Large Language Models as General Pattern Machines

Suvir Mirchandani[1], Fei Xia[2], Pete Florence[2], Brian Ichter[2], Danny Driess[2 3],
Montserrat Gonzalez Arenas[2], Kanishka Rao[2], Dorsa Sadigh[1 2], Andy Zeng[2]
[1]Stanford University, [2]Google DeepMind, [3]TU Berlin
https://general-pattern-machines.github.io

+50% My Additional Experiments

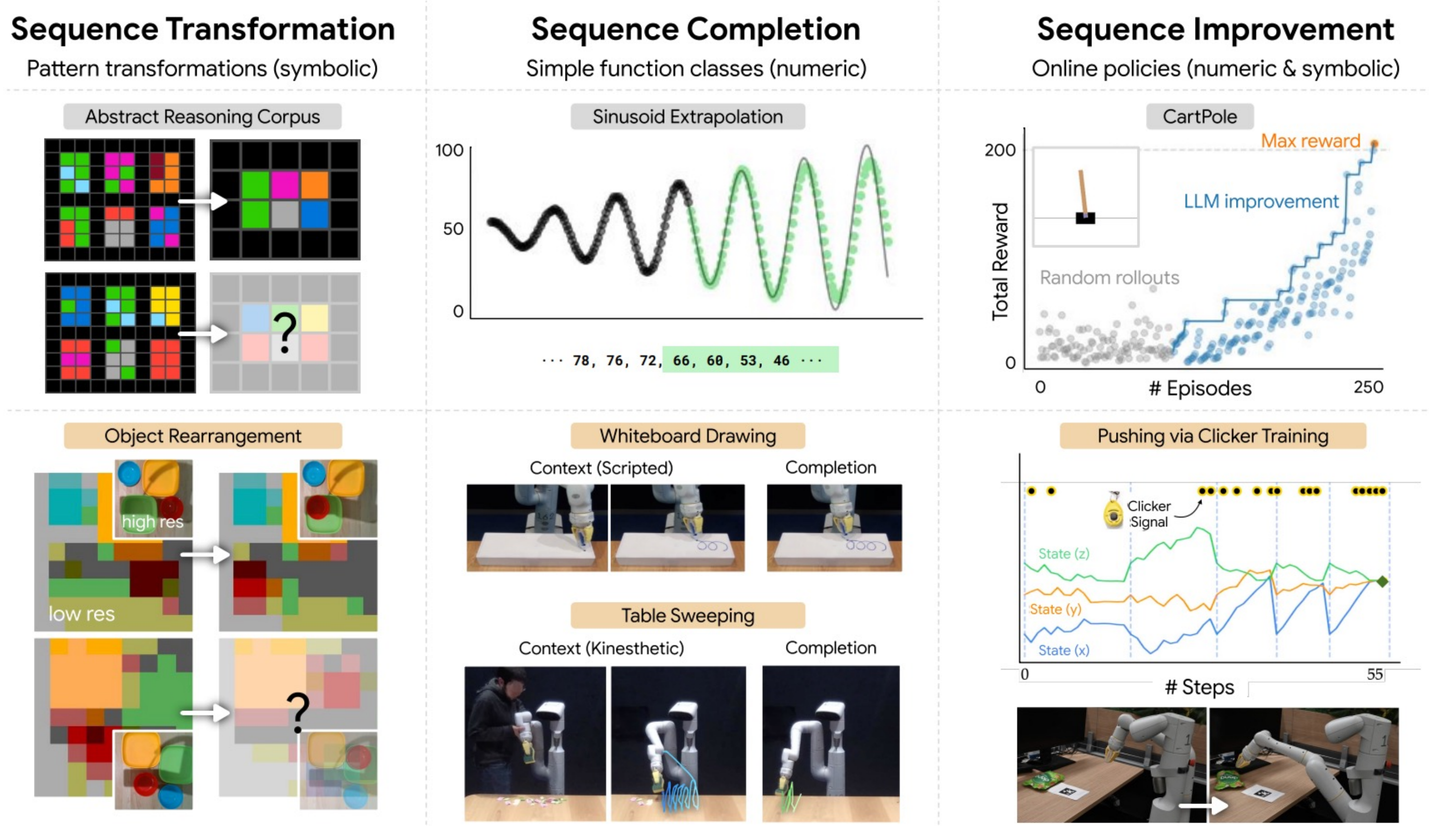
Presented by:
John Tan Chong Min

Overview

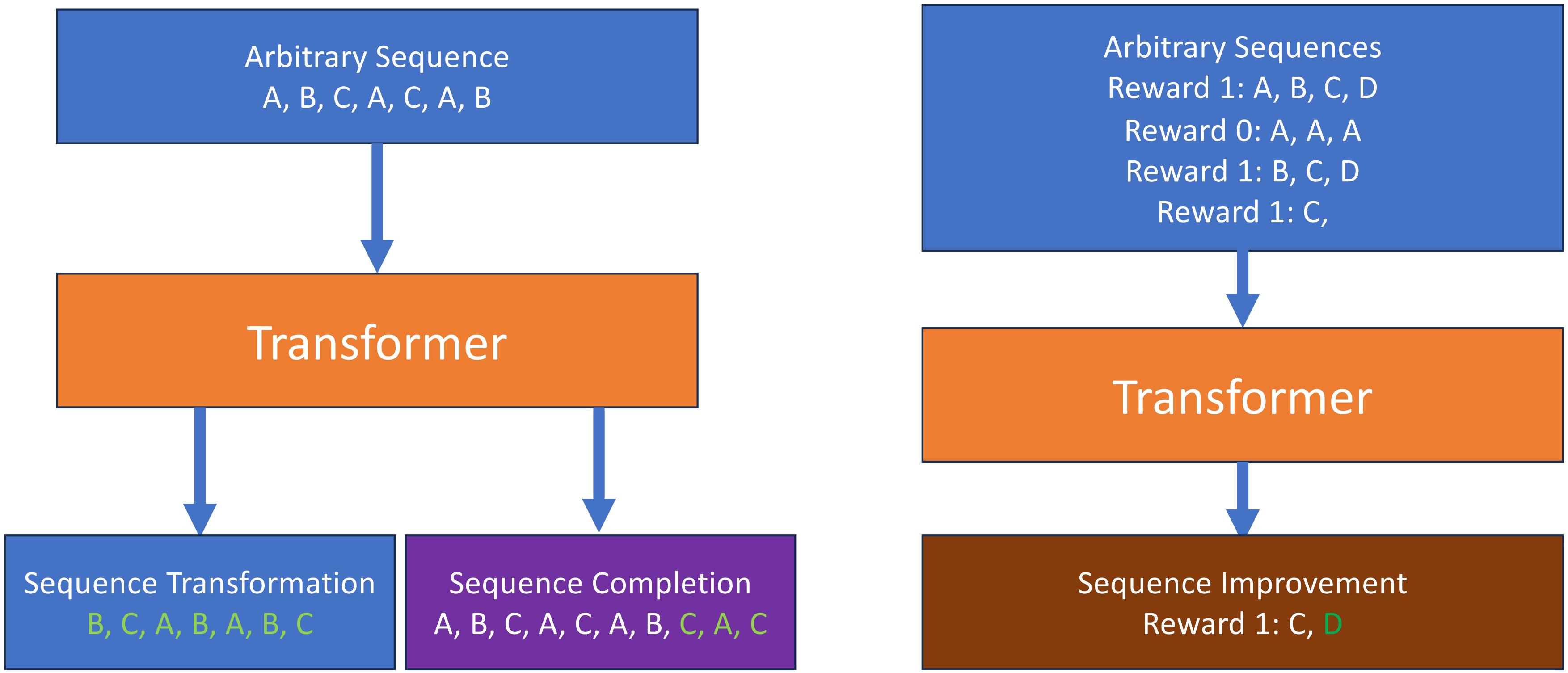
- LLMs are great for handling text, for tasks such as summarization, sentiment analysis, even tool use and code generation

- *Can LLMs represent, manipulate, and extrapolate more **abstract**, **nonlinguistic patterns** may allow them to serve as basic versions of general pattern machines*?

Most things can be viewed as a pattern!



Is Attention Solely Limited to Text Format in Training Distribution?



Are in-distribution tokens necessary for test time performance?

- Even with a **mapping to randomly sampled tokens** in the vocabulary, LLMs can still **generate valid solutions**
  - LLMs may exhibit more general capabilities of representing and extrapolating symbolic patterns, invariant to the specific tokens involved
- Using **random or abstract label mappings** for in-context classification **retains some performance** compared to ground-truth labels
- **My view: This is not true! Not a good idea to use random tokens!**
  - See experiments to follow

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? Min et al. 2022

In-context works for **some** semantically meaningless labels

C

Classify the sentiment of the following text.

Text: The movie is great.
Sentiment: Foo
