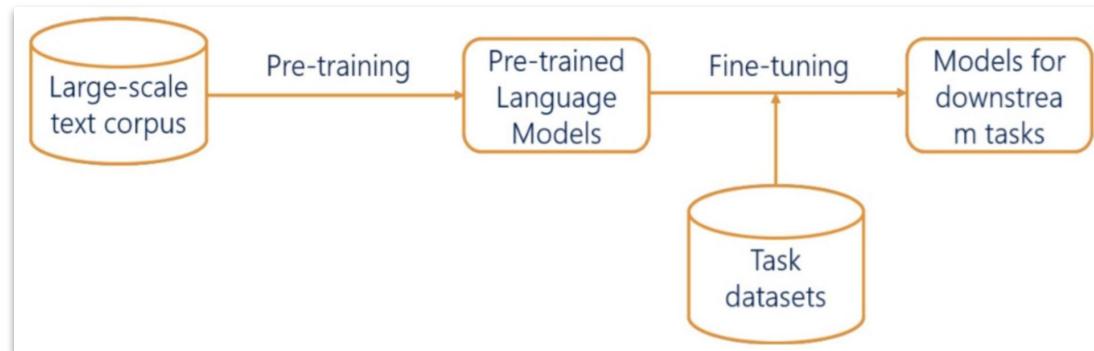
Sentence: “the cat sat on the mat”

$$\begin{aligned} P(\text{the cat sat on the mat}) &= P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ &\quad * P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ &\quad * P(\text{mat}|\underline{\text{the cat sat on}}) \end{aligned}$$

# Traditional learning paradigm

## Pretrain, Fine-tune

- The model is trained via repeated gradient updates using a large corpus of example tasks.

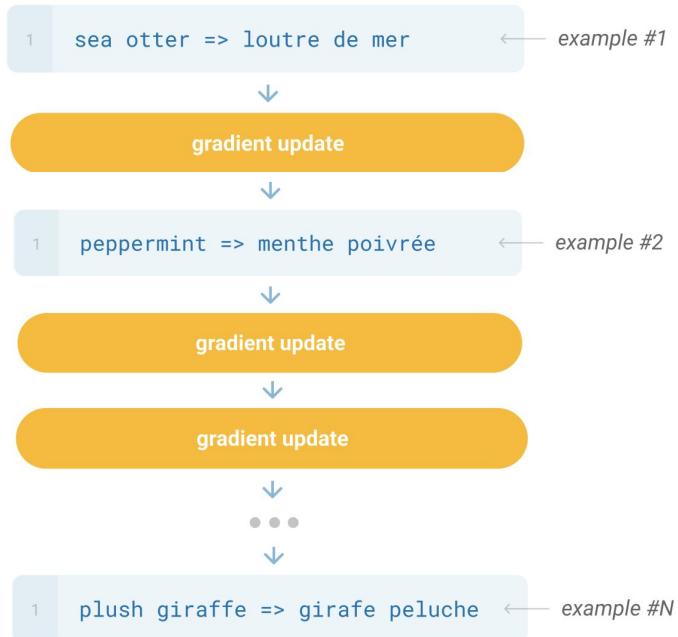


# Traditional learning paradigm

## Pretrain, Fine-tune

- The pretrained model (BERT, T5...) is trained via repeated gradient updates using a large corpus of example tasks.

## Fine-tune (updating parameters)

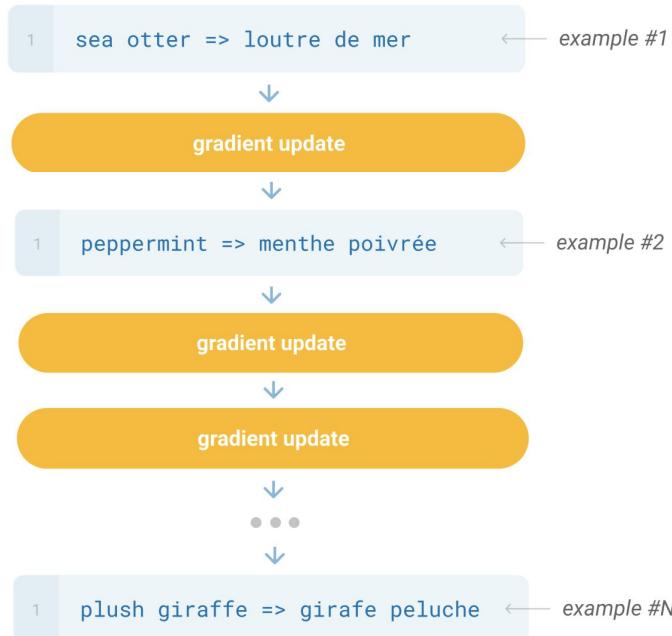


# Traditional learning paradigm

## Pretrain, Fine-tune, Predict

- The pretrained model (BERT, T5...) is trained via repeated gradient updates using a large corpus of example tasks.
- GPT-3/Codex doesn't update their parameters

## Fine-tune (updating parameters)



## Predict



# New learning paradigm: in-context learning

## Pretrain, Prompt

- **Zero-shot:** the model predicts the answer given **only** a natural language description of the task.
- You interact with the model **by simply providing a “prompt”.**
- **No fine-tuning**, GPT-3 doesn't update their parameters!

