

# Interacting with LLMs for Grounded Tasks

Daniel Fried



Language  
Technologies  
Institute

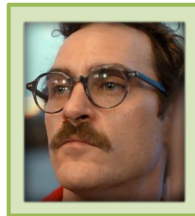
Carnegie  
Mellon  
University

# Language Interfaces

---

## Science Fiction

*Her*, 2013



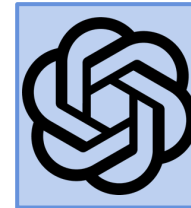
Let's start with your emails. You have several thousand emails regarding LA Weekly, but it looks like you haven't worked there in many years.

Oh yeah, I guess I was saving those because in some of them I thought I might have written some funny stuff.

Yeah, there are some funny ones. I'd say there are about 86 that we should save. We can delete the rest.

## Today

ChatGPT, 2023



Please help me organize my emails.

Sure! Here are some tips for organizing your emails.

1. Unsubscribe: Reduce the number of unwanted emails by unsubscribing from mailing lists that you no longer need.
2. ...

Thanks, but I want you to do it.

I'm sorry, I am a text-based AI language model and do not have access to your email or the ability to physically sort them for you.

Perception?

Uncertainty?

Planning?

Semantics?

In-context  
learning

Zero-shot  
abilities

Code generation

LLM

Dialogue  
abilities

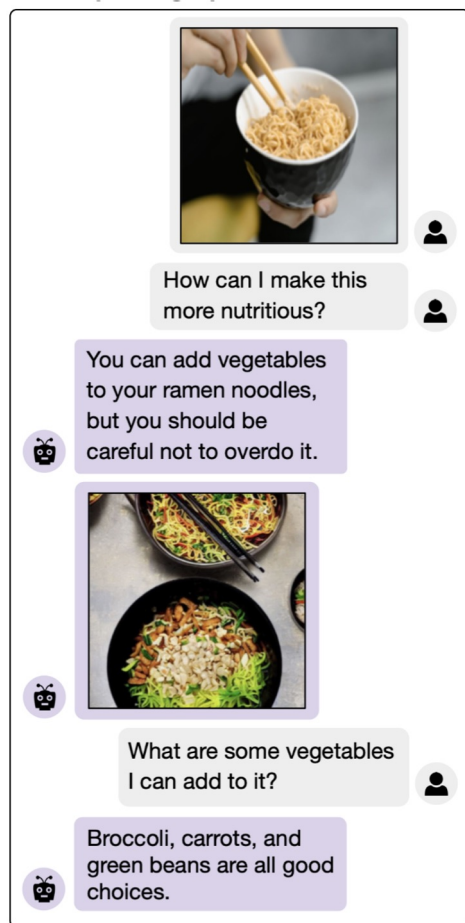
(Greater)  
sensitivity to  
input prompts

Generate long  
and coherent text

World  
knowledge

# Interacting with LLMs for...

## Multimodal dialogue



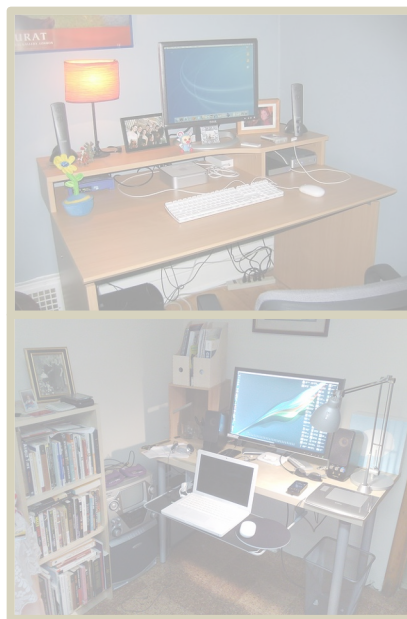
How can I make this more nutritious?

You can add vegetables to your ramen noodles, but you should be careful not to overdo it.

What are some vegetables I can add to it?

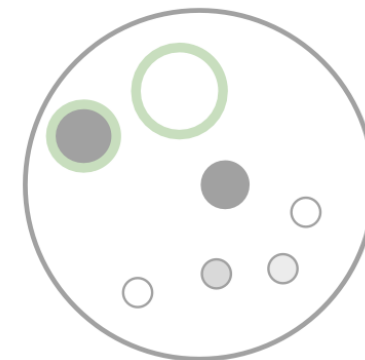
Broccoli, carrots, and green beans are all good choices.

## Question-based retrieval



*"what is next to the computer?"*

## Referential tasks



*"big light dot next to dark dot"*

```
def is_light(n, ctx):  
    return ctx[x, -1] > 0.3  
  
def are_close(x, y, ctx):  
    dist = np.linalg.norm(  
        ctx[x, :-2] - ctx[y, :-2]  
    )  
    return dist < 0.3
```



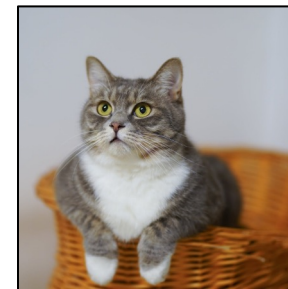
# Multimodal Dialogue

---

Inputting  
Images

Generating  
Images

Dialogue and  
Discourse



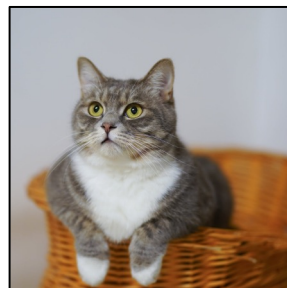
Visual  
Encoder



LLM



Visual  
Decoder





# GILL Generating Images with Large Language Models

Parameter-efficient fusion of existing LLMs and image models; trainable in ~4 GPU days.

The figure displays three chat interface panels illustrating the GILL system's capabilities. Each panel shows a sequence of user prompts, retrieved images, and generated responses.

- Panel 1 (Ramen):** A user prompt asks how to make ramen more nutritious. A retrieved image shows a bowl of ramen. The generated response suggests adding vegetables like broccoli, carrots, and green beans. A second user prompt asks for vegetable suggestions, and the generated response lists broccoli, carrots, and green beans.
- Panel 2 (Tattoo):** A user prompt asks for tattoo ideas for a petite person. A retrieved image shows a tattoo of a fox's head. The user then asks for a sunflower tattoo style. A retrieved image shows a sunflower tattoo on an arm. The generated response suggests a complementary piece.
- Panel 3 (Cupcakes):** A user prompt asks how to publicize cupcakes. A retrieved image shows several cupcakes. The generated response suggests a simple sign with the business name and a picture of the cupcakes. A retrieved image shows a sign that says "T! TUHE BALVEK'S" with a picture of cupcakes.

Legend:  User prompts  Retrieved  Generated