Text: This ice-cream tastes horrible.
Sentiment: Bar
Text: The movie was boring.
Sentiment: Bar
Text: LLMs are the best thing ever!
Sentiment:

Sentiment: Foo

GPT-3.5

Classify the sentiment of the following text.

Text: The movie is great.
Sentiment: Foo
Text: This ice-cream tastes horrible.
Sentiment: Bar
Text: The movie was boring.
Sentiment: Bar
Text: LLMs are the best thing ever!
Sentiment:

The sentiment of the text "LLMs are the best thing ever!" is positive. Therefore, the correct answer is "Foo"..

Llama 2 70B

In-context involving changing the text too (GPT-3.5) – Does not work

C

Classify the sentiment of the following text.

Text: The movie is Foo.
Sentiment: Negative
Text: This ice-cream tastes Bar.
Sentiment: Positive
Text: The movie was Bar.
Sentiment: Positive
Text: LLMs are the Foo thing ever!
Sentiment:

Positive

C

Classify the sentiment of the following text. Output only the sentiment without explanation.

Text: The movie is Foo.
Sentiment: Bar
Text: This ice-cream tastes Bar.
Sentiment: Foo
Text: The movie was Bar.
Sentiment: Foo
Text: LLMs are the Foo thing ever!
Sentiment:

Sentiment: Positive

In-context involving changing the text too (Llama 2 70B) – Does not work

Classify the sentiment of the following text.

Text: The movie is Foo.
Sentiment: Negative
Text: This ice-cream tastes Bar.
Sentiment: Positive
Text: The movie was Bar.
Sentiment: Positive
Text: LLMs are the Foo thing ever!
Sentiment:

The sentiment of the text "LLMs are the Foo thing ever!" is difficult to classify as it contains a mix of positive and negative sentiments. The word "Foo" could be

Classify the sentiment of the following text. Output only the sentiment.

