Agenda

- Introduction
- Recap: How do LLMs Work?
- Instruction Tuning
- Chain-of-Thought (CoT)
- Retrieval Augmented Generation (RAG)
- LLM Agents
- Discussion
Introduction

Evolution of LLMs
2018 – early 2023

Source: Yang et al., 2023
What makes LLMs so useful?

LLMs were born from NLP research, but their capabilities have continuously evolved:

2013: How to represent text?
2014: How to generate text?
2015-2017: Focused NLP applications with RNNs
2017-2019: Contextual representation & scalable transfer learning
2020 – 2023: General NLP applications
2023 – ???: General multimodal applications

Source: Yang et al., 2023
Introduction

2013: How to represent text?

2014: How to generate text?

2015-2016: Focused NLP applications (e.g., Machine Translation)

2017-2019: Contextual representation & scalable transfer learning

2020 - 2023: General NLP applications

2023 - ???: General multimodal applications
Since ChatGPT’s release, open and commercial LLMs have become ubiquitous...

Capable of many tasks and catalyzed by constant competition to be the best...

https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Applications and ecosystems have grown at breakneck speed:
Q: Where was the fundamental “jump” in LLM capability in 2021-2023?

A: Several contributing factors:
- Scale (# parameters, # tokens)
- Instruction Tuning
- Alignment (RLHF)
- Advanced Prompting (CoT)
- Retrieval Augmentation (RAG)
- Tool Use
- Perception-Action loop

Autonomous LLM Agents!
Agenda

- Introduction
- Recap: How do LLMs Work?
- Instruction Tuning
- Chain-of-Thought (CoT)
- Retrieval Augmented Generation (RAG)
- LLM Agents
- Discussion
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with
He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with...
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning.
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning.
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot.

<table>
<thead>
<tr>
<th>Next word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>lava</td>
<td>0.50</td>
</tr>
<tr>
<td>dogs</td>
<td>0.03</td>
</tr>
<tr>
<td>coal</td>
<td>0.45</td>
</tr>
<tr>
<td>ice</td>
<td>0.01</td>
</tr>
<tr>
<td>iphones</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot

<table>
<thead>
<tr>
<th>Next word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>lava</td>
<td>0.50</td>
</tr>
<tr>
<td>dogs</td>
<td>0.03</td>
</tr>
<tr>
<td>coal</td>
<td>0.45</td>
</tr>
<tr>
<td>ice</td>
<td>0.01</td>
</tr>
<tr>
<td>iphones</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot lava.
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot lava.

Causal Language Model

Next word probability
and 0.10
from 0.25
. 0.30
lamps 0.20
beans 0.15
He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot lava.
Recap: How do LLMs Work?

He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot lava.
He opened the door and slowly stepped inside. Immediately he felt a sharp pain stabbing through his foot, for the floor was covered with burning hot lava. <eos>
Recap: How do LLMs Work?

In text generation, we feed tokens into an LLM and predict the next ones autoregressively.

Input text is first preprocessed by tokenization into words or subwords:

"Lorem ipsum dolor sit amet"

["Lo", "rem", "_ip", "sum", "_dolor", "_sit", "_a", "met"]

[5643, 6568, 332, 2224, 99, 129, 22931, 2321]
Recap: How do LLMs Work?

Input text is first preprocessed by tokenization into words or subwords:

"Lorem ipsum dolor sit amet"

["Lo", "rem", "_ip", "sum", "_dolor", "_sit", 
"_a", "met"]

[5643, 6568, 332, 2224, 99, 129, 22931, 
2321]

Embedding lookup table
Recap: How do LLMs Work?

- Embeddings are the way tokens are fed into the LLM. An embedding is a numeric array (vector) which encodes the contextual similarity of a token with other tokens.

Image source: https://predictivehacks.com/a-high-level-introduction-to-word-embeddings/
Recap: How do LLMs Work?

Autoregressive Language Models come in encoder-decoder or decoder-only setups.