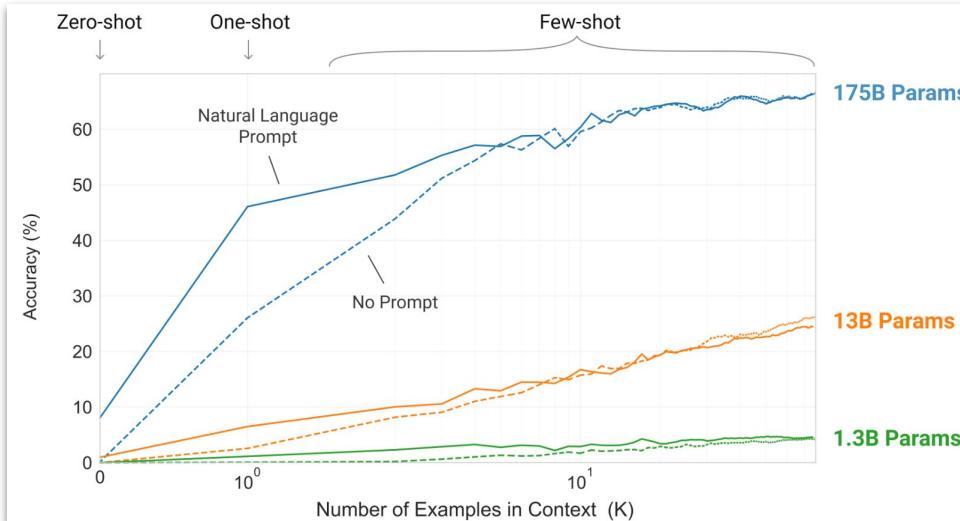
## Prompt (**no training**)



# How to make these models work?

## Prompt engineering is a key!

- Instead of changing model architectures and learning objectives in traditional NLP, designing good task prompts for these huge language models seems to be critical...