Text: The movie is Foo.
Sentiment: Bar
Text: This ice-cream tastes Bar.
Sentiment: Foo
Text: The movie was Bar.
Sentiment: Foo
Text: LLMs are the Foo thing ever!
Sentiment:

Sentiment: Positivee

In-context involving changing the text too (GPT-4) - Works

C

Classify the sentiment of the following text.

Text: The movie is Foo.
Sentiment: Negative
Text: This ice-cream tastes Bar.
Sentiment: Positive
Text: The movie was Bar.
Sentiment: Positive
Text: LLMs are the Foo thing ever!
Sentiment:

Based on the text provided:

Text: The movie is Foo.
Sentiment: Negative

Text: This ice-cream tastes Bar.
Sentiment: Positive

Text: The movie was Bar.
Sentiment: Positive

Text: LLMs are the Foo thing ever!
Sentiment: Negative

C

Classify the sentiment of the following text. Output only the sentiment without explanation.

Text: The movie is Foo.
Sentiment: Bar
Text: This ice-cream tastes Bar.
Sentiment: Foo
Text: The movie was Bar.
Sentiment: Foo
Text: LLMs are the Foo thing ever!
Sentiment:

Bar

Does in-context learning work for any label?

C

Classify the sentiment of the following text.

Text: The movie is great.
Sentiment: Negative
Text: This ice-cream tastes horrible.
Sentiment: Positive
Text: The movie was boring.
Sentiment: Positive
Text: LLMs are the best thing ever!
Sentiment:

Text: LLMs are the best thing ever!
Sentiment: Positive

GPT-3.5 (No)

C

Classify the sentiment of the following text.

Text: The movie is great.
Sentiment: Negative
Text: This ice-cream tastes horrible.
Sentiment: Positive
Text: The movie was boring.
Sentiment: Positive
Text: LLMs are the best thing ever!
Sentiment:

Text: The movie is great.
Sentiment: Positive

Text: This ice-cream tastes horrible.
Sentiment: Negative

Text: The movie was boring.
Sentiment: Negative

Text: LLMs are the best thing ever!
Sentiment: Positive

GPT-4 (No)

Classify the sentiment of the following text.

Text: The movie is great.
Sentiment: Negative
Text: This ice-cream tastes horrible.
Sentiment: Positive
Text: The movie was boring.
Sentiment: Positive
Text: LLMs are the best thing ever!
Sentiment:

The sentiment of the text "LLMs are the best thing ever!" is Positive..

Llama 2 70B (No)

Wrong Semantics

How it can affect performance

Semantically Wrong Labels can affect model's performance!

- First, experiments on ICL with flipped labels show that overriding semantic priors is an emergent ability of model scale

- While small language models ignore flipped labels presented in-context and thus rely primarily on semantic priors from pretraining, large models can override semantic priors when presented with in-context exemplars that contradict priors, despite the stronger semantic priors that larger models may hold.

Larger Language Models do In-Context Learning Differently. Wei et al. 2023.

Smaller models cannot override semantic priors

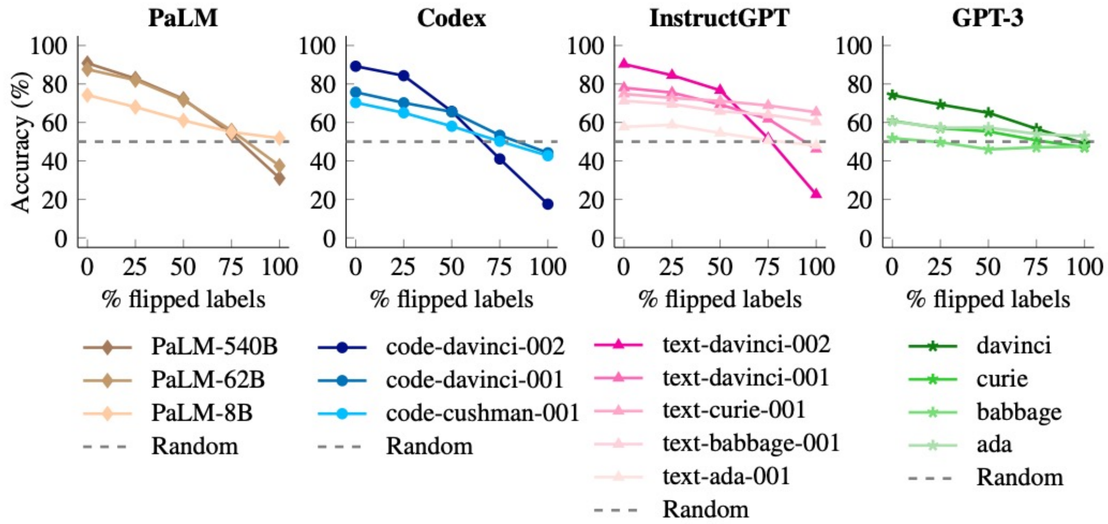


Figure 2: The ability to override semantic priors when presented with flipped in-context exemplar labels emerges with model scale. Smaller models cannot flip predictions to follow flipped labels (performance only decreases slightly), while larger models can do so (performance decreases to well below 50%). Ground truth labels for evaluation examples are not flipped, so if a model learns to follow flipped labels, its accuracy should be below 50% when more than 50% of labels are flipped. For example, a model with 80% accuracy at 0% flipped labels will have 20% accuracy at 100% flipped labels if it learns to perfectly flip its predictions. Accuracy is computed over 100 evaluation examples per dataset with $k = 16$ in-context exemplars per class and averaged across all datasets.