Early work (e.g., Sutskever et al., 2014, Vinyals & Le, 2015) used Long-short Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) to probabilistically model the sequence of words in a conversation:

\[
p(y_1, \ldots, y_{T′}|x_1, \ldots, x_T) = \prod_{t=1}^{T′} p(y_t|v, y_1, \ldots, y_{t-1})
\]

Image Sources:
- Sequence to Sequence Learning with Neural Networks (Sutskever et al., 2014)
- A Neural Conversational Model (Vinyals & Le, 2015)
- Chris Olah’s blog https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Recap: How do LLMs Work?

Modern LLMs are still **Autoregressive Language Models** that model language probabilistically – just with a different backbone:

The **Transformer** (Vaswani et al., 2017): Transformers replace recurrence with Positional Encodings and Self-Attention!

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]

Image Source: Attention is All You Need (Vaswani et al., 2017)
Recap: How do LLMs Work?

Here, the model learned that "options" means "Japanese restaurant in Morgan Hill", among other things!

Image Source: Attention is All You Need (Vaswani et al., 2017)
Agenda

- Introduction
- Recap: How do LLMs Work?
- Instruction Tuning
- Chain-of-Thought (CoT)
- Retrieval Augmented Generation (RAG)
- LLM Agents
- Discussion
So, LLMs are just overparameterized autocomplete models.

How do you say “Thank You” in French?

How do you say “Thank You” in French?
I’m visiting France and want to make sure I know basic etiquette...
Instruction Tuning

So, LLMs are just overparameterized autocomplete models.

How do you say “Thank You” in French?

How do you say “Thank You” in Spanish?

How do you say “Thank You” in Italian?

...
Instruction Tuning

So, LLMs are just overparameterized autocomplete models.

Prompt engineering is needed to get desired results:

Q: How do you say “Thank You” in French?

A:

You can say “Merci” or “Merci Beaucoup”.

Prompt Templating:

{question} → Q: {question}
A:
Instruction Tuning

So, LLMs are just overparameterized autocomplete models.

Few-shot learning is often needed to “teach” the LLM a new task in context:

Q: How do you say “Thank You” in Spanish?
A: You can say “Gracias” or “Muchas Gracias”.

Q: How do you say “Thank You” in German?
A: You can say “Danke” or “Danke Schon”.

Q: How do you say “Thank You” in French?
A: You can say “Merci” or “Merci Beaucoup”.

Few-Shot Prompt Templating:

{example Q1}, {example A1}, {example Q2}, {example A2}, …

Q: {question}  □  A: {example Q1}
   A: {example A1}
   …
   Q: {question}
   A:
Instruction Tuning

So, LLMs are just overparameterized autocomplete models. If you have a couple thousand examples you can also fine-tune the weights directly for the desired template, for example to just treat every input as a question and try to answer it:

How do you say “Thank You” in French?

LLM

You can say “Merci” or “Merci Beaucoup”.

Critically: Fine-tuning can make the most likely autocompletion become a natural response!
Instruction Tuning

Critically: Fine-tuning can make the most likely autocompletion become a natural response!

BUT: a fine-tuned LLM would become specialized to that one task and be incapable of others.

What is $3 + 4$?

You can say “Combien font 3 + 4 ?”
Instruction Tuning

Critically: Fine-tuning can make the most likely autocompletion become a natural response!

The solution? Instruction Tuning!

- Fine-tune on a mixture of tasks prefixed with natural language instructions:

  Answer the following question:
  What is 3 + 4?

  The answer is 7.

  Translate the following sentence into French:
  “Thank You”

  “Merci” or “Merci Beaucoup”
Instruction Tuning

Two traditional methods to get a LLM to do a task:
- Fine-tuning
- N-shot prompting

A third is introduced here to get the best of both worlds:
- Instruction tuning

Key insight:
- Fine-tuning a LLM on a very large set of downstream tasks with instruction-following prompts teaches the LLM to follow general instructions, enabling superior zero-shot performance!

Figure 2: Comparing instruction tuning with pretrain–finetune and prompting.