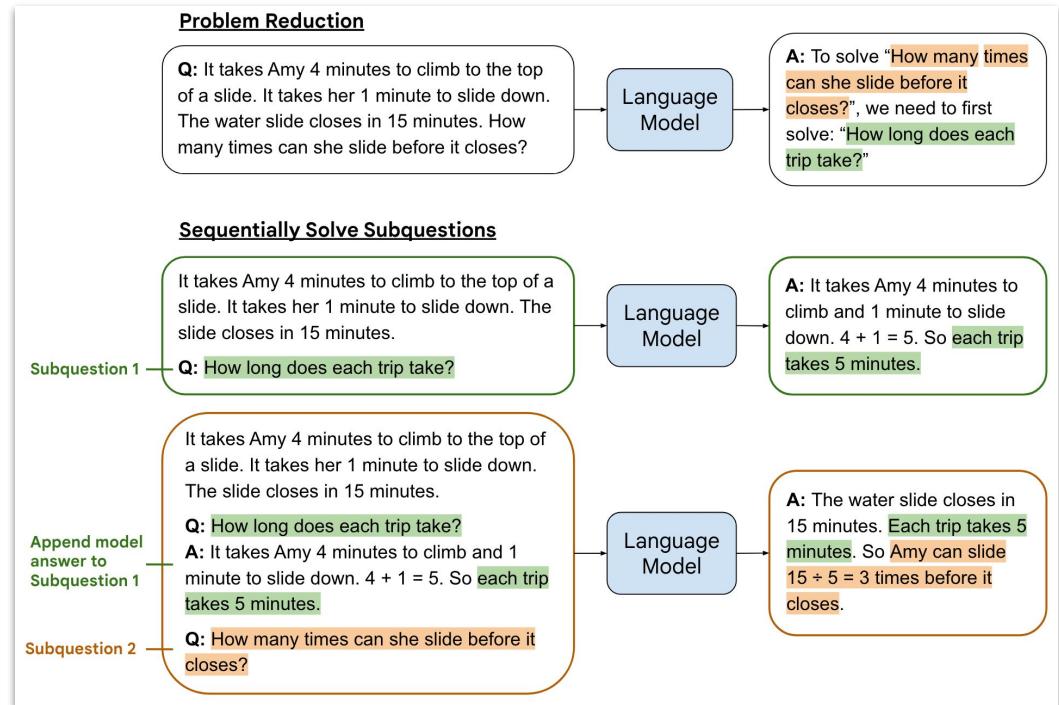


# Step-by-step prompting

## Least-to-most prompting

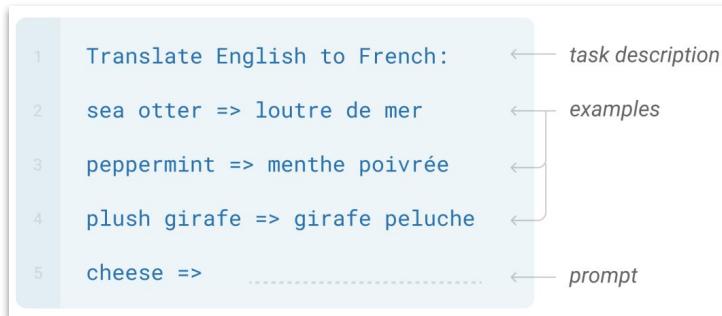
- First showing it how to break down a complex problem into subproblems
- and solve them sequentially.

Prompting method	code-davinci-002
Standard prompting	16.7
Chain-of-Thought	16.2
Least-to-Most	<b>99.7</b>



# Few-shot in-context learning

- **Few-shot:** In addition to the task description, the model sees **a few examples** of the task.
- **Again, no fine-tuning,** GPT-3 doesn't update their parameters!



Q&A  
Answers Generation Conversation

Answer questions based on existing knowledge.

Prompt

I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".

task description

Q: What is human life expectancy in the United States?  
A: Human life expectancy in the United States is 78 years.

task examples

Q: Who was president of the United States in 1955?  
A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?  
A: He belonged to the Republican Party.

Q: What is the square root of banana?  
A: Unknown

Q: How does a telescope work?  
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?  
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?  
A: Unknown

Q: Where is the Valley of Kings?  
A:

task prompt

Sample response

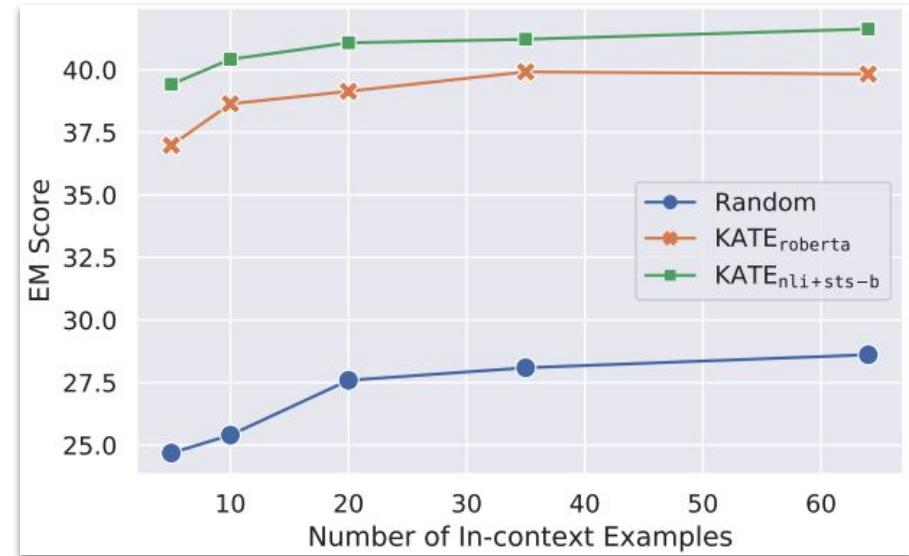
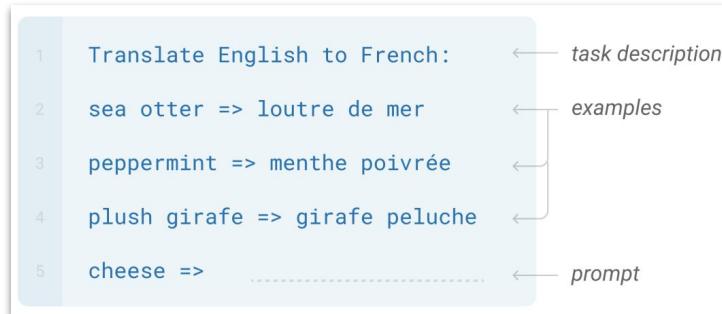
The Valley of Kings is located in Luxor, Egypt.

GPT-3 output

# Few-shot in-context learning

## Retrieving in-context examples is also important!

- For each test instance, we can retrieve different training examples (usually similar to it) to achieve much better results.