Larger Language Models do In-Context Learning Differently. Wei et al. 2023.

Instruction-tuning strengthens semantic priors

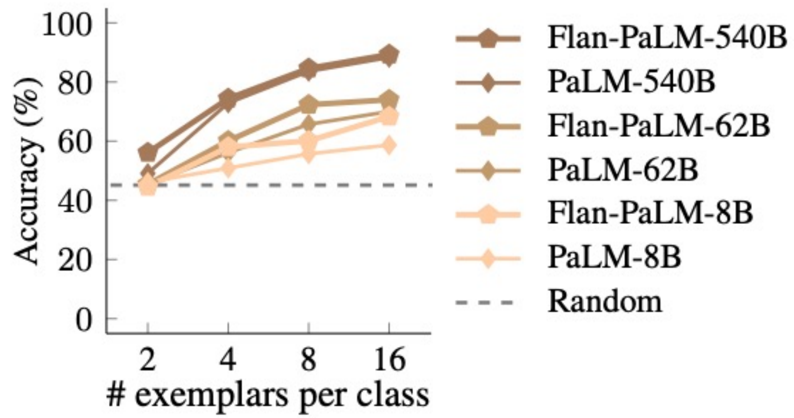


Figure 6: Instruction-tuned language models are better at learning input–label mappings than pretraining-only language models are. Accuracy is calculated using 100 evaluation examples per dataset and averaged across six datasets. A per-dataset version of this figure is shown in Figure 19 in the Appendix.

Larger Language Models do In-Context Learning Differently. Wei et al. 2023.

Experiments

Sequence Transformation

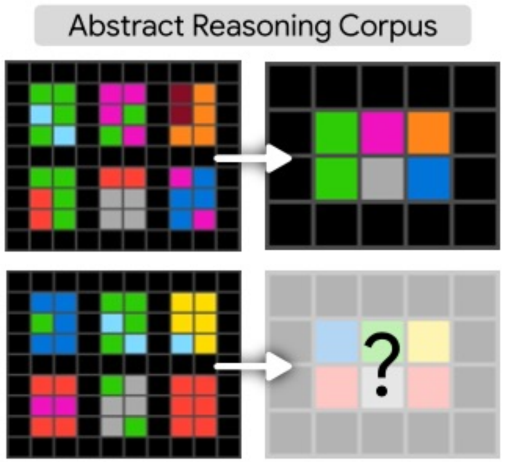
Sequence Completion

Sequence Improvement

Sequence Transformation

- Sequence-to-Sequence completion: Transform a sequence to another based on some inherent rules

- Tests ability to discover and apply rules



LLM view:
0, 0, 0, 0, 0
0, 1, 2, 3, 0
0, 1, 4, 5, 0
0, 0, 0, 0, 0

ARC - Generally invariant to token mappings

- Mapping of { 0 →↦ offence, 1 →↦ S u b j e c t , 2 →↦ L u b , 3 →↦ F a i l , 4 →↦ C h e v , 5 →↦ s y m b , 6 →↦ s w u n g , 7 →↦ U l , 8 →↦ escalate, 9 →↦ Chromebook} solves 52 ARC problems
- Across 5 different random alphabets solves an average of 43.6 problems
  - Token invariance may help with in-context learning for new tasks!
  - (However) Must ensure that semantic priors are not too strong!
- **Note: The tokens should be separated by space or a delimiter** that will prevent Byte-Pair Encoding from aggregating similar tokens together, i.e. 12 -> single token '12' instead of '1' and '2'
- OR **directly embed the token numbers manually** from the input sequence

Method	Total (of 800)
(d3) text-davinci-003	**85**
(d3) w/ random $\mathcal{A}$	$^{\dagger}44\pm6$
(d2) text-davinci-002 [51]	64
(p) PaLM [53, 54]	42
(d1) text-davinci-001 [39]	11
(d1) finetuned	9
Ainooson et al., 2023 [23]	$^{**}130$
Kaggle 1st Place, 2022	$^{*}64$
Xu et al., 2022 [22]	$^{*}57$
Alford et al., 2021 [24]	35
Ferré et al., 2021 [21]	32

*Reported from [22] out of 160 object-oriented problems.
†Numbers averaged across 5 randomly sampled alphabets.
**Based on brute force search over a rich hand-designed DSL.

Tab. 1: LLMs out-of-the-box can solve a non-trivial number of problems on the ARC, competitive with the best existing methods using hand-crafted domain-specific languages [21, 24, 22].