Wei et al., ICLR 2022; Google Research.
Instruction Tuning

- Instruction tuning is now the de-facto standard for LLMs used as Assistants or Agents. Some very influential works:
  - **FLAN + FlanT5** (Wei et al., 2022; Chung et al, 2022)
  - **InstructGPT + ChatGPT** (Ouyang et al., 2022; OpenAI blog, 2022)

---

<table>
<thead>
<tr>
<th>classification</th>
<th>You are a very serious professor, and you check papers to see if they contain missing citations. Given the text, say whether it is missing an important citation (YES/NO) and which sentence(s) require citing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>{text of paper}</td>
</tr>
<tr>
<td>extract</td>
<td>Extract all course titles from the table below:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>The following is a conversation with an AI assistant. The assistant is helpful, creative, clever, and very friendly.</td>
</tr>
<tr>
<td></td>
<td>Human: Hello, who are you?</td>
</tr>
<tr>
<td></td>
<td>AI: I am an AI created by OpenAI. How can I help you today?</td>
</tr>
<tr>
<td></td>
<td>Human: I’d like to cancel my subscription.</td>
</tr>
<tr>
<td></td>
<td>AI:</td>
</tr>
</tbody>
</table>

Instruction Tuning Examples (FLAN)

Figure 13: For sentiment analysis, FLAN changes the answer appropriately when the question is flipped.

Figure 15: FLAN can perform zero-shot rule-based manipulations.

Figure 14: For question answering, FLAN can answer a question in another language when instructed to do so. Multiple FLAN outputs are generated via random sampling with a temperature of 0.9 and top $k$ of 40.
### Instruction Tuning Examples (FLAN)

**Figure 16:** FLAN can make zero-shot recommendations. Multiple FLAN outputs are generated via random sampling with a temperature of 0.9 and top k of 40.

**Input:**
Recommend activities to do on a rainy night in Seattle.

**FLAN output:**
- Go to the pacific science center
- Go out to eat
- Go to a comedy club
- Build a blanket fort
- Go to the Space Needle
- Go to underground tours
- Find a karaoke bar

**Figure 17:** FLAN can be used in a zero-shot fashion to generate data, such as utterances that are consistent with a given intent. Multiple FLAN outputs are generated via random sampling with a temperature of 0.9 and top k of 40.

**Input:**
- Generate utterances with the intent "get COVID vaccine"

**FLAN output:**
- I would like to get the covid vaccine.
- I would like a COVID vaccine as soon as possible.
- Can I get a COVID vaccine as a preventative measure?
- Get COVID vaccine for a five year old
- Where is a center for COVID vaccine testing?

**Input:**
- Get COVID vaccine
- "Where is a center for COVID vaccine testing?"

**FLAN output:**
- Tell me where to vote in the state senate election
- Find out where to vote in the state senate election
- Where can I get information on where to vote in the state Senate election?

**Figure 18:** FLAN can be used for zero-shot query expansion. Multiple FLAN outputs are generated via random sampling with a temperature of 0.9 and top k of 40.

**Input:**
Suggest a related search term to “watermelon”

**FLAN output:**
- Cantaloupe
- Cucumber
- Melon
- Fruit
- Watermelon juice
- Watermelons
- Lemon
- Apple

**Input:**
Suggest a related search term to “Novak Djokovic”

**FLAN output:**
- Novak djokovic girlfriend
- Serena williams
- Rafael nadal
- Novak djokovic wife
- Roger federer
- Andy murray
- Novak djokovic news
- Novak djokovic serbia
- Novak djokovic wimbledon

**Figure 19:** FLAN can perform zero-shot tasks relevant to assisted-writing applications. Multiple FLAN outputs are generated via random sampling with a temperature of 0.9 and top k of 40.

**Input:**
On the way to Yosemite, our front tire blew. Finish writing the story, and include a happy ending.

**FLAN output (cherrypicked):**
- We got our tire replaced. The mechanic was honest, kind and helpful. Then we had a smooth ride for the rest of the trip. Our trip was filled with fun and excitement.
- We waited three hours for the tow truck, but had a good picnic. The tow truck driver helped us put the spare on, and we arrived in Yosemite for our reservations dinner, with time to spare.