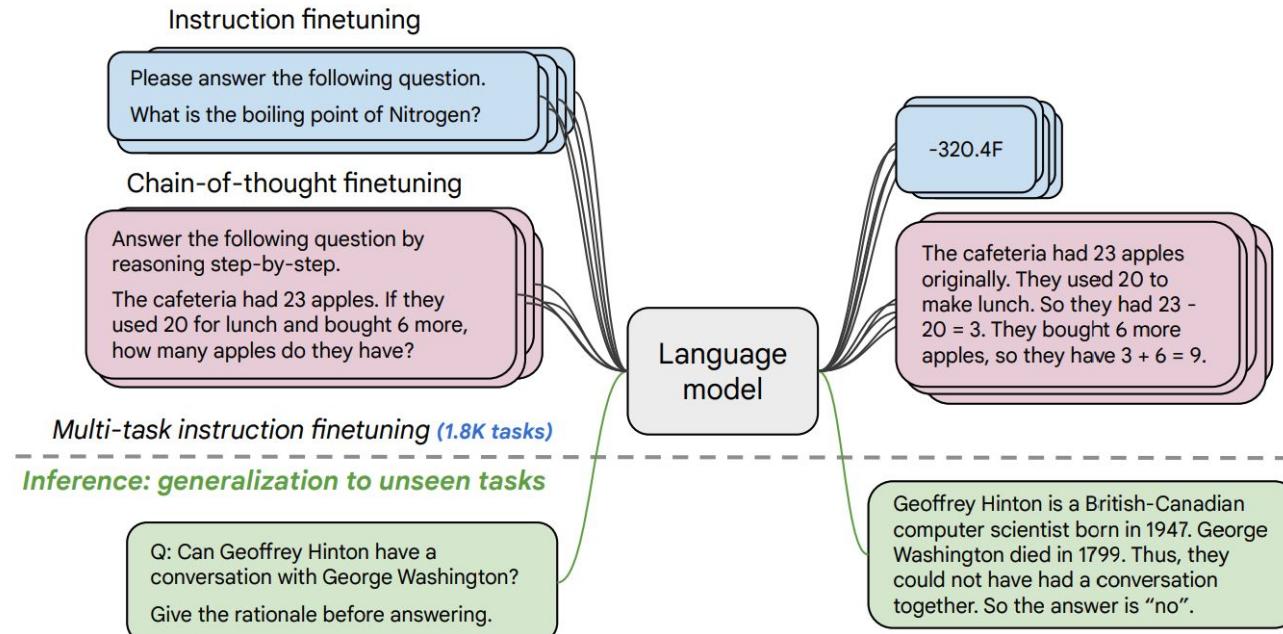
# Why in-context learning with large language models?

- Amazing zero/few-shot performance
  - Save a lot of annotation! 🎉
- Easy to use without training
  - Just talk to them! 👍
- One model for many NLP applications 😊
  - No need to annotate and fine-tune for different tasks