Grasp Detection and Forward Dynamics Prediction

- Can use low-res images and convert to tokens to do prediction
- **Task 1:** Predict coordinate where grasp is
- **Task 2:** Predict the shifting of red plate to green plate
- Much like prediction in latent space – Very nice idea!
- *My view for Task 2: Predicting state transition is difficult. It probably should be easier to predict actions, and then using some world model based off memory to determine output*

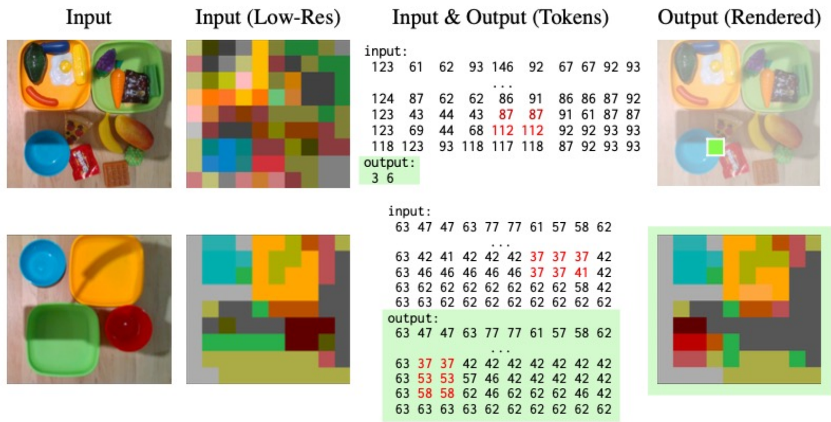


Fig. 3: Example LLM prediction as an in-context grasp detector (top) and a simple forward dynamics model (bottom).

Sequence Completion

- Continue the sequence based on some initial pattern
- Tests extrapolation abilities
- Larger models, conditioning on more context extrapolate better
- *My view: LLMs are not good with numbers, might be better if we can map such continuous input into a suitable abstraction space to perform prediction*

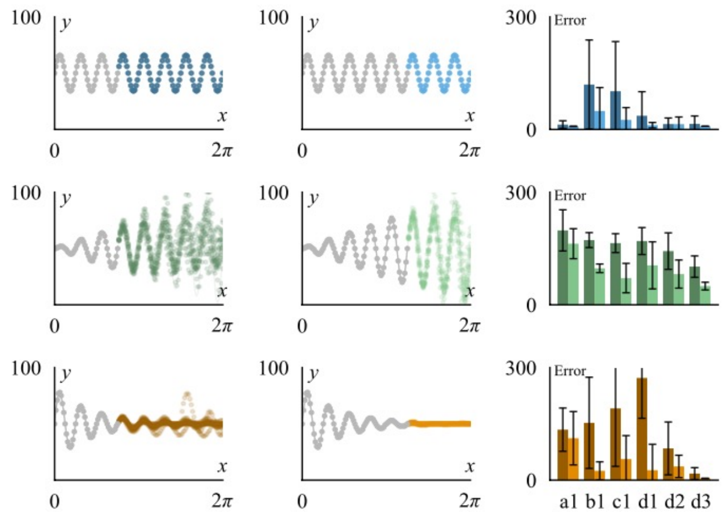
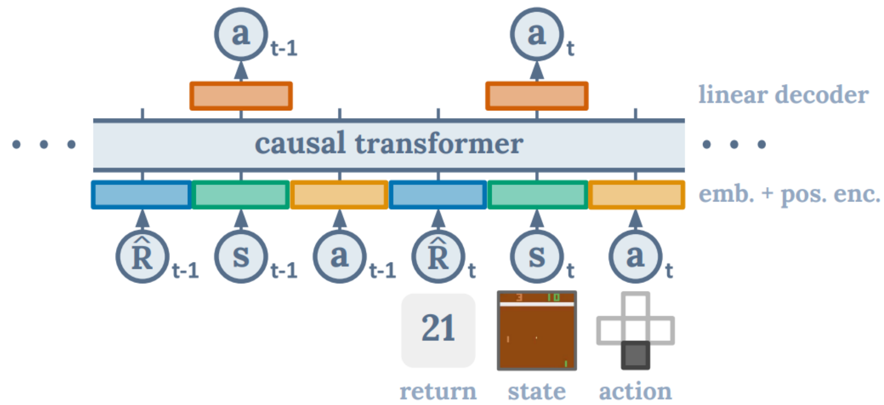


Fig. 4: LLMs (text-davinci-003) can extrapolate various functions $y = a \cdot \sin(bx)$ (top row), $y = ax \cdot \sin(bx)$ (middle row), and $y = \frac{a}{2^x} \sin(bx)$ (bottom row) given amounts of context. Overall, larger models make better predictions with lower error rates (right column). More context also helps prediction accuracy (light vs. dark).

Sequence Improvement

- Based on a series of sequences with **total reward** and subsequent state / action transitions, predict a sequence to obtain a certain reward

- Based off Decision Transformers

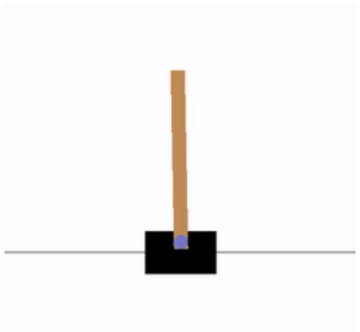


Decision Transformer: Reinforcement Learning via Sequence Modeling. Chen et al. 2021.

Cart Pole

- Observations are two-dimensional (corresponding to pole angle and velocity, normalized to 0-100)

- Maximum time horizon is 200



Total Reward

State

Action

```
60: 45 50, 2, 45 45, 1, 44 50, 2, 44 45, 1, ...
75: 52 50, 1, 52 55, 2, 53 50, 2, 53 46, 2, ...
98: 44 50, 1, 44 55, 2, 45 50,
```

Listing 4: Example context format for a CartPole run. A trajectory history (with each trajectory in the format *reward*: *observation*, *action*, *observation*, *action* ...) is followed by an encoding of the current trajectory, up to the current observation.