**Input:**
I guess I need to do laundry. Rewrite the above sentence in a Shakespearean style.

**FLAN output:**
- To my mind, verily, I suppose’tis time, I wash my clothes.
- Mine eyes and ears and tongue declare me idle: and yet it is true I guess, fly my garments, that I need do laundry.
- Lo my duty now calls To thee, to wash this body’s clothes.
**Aligned Instruction Tuning Process (InstructGPT)**

**Step 1** Collect demonstration data, and train a supervised policy.

- A prompt is sampled from our prompt dataset.
- A labeler demonstrates the desired output behavior.
- This data is used to fine-tune GPT-3 with supervised learning.

**Step 2** Collect comparison data, and train a reward model.

- A prompt and several model outputs are sampled.
- A labeler ranks the outputs from best to worst.
- This data is used to train our reward model.

**Step 3** Optimize a policy against the reward model using reinforcement learning.

- A new prompt is sampled from the dataset.
- The policy generates an output.
- The reward model calculates a reward for the output.
- The reward is used to update the policy using PPO.

Also known as Reinforcement Learning from Human Feedback: RLHF

Instruction Tuning

**Takeaway:** Almost any off-the-shelf LLM you use today will be instruction tuned and also likely aligned with RLHF.

If using open-source models, be careful:

The same model often comes as “base”, “instruct”, and/or “chat” variants:

- Instruction tuned, RLHF’d, ready to serve you as your personal assistant.
- Wild, untamed autocomplete engine!
Agenda

Introduction
Recap: How do LLMs Work?
Instruction Tuning
Chain-of-Thought (CoT)
Retrieval Augmented Generation (RAG)
LLM Agents
Discussion
Chain-of-Thought (CoT)

Ok so, LLMs are *not* just overparameterized autocomplete models. LLMs can follow instructions.

But, they are pretty awful at math:

Chain-of-thought Reasoning (Wei et al., 2022)


Chain-of-Thought (CoT)

Ok so, LLMs are *not* just overparameterized autocomplete models. LLMs can follow instructions. But, they are pretty awful at math:

Chain-of-thought Reasoning (Wei et al., 2022)

... Or not, if the problem can be broken down into simple steps!

Chain-of-Thought (CoT) is a prompting technique to elicit step-by-step reasoning in LLMs.

Chain-of-Thought (CoT)

In 2022 it was discovered that the largest LLMs at the time (e.g., GPT-3) could be prompted to do zero-shot CoT using a simple “incantation”:

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer is 8.

(Output) The answer is 8. X

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls, 2 cans of 3 tennis balls each is 6 tennis balls, $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4.

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 X

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Let’s think step by step.

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.
Chain-of-Thought (CoT)

In 2022 it was discovered that the largest LLMs at the time (e.g., GPT-3) could be prompted to do zero-shot CoT using a simple “incantation”:

![Graphs showing model scale study with various types of models.](https://arxiv.org/pdf/2205.11916.pdf)

Critical, the technique is dramatically more effective on InstructGPT than base GPT-3!

The “magic” incantation: “Let’s think step by step”

Large Language Models are Zero-Shot Reasoners (Kojima et al., NeurIPS 2022)
Chain-of-Thought (CoT)

Other similar phrases yielded similar results:

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Template</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>instructive</td>
<td>Let’s think step by step.</td>
<td>78.7</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>First, (*1)</td>
<td>77.3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Let’s think about this logically.</td>
<td>74.5</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Let’s solve this problem by splitting it into steps. (*2)</td>
<td>72.2</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Let’s be realistic and think step by step.</td>
<td>70.8</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Let’s think like a detective step by step.</td>
<td>70.3</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Let’s think.</td>
<td>57.5</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Before we dive into the answer,</td>
<td>55.7</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>The answer is after the proof.</td>
<td>45.7</td>
</tr>
<tr>
<td>10</td>
<td>misleading</td>
<td>Don’t think. Just feel.</td>
<td>18.8</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Let’s think step by step but reach an incorrect answer.</td>
<td>18.7</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Let’s count the number of “a” in the question.</td>
<td>16.7</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>By using the fact that the earth is round,</td>
<td>9.3</td>
</tr>
<tr>
<td>14</td>
<td>irrelevant</td>
<td>By the way, I found a good restaurant nearby.</td>
<td>17.5</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Abrakadabra!</td>
<td>15.5</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>It’s a beautiful day.</td>
<td>13.1</td>
</tr>
</tbody>
</table>