**But**, again, they are sensitive to prompts! Need to design a good **prompt** or train a good example **retriever!** 😅

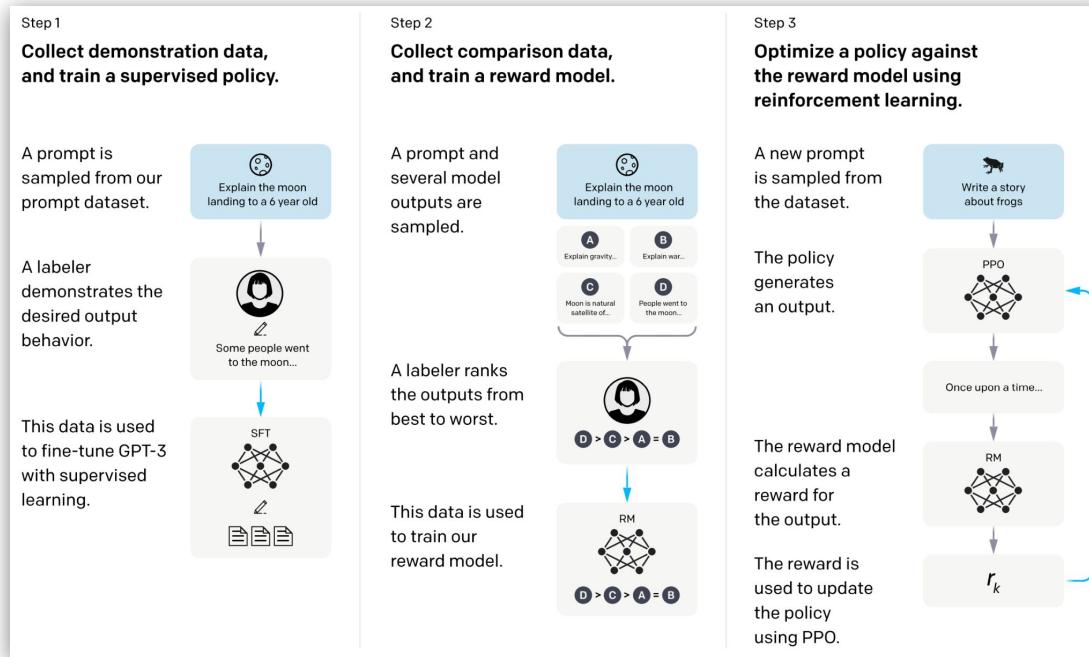
# Instruction tuning

- LM learns to perform many tasks via natural language instructions



# LLM alignment and RLHF

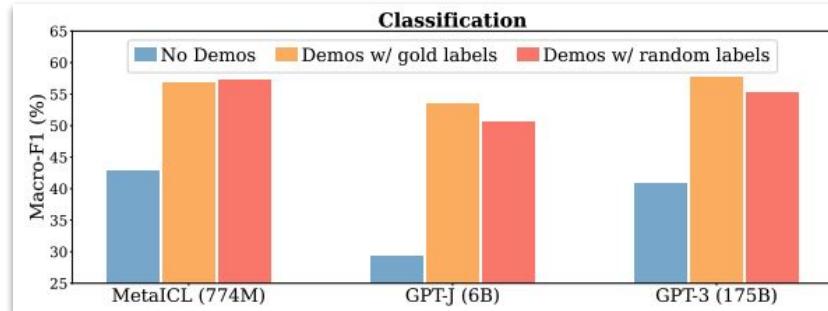
- Align LLMs with user intent on a wide range of tasks by fine-tuning with human feedback



# LLM analysis

## What are these LLMs really doing?

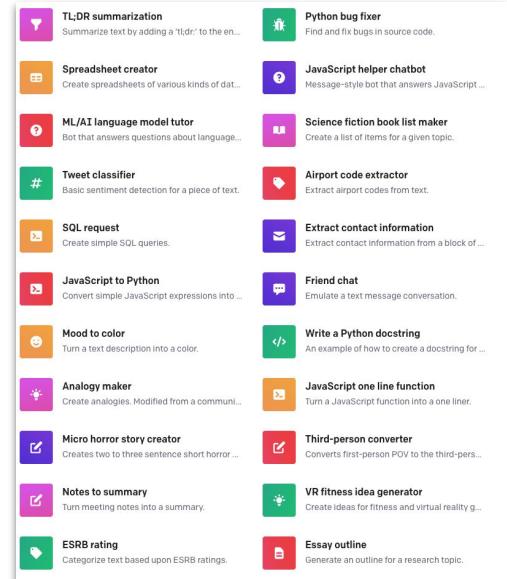
- Large language models (LMs) are able to learn from in-context examples about the task.
- However, there has been little understanding of how the model learns and which aspects of the demonstrations contribute to end task performance.



# Sparse/retrieval-based LLMs

## Do we really need such huge models?

- **Task/domain expert language models: much smaller models for different tasks/domains**
  - Huge models are expensive and slow to run!
  - What if we just care about model performance on certain types of downstream tasks/domains?

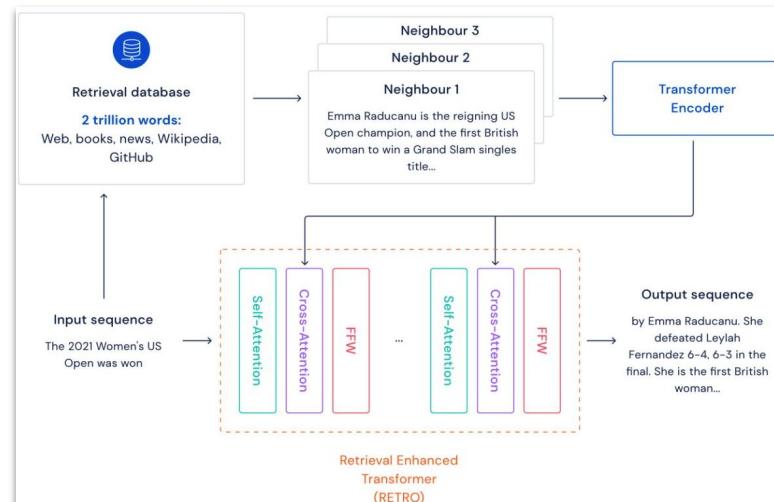


# Sparse/retrieval-based LLMs

## Do we really need to save everything in such large models?

- **Retrieval-augmented language models**

- Augment models with retrieval over a database of text passages including web pages, books, news and code.
- No need to save everything in models' parameters, retrieving knowledge from an external database when it is needed.



# LLM agents

# Traditional NLP tasks -> LLM agent

## Foundational Technologies

- Language Modeling
- Part-of-speech Tagging
- Syntactic Parsing
- Dependency Parsing
- Named Entity recognition
- Coreference resolution
- Word Sense Disambiguation
- Semantic Role Labelling
- .....