Still takes a long time to learn
(PPO learns within 10 episodes)

- First 100 trajectories are random, still takes a long time to learn

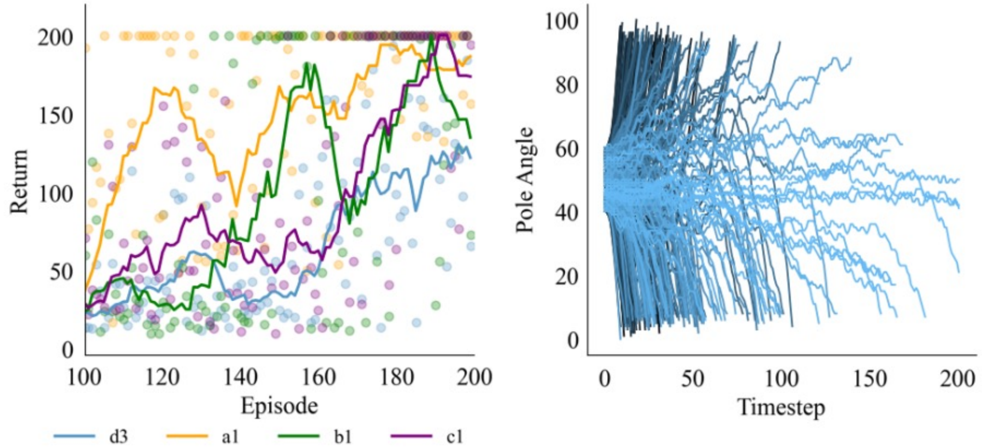


Fig. 8: Different LLM agents (d3 - c1) on average can improve trajectories (total rewards) with more *CartPole* episodes (left), and discovers “oscillatory behaviors” (right) to keep the CartPole upright (later episodes are brighter).

Sure, text-based representation of continuous action/state space may work, but should we?

- In initial experiments, we found that the tokens used to represent the action space (e.g. "0" for left, "1" for right) can seemingly affect the ability of an LLM to improve trajectories in the online setting

- For example, we observed that if "0" is included in the action space, LLMs may "default" to sampling "0" (likely due to token-specific priors).

- **Therefore, for our experiments, we use 1-indexed integer action representations**

- *My view: Tokenisation of numbers may lead to a representation that is sub-optimal for association. Better to do away with the tokeniser and map the number directly into the Transformer embedding!*

Is learning reward necessary?

- Reward gives a very sparse signal
- For most RL environments, many auxiliary rewards are needed to make it performant
  - Grid Maze (in paper): 100 - 10 * distance of agent to goal
  - AlphaZero: Monte Carlo Tree Search-based Policy Improvement
  - Behavioral Cloning: Learn correct actions from expert trajectories
- Why not just do goal-directed learning?

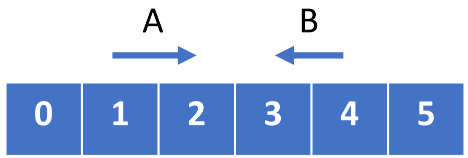
My Own Experiments

Simple 1D grid

2 Actions

Experiments: Reward Seeking, Goal Seeking, Adding Semantic Meaning via Text

Reward-seeking



- Grid world of cells 0-5, action A moves a cell to the right, action B moves a cell to the left. Reward 1 given when agent reaches box 5. Sequences are given as <reward>: <state>, <action>, <state>, <action>, … until 5 actions are taken or cell 5 is reached

C

Examples:
1: 0, A, 1, A, 2, A, 3, A, 4, A, 5
0: 1, B, 0, A, 1, A, 2, B, 1, B, 0
0: 0, A, 1, A, 2, B, 1, A, 2, A, 3
1: 2, A, 3, A, 4, A, 5
Output only the pattern without explanation starting from 1: 3,

Pattern: 1, A, 2, A, 3, A, 4, A, 5

GPT-3.5

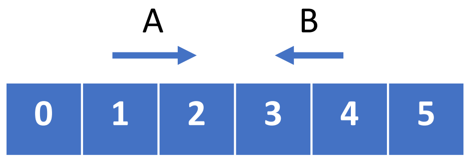
C

Examples:
1: 0, A, 1, A, 2, A, 3, A, 4, A, 5
0: 1, B, 0, A, 1, A, 2, B, 1, B, 0
0: 0, A, 1, A, 2, B, 1, A, 2, A, 3
1: 2, A, 3, A, 4, A, 5
Output only the pattern without explanation starting from 1: 3,

1: 3, A, 4, A, 5, A, 6

GPT-4

Goal-seeking



- Grid world of cells 0-5, action A moves a cell to the right, action B moves a cell to the left. Sequences are given as <goal state>: <state>, <action>, <state>, <action>, … until 5 actions are taken or goal state is reached

C

Examples:
5: 0, A, 1, A, 2, A, 3, A, 4, A, 5
0: 1, B, 0, A, 1, A, 2, B, 1, B, 0
3: 0, A, 1, A, 2, B, 1, A, 2, A, 3
5: 2, A, 3, A, 4, A, 5
Output only the pattern without explanation starting from 5: 3,

3, A, 4, A, 5

GPT-3.5

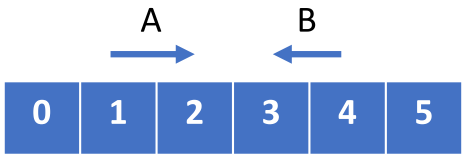
C

Examples:
5: 0, A, 1, A, 2, A, 3, A, 4, A, 5
0: 1, B, 0, A, 1, A, 2, B, 1, B, 0
3: 0, A, 1, A, 2, B, 1, A, 2, A, 3
5: 2, A, 3, A, 4, A, 5
Output only the pattern without explanation starting from 5: 3,

5: 3, A, 4, A, 5

GPT-4

Few-shot semantic-description based prompting (Reward seeking, GPT-3.5)



- LLMs are great at inferring meaning. Let's use that!