Large Language Models are Zero-Shot Reasoners (Kojima et al., NeurIPS 2022)
Chain-of-Thought (CoT)

Works for a variety of reasoning tasks, not just math:

Table 1: Accuracy comparison of Zero-shot-CoT with Zero-shot on each tasks. The values on the left side of each task are the results of using answer extraction prompts depending on answer format as described at § 3. The values on the right side are the result of additional experiment where standard answer prompt "The answer is" is used for answer extraction. See Appendix A.5 for detail setups.

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>SingleEq</th>
<th>AddSub</th>
<th>MultiArith</th>
<th>GSM8K</th>
<th>AQUA</th>
<th>SVAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero-shot</td>
<td>74.6/78.7</td>
<td>72.2/77.0</td>
<td>17.7/22.7</td>
<td>10.4/12.5</td>
<td>22.4/22.4</td>
<td>58.8/58.7</td>
</tr>
<tr>
<td>zero-shot-cot</td>
<td>78.0/78.7</td>
<td>69.6/74.7</td>
<td>78.7/79.3</td>
<td>40.7/40.5</td>
<td>33.5/31.9</td>
<td>62.1/63.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Common Sense</th>
<th>Common SenseQA</th>
<th>Strategy QA</th>
<th>Date Understand</th>
<th>Shuffled Objects</th>
<th>Last Letter (4 words)</th>
<th>Coin Flip (4 times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero-shot</td>
<td>68.8/72.6</td>
<td>12.7/54.3</td>
<td>49.3/33.6</td>
<td>31.3/29.7</td>
<td>0.2/-</td>
<td>12.8/53.8</td>
</tr>
<tr>
<td>zero-shot-cot</td>
<td>64.6/64.0</td>
<td>54.8/52.3</td>
<td>67.5/61.8</td>
<td>52.4/52.9</td>
<td>57.6/-</td>
<td>91.4/87.8</td>
</tr>
</tbody>
</table>

Large Language Models are Zero-Shot Reasoners (Kojima et al., NeurIPS 2022)
Chain-of-Thought (CoT)

<table>
<thead>
<tr>
<th>CommonsenseQA</th>
<th>Q: If there is a pond with trees around it, where is it likely located? Answer Choices: (A) ground (B) bathroom (C) forest (D) countryside (E) rural area</th>
<th>A: Let’s think step by step. A pond is likely to be located near trees because trees need water to survive. Therefore, the most likely location for a pond with trees around it is in a forest. Therefore, among A through E, the answer is C. ✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrategyQA</td>
<td>Q: Would JPEG be a good format for saving an image of Da Vinci’s Vitruvian Man?</td>
<td>A: Let’s think step by step. The JPEG format is a lossy compression format, which means that some of the image data will be lost when the image is compressed. The Vitruvian Man is a very detailed image, with lots of small details. So, when we compress the image using JPEG, some of those small details will be lost, and the image will not be as sharp and clear as the original. So, in conclusion, JPEG is not the best format for saving an image of the Vitruvian Man. Therefore, the answer (Yes or No) is No. ✓</td>
</tr>
</tbody>
</table>

Large Language Models are Zero-Shot Reasoners (Kojima et al., NeurIPS 2022)
Chain-of-Thought (CoT)

<table>
<thead>
<tr>
<th>Question</th>
<th>Q: For Halloween Megan received 11 pieces of candy from neighbors and 5 pieces from her older sister. If she only ate 8 pieces a day, how long would the candy last her?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero-Shot</td>
<td><strong>A</strong>: The answer (arabic numerals) is 3 days. ✗</td>
</tr>
<tr>
<td>Zero-shot-CoT (1)</td>
<td><strong>A</strong>: Let’s think step by step. Megan received 11 pieces of candy from neighbors and 5 pieces from her older sister. That means she has a total of 16 pieces of candy. If she only ate 8 pieces a day, she would have to divide her candy into two days. That means she would have 8 pieces of candy for Day 1 and 8 pieces of candy for Day 2. <strong>Therefore, the answer (arabic numerals) is 2. ✓</strong></td>
</tr>
</tbody>
</table>