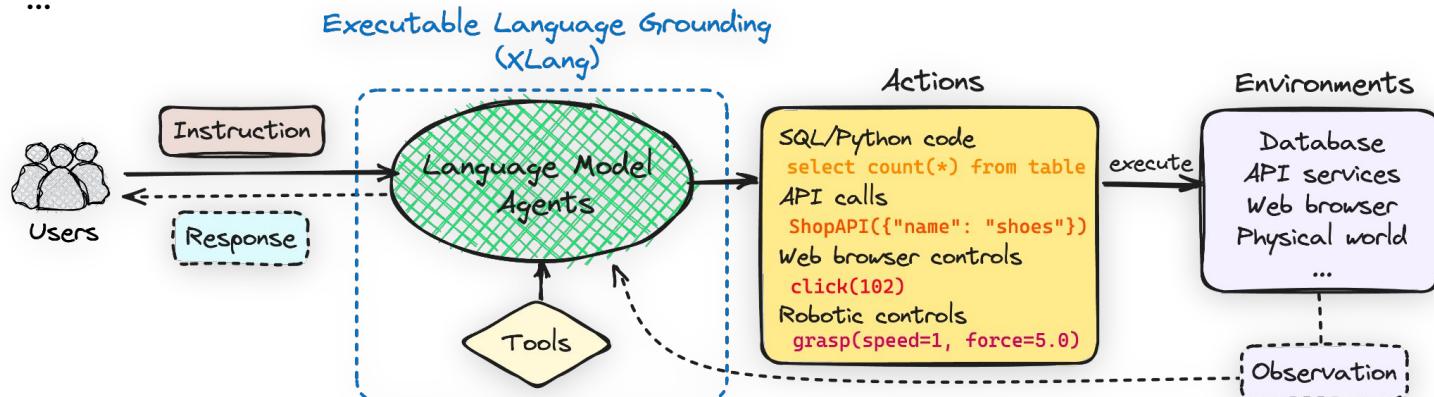
## High-Level Tasks and Applications

- Sentiment Analysis
- Information Extraction
- Machine Translation
- Question Answering
- Semantic Parsing
- Summarization
- Dialogue systems
- Language and Vision
- Data-to-Text Generation
- .....

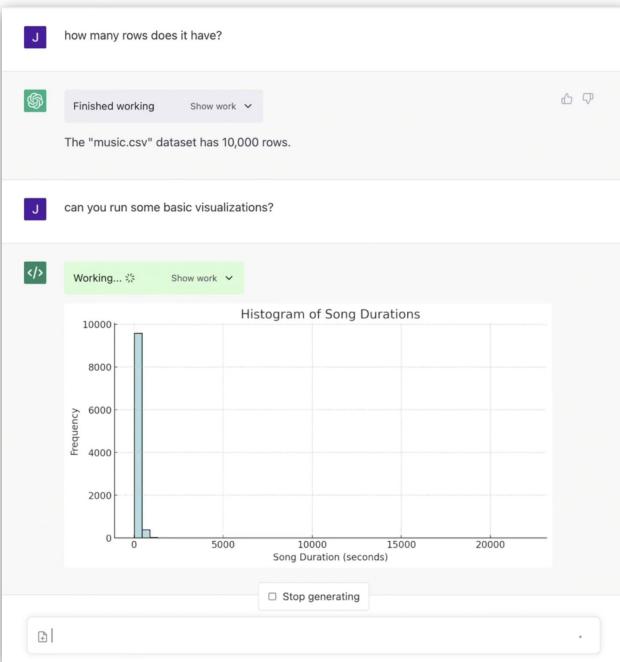
Input X	Output Y	Task
Text	Label	Text Classification (e.g., Sentiment Analysis)
Text	Linguistic Structure	Structured Prediction (e.g., Part-of-Speech Tagging)
Text	Text	Text Generation (e.g., Translation, Summarization)

# Language model agents

- Interact with and learn from humans and real-world environments (*database, web browser, systems, physical world*)
- Access and know how to use tools (*code interpreter, web/apps, robotic arms, search engines, calculator...*)
- Make decisions for solving complex/abstract problems
  - Reason and plan
  - Ground request and take actions in environments
  - Know to ask questions and abstain
  - Self explore and correct
  - ...



# LLM applications: LLM Agent for data analysis



ChatGPT code interpreter

# ChatGPT code interpreter



I am interested in doing some predictive modelling, where we can predict what powers a hero might have based on other factors. how should we approach this?



Predictive modeling for this case can be a bit complex due to the nature of the data. Here are a few things to consider:

- 1. Multi-label Classification:** Each superhero can have multiple powers, so this is a multi-label classification problem. In other words, each superhero can belong to more than one class (power).
- 2. Feature Selection:** We need to decide which features (columns) we want to use as predictors. The superhero's personal characteristics like 'Gender', 'Eye color', 'Race', 'Hair color', 'Height', 'Publisher', 'Skin color', 'Alignment', and 'Weight' could potentially be used as features. Remember that categorical features will need to be converted into a format that can be used in the model (e.g., one-hot encoding).
- 3. Data Preprocessing:** The dataset needs further preprocessing. For instance, we have missing values represented as `np.nan` that we'll need to handle, and we may need to normalize numerical features like 'Height' and 'Weight'.
- 4. Model Selection:** There are many machine learning algorithms that can handle multi-label classification, such as Decision Trees, Random Forests, K-Nearest Neighbors (KNN), and more sophisticated methods like neural networks. We would need to choose a suitable model.
- 5. Evaluation Metrics:** We need to decide on how we will evaluate the performance of our model. Common metrics for multi-label classification include Hamming Loss, Zero-One Loss, and the Jaccard Score.

Given these considerations, a potential first step could be to preprocess the data and prepare it for use in a machine learning model. Does this approach sound good to you?



sure



sure



Finished working Show work ▾

The data preprocessing steps have been completed. Here's what we did:

- Filled in missing values. For categorical features, we filled missing values with the most frequent category. For numerical features, we filled missing values with the mean.
- Normalized numerical features ('Height' and 'Weight') to have a mean of 0 and a standard deviation of 1. This is a common requirement for many machine learning algorithms to perform well.
- Converted categorical features into a format that can be used in the model using one-hot encoding. This is a process by which categorical variables are converted into a form that could be provided to ML algorithms to improve their performance.

```
# Import libraries
from sklearn.preprocessing import LabelEncoder, StandardScaler, MultiLabelBinarizer
from sklearn import tree
from sklearn import datasets
import numpy as np
import pandas as pd
import os

# Define the features and target
features = ['Gender', 'Eye color', 'Race', 'Hair color', 'Height', 'Publisher', 'Skin color', 'Alignment', 'Weight']
target = 'powers'
data = pd.read_csv('superhero_powers.csv')
data['Publisher'] = data['Publisher'].str.replace(' ', '_')
data['Skin color'] = data['Skin color'].str.replace(' ', '_')

# Create a copy of the dataset
data = combined_data.copy()

# Handle missing values
# For categorical features, we'll fill missing values with the most frequent
# For numerical features, we'll fill missing values with the mean.
for feature in features:
    if data[feature].dtype == 'object':
        imputer = SimpleImputer(strategy='most_frequent')
        data[feature] = imputer.fit_transform(data[feature].values.reshape(-1, 1))
    else:
        imputer = SimpleImputer(strategy='mean')
        data[feature] = imputer.fit_transform(data[feature].values.reshape(-1, 1))

# Normalize numerical features to have a mean of 0 and standard deviation of 1
scaler = StandardScaler()
data[['Height', 'Weight']] = scaler.fit_transform(data[['Height', 'Weight']])

# Convert categorical features into a format that can be used in the model
for feature in ['Gender', 'Eye color', 'Race', 'Hair color', 'Alignment']:
    data[feature] = pd.get_dummies(data[feature], prefix=feature)
data = pd.concat([data, pd.get_dummies(data['Publisher'], prefix='Publisher')], axis=1)
data.drop(['Publisher'], axis=1, inplace=True)

# Save the preprocessed dataset
data.to_csv('preprocessed_dataset.csv', index=False)
```

Example  
of code



yes



Great! Before we proceed, we need to further refine the scope of our predictive task. As you might notice, we have a large number of potential powers (labels) to predict, which could

# Web/App Interfaces

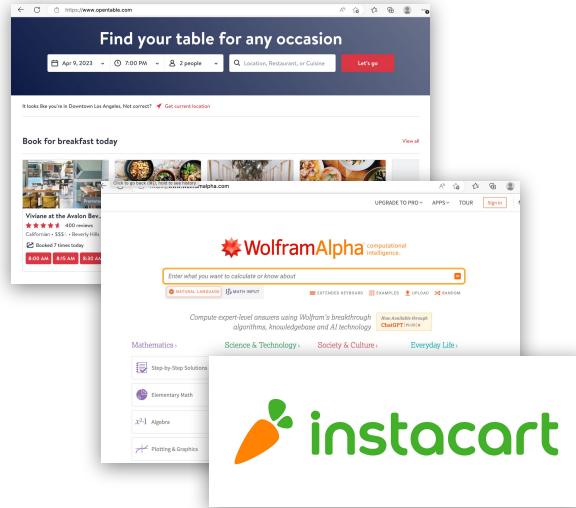
Looking to eat vegan food in San Francisco this weekend. Could you get me one great restaurant suggestion for Saturday and a simple recipe for Sunday (just the ingredients) ? Please calculate the calories for the recipe using WolframAlpha. Finally order the ingredients on Instacart.