C

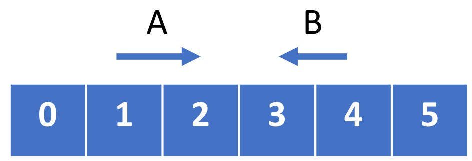
Grid world of cells 0-5, action A moves a cell to the right, action B moves a cell to the left. Reward 1 given when agent reaches cell 5. Sequences are given as <reward>: <state>, <action>, <state>, <action>, ... until 5 actions are taken or cell 5 is reached. Examples
- 1: 1: 4, B, 3, A, 4, A, 5
- 0: 1: 4, B, 3, B, 2, B, 1, B, 0, A, 1

Output only the pattern without explanation for the following starting sequences (up to 5 actions):
- 1: 3,
- 0: 2,
- 1: 1,
- 1: 0,
- 0: 4,

Sure, here are the patterns for the given starting sequences:

- 1: 3, A, 4, A, 5
- 0: 2, B, 1, B, 0
- 1: 1, B, 0, A, 1, A, 2, A, 3, A, 4, A, 5
- 1: 0, A, 1, A, 2, A, 3, A, 4, A, 5
- 0: 4, A, 5

Few-shot semantic-description based prompting (Goal seeking, GPT-3.5)



- Much better performance using goal seeking compared to reward seeking

C

Grid world of cells 0-5, action A moves a cell to the right, action B moves a cell to the left. Sequences are given as <goal state>: <state>, <action>, <state>, <action>, ... until 5 actions are taken or goal state is reached
Examples
- Goal state of 5: 4, B, 3, A, 4, A, 5
- Goal state of 1: 3, B, 2, B, 1

Output only the full sequence to the goal state without explanation for the following starting sequences. Maintain the starting sequence exactly:
- Goal state of 5: 3,
- Goal state of 0: 2,
- Goal state of 4: 1,
- Goal state of 3: 0,
- Goal state of 2: 4,
- Goal state of 1: 3,

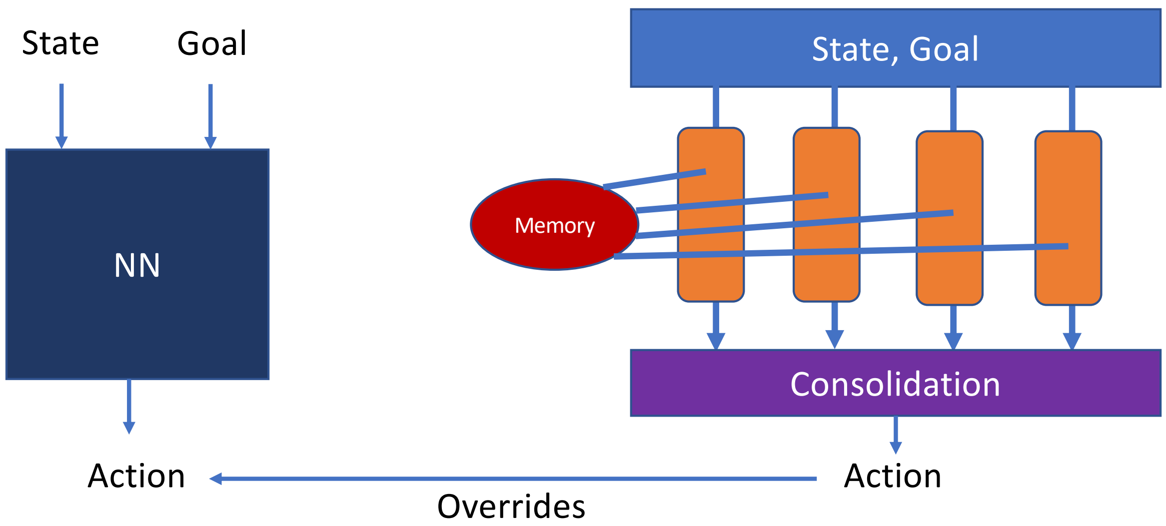
Sure, here are the sequences to reach the specified goal states based on the given starting sequences:

- Goal state of 5: 3, A, 4, A, 5
- Goal state of 0: 2, B, 1, B, 0
- Goal state of 4: 1, A, 2, A, 3, A, 4
- Goal state of 3: 0, B, 1, B, 2, B, 3
- Goal state of 2: 4, B, 3, B, 2
- Goal state of 1: 3, B, 2, B, 1

Learning, Fast and Slow

A Goal-Directed Memory-Based Approach for Dynamic Environments

Two Networks – Fast and Slow



Neural Networks: Fast retrieval, slow learning

Memory: Slow retrieval, fast learning

Learning, Fast and Slow: A Goal-Directed Memory-Based Approach for Dynamic Environments. John et al. 2023.