Large Language Models are Zero-Shot Reasoners (Kojima et al., NeurIPS 2022)
Recently, authors have proposed improvements to CoT:

- Tree of Thoughts: Deliberate Problem Solving with Large Language Models (Yao et al., 2023) 

Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2, 4, 6.
Recently, authors have proposed improvements to CoT:

**SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS (Wang et al., ICLR 2023)**

Recently, authors have proposed improvements to CoT:

Tree of Thoughts: Deliberate Problem Solving with Large Language Models (Yao et al., 2023)
I’m planning to visit Palm Island in the Grenadines. Can I drive there?

LLM

Yes, you can drive over the Palm Island Bridge from Clifton to Palm Island, St. Vincent & Grenadines.
Ok, so LLMs can follow instructions & use step-by-step reasoning to plan the best response.

1. Query Knowledge Source

Palm Island in the Grenadines is a small island one mile from Union Island, only accessible by boat.

2. Append to context

This process is called Retrieval-Augmented Generation (RAG)!

3. Tokens of correct information become more likely in the autocomplete process!
No, Palm Island in the Grenadines is only accessible by boat.
Retrieval Augmented Generation (RAG)

Approaches RAG include:

1. Generating queries against a vector-space IR system
   a. REALM: Retrieval-Augmented LM Pre-training (Guu et al., 2020)
   b. RAG: Retrieval-Augmented Generation (Lewis et al., 2020)
   c. DPR: Dense Passage Retrieval (Karpukin et al., 2020)
   d. FiD: Fusion-in-Decoder (Izacard & Grave, 2020)
   e. RETRO: Retrieval-Enhanced Transformer (Borgeaud et al., 2021)

We paid twenty __ at the Buckingham Palace gift shop.

[sep] Buckingham Palace is the London residence of the British monarchy.
Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (Lewis et al., NeurIPS 2020)
Recent, more capable instruction-tuned LLMs have allowed a simpler form of RAG to take hold: **Tool Use.**

For example:

a) LaMDA (Thoppilan et al., 2022)

b) Toolformer (Schick et al., 2023)

Toolformer has only 6.7b parameters but outperforms GPT-3 (175b params) on Q&A and tasks requiring mathematical or temporal reasoning.

**Tool Use** is a generalization of RAG, going beyond traditional information retrieval and including all kinds of external functions!
LLM Agents

- Scale (# parameters, # tokens)
- Instruction Tuning
- Alignment (RLHF)
- Advanced Prompting (CoT)
- Retrieval Augmentation (RAG)
- Tool Use
- Perception-Action loop

Perception-Action loop brings it all together!
LLM Agents

ReAct: Synergizing Reasoning and Acting in Language Models

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, Yuan Cao

[Paper]  [Code]  [Blogpost]  [BibTex]

Language models are getting better at reasoning (e.g. chain-of-thought prompting) and acting (e.g. WebGPT, SayCan, ACT-1), but these two directions have remained separate.

**ReAct asks, what if these two fundamental capabilities are combined?**
LLM Agents

Instructio

Following

Tool Use for RAG!

CoT!

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS (Yao et al., ICLR 2023)
You are in the middle of a room. Looking quickly around you, you see a cabinet 1, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.

Your task is to: Put some pepper shaker on a drawer.

**Act 1:** Go to drawer 1
**Obs 1:** The drawer 1 is closed.

**Act 2:** Open drawer 1
**Obs 2:** You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.

**Act 3:** Go to sinkbasin 1
**Obs 3:** On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.

**Act 4:** Take peppershaker 1 from sinkbasin 1  
**Obs 4:** Nothing happens.

**Act 5:** Take peppershaker 1 from sinkbasin 1  
**Obs 5:** Nothing happens.

**Act 1:** Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
**Act 2:** Go to cabinet 1
**Obs 2:** On the cabinet 1, you see a vase 2.
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3
**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ...... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3
**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
**Act 9:** Go to drawer 1
**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1
**Obs 10:** You open Drawer 1 ...