The screenshot shows the OpenTable website interface. At the top, there's a search bar with placeholder text "Find your table for any occasion". Below it, a date selector shows "Apr 9, 2023" and a time selector shows "7:00 PM". There are dropdown menus for "2 people" and a search field for "Location, Restaurant, or Cuisine". A red "Let's go" button is prominent. Below the search area, a message says "It looks like you're in Downtown Los Angeles. Not correct? Get current location". Under the heading "Book for breakfast today", there are five restaurant cards with images, names, ratings, reviews, and booking buttons for specific times (8:00 AM, 8:15 AM, 8:30 AM). The restaurants listed are Viviane at the Avalon, Caldo Verde, Cara, Mama Shelter Downtown, and Zinque.

The screenshot shows the WolframAlpha search interface. At the top, there's a search bar with the placeholder text "Enter what you want to calculate or know about". Below the search bar are tabs for "NATURAL LANGUAGE" and "MATH INPUT". To the right are buttons for "UPGRADE TO PRO", "APPS", and "TOUR", along with a "Sign in" link. A promotional banner for ChatGPT Pro is visible. The main content area is divided into several categories: Mathematics (Step-by-Step Solutions, Elementary Math, Algebra, Plotting & Graphics), Science & Technology (Units & Measures, Physics, Chemistry, Engineering), Society & Culture (People, Arts & Media, Personal Finance, Words & Linguistics), and Everyday Life (Personal Health, Dates & Times, Surprises).



# Chat interface: ChatGPT Plugins



Model: Plugins ALPHA • Enabled plugins: 🌐 🎨 🍳

CH Looking to eat vegan food in San Francisco this weekend. Could you get me one great restaurant suggestion for Saturday and a simple recipe for Sunday (just the ingredients)? Please calculate the calories for the recipe using WolframAlpha. Find ingredients on Instacart.

New chat +

Now, let's calculate the calories for this recipe using WolframAlpha.



Greens Restaurant - San Francisco, CA on OpenTable

Used Instacart

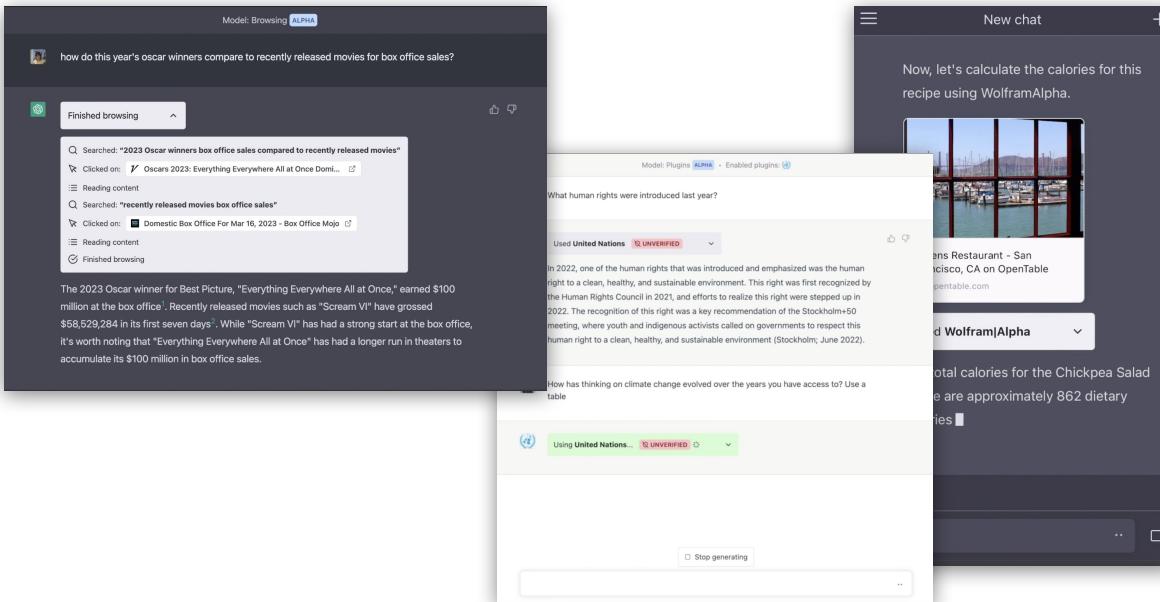
Here's your shopping list for the Chickpea Salad. If you're new to **Instacart**, you can get free delivery when you place this order. Bon appétit!

The total calories for the Chickpea recipe are approximately 862 diet Calories.



Shop this recipe, get free

# Chat interface: ChatGPT Plugins



Plugins: browsing, retrieval, third-party apps/webs...