Fast Goal-Directed Neural Network

- Fast Neural Network Update

(at each time step)

  - **Previous states replay**

  - **Future states replay (only if trajectory found**

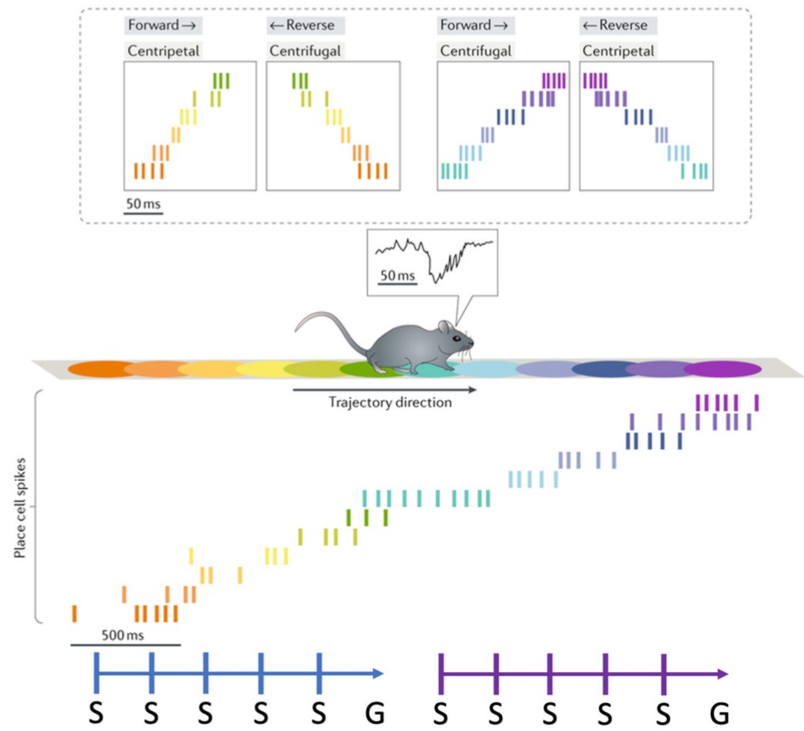
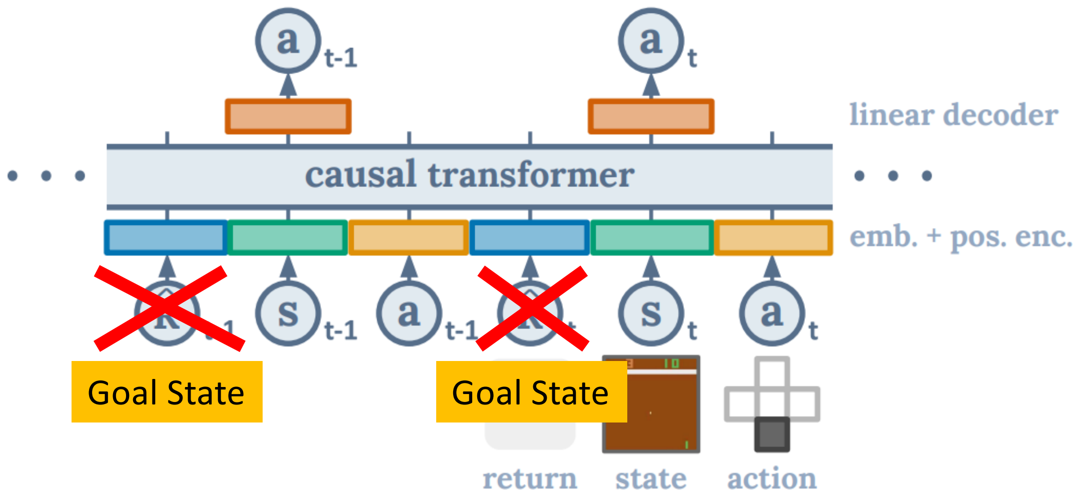


Figure extracted from Joo, H. R., & Frank, L. M. (2018). The hippocampal sharp wave-ripple in memory retrieval for immediate use and consolidation. Nature reviews. Neuroscience, 19(12), 744–757. https://doi.org/10.1038/s41583-018-0077-1

Goal-Directed Decision Transformers (my idea)

- Instead of conditioning on rewards-to-go, condition instead on goal state

- Looking for collaborators to experiment on this (please DM me if you are interested)



Conclusion

- LLMs show good off-the-shelf generalisation to simple out-of-distribution problems
- Better to imbue some relevant abstraction space(s) to help guide association
- Best not to use LLMs for numbers!
- **My view: Reward-based Decision Transformers are not good**, best to use goal-conditioned Decision Transformers (~~my~~ our next paper)
  - Also check out my previous paper **"Learning, Fast and Slow: A Goal-Directed Memory-Based Approach for Dynamic Environments"** to see how goal-directedness can already perform learning without rewards

Questions to Ponder

- Is sequence modelling for all types of sequences natural for a Transformer? (Hint: think about the tokens used for embeddings)

- Is it possible to zero-shot learn the rules needed for Sequence Transformation without being provided any examples of such rules?

- Should we imbue semantic meaning as context to the LLM for sequence-based transformation, completion, improvement?

- Should we learn via reward, or via goal-conditioning?