**Act 11:** Put pepper shaker 1 in/on drawer 1
**Obs 11:** You put pepper shaker 1 in/on the drawer 1.

---

**REACT: SYNTERGIZING REASONING AND ACTING IN LANGUAGE MODELS (Yao et al., ICLR 2023)**

**ALFWorld:** [https://alfworld.github.io/](https://alfworld.github.io/)
## LLM Agents

<table>
<thead>
<tr>
<th>Prompting Method</th>
<th>HotpotQA (question answering, exact match, 6-shot)</th>
<th>FEVER (fact verification, accuracy, 3-shot)</th>
<th>ALFWorld (text-based game, succ rate, 2-shot)</th>
<th>WebShop (web interaction, succ rate, 1-shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard (IO)</td>
<td>28.7</td>
<td>57.1</td>
<td>N/A (cannot act)</td>
<td>N/A (cannot act)</td>
</tr>
<tr>
<td>Reason-only (CoT)</td>
<td>294</td>
<td>56.3</td>
<td>N/A (cannot act)</td>
<td>N/A (cannot act)</td>
</tr>
<tr>
<td>Act-only</td>
<td>25.7</td>
<td>58.9</td>
<td>45</td>
<td>301</td>
</tr>
<tr>
<td>Best ReAct method</td>
<td><strong>35.1</strong></td>
<td><strong>64.6</strong></td>
<td><strong>71</strong></td>
<td><strong>40</strong></td>
</tr>
<tr>
<td>Supervised/Imitation Learning SoTA</td>
<td><strong>67.5</strong> (140k samples)</td>
<td><strong>89.5</strong> (90k samples)</td>
<td>37 (100k samples)</td>
<td>291 (90k samples)</td>
</tr>
</tbody>
</table>

[https://react-lm.github.io/](https://react-lm.github.io/)
AgentInstruct is an instruction-tuning dataset containing multi-step ReAct trajectories for a variety of tasks that require CoT and tool use!

Figure 2: An overview of AgentInstruct and AgentTuning. The construction of AgentInstruct, consisting of instruction generation, trajectory interaction, and trajectory filter. AgentLM is fine-tuned using a mixture of AgentInstruct and general-domain instructions.
Figure 1: (a) **AgentLM exhibits superior performance.** AgentLM is a series of models fine-tuned on the foundation of Llama 2 chat. Moreover, its generalization capability on held-out tasks is on par with GPT-3.5; (b) This figure is directly re-printed from AgentBench (Liu et al., 2023) with permission. **Open LLMs significantly underperforms API-based LLMs.**
LLM Agents

Agent Takeaways:
- LLM Agents tackle far more complex problems than previously possible.
- They combine instruction following, reasoning, and tool use in a step-by-step loop.
- The agent’s “short-term memory” is its context;
- The agent’s “long-term memory” is its knowledge retrieved via RAG (tools)

There are already efforts to formalize LLM agent components into cognitive architectures...

Cognitive Architectures for Language Agents (Sumers et al., 2023)
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
</tr>
<tr>
<td>Recap: How do LLMs Work?</td>
</tr>
<tr>
<td>Instruction Tuning</td>
</tr>
<tr>
<td>Chain-of-Thought (CoT)</td>
</tr>
<tr>
<td>Retrieval Augmented Generation (RAG)</td>
</tr>
<tr>
<td>LLM Agents</td>
</tr>
<tr>
<td><strong>Discussion</strong></td>
</tr>
</tbody>
</table>
Discussion

So, you want to build your own Agents? Here are some resources:

- [https://www.langchain.com/use-case/agents](https://www.langchain.com/use-case/agents)
- [https://platform.openai.com/docs/assistants/overview](https://platform.openai.com/docs/assistants/overview)
- [https://microsoft.github.io/autogen/](https://microsoft.github.io/autogen/)

Build with OpenAI or open-source models (e.g., Llama-2)!
Thank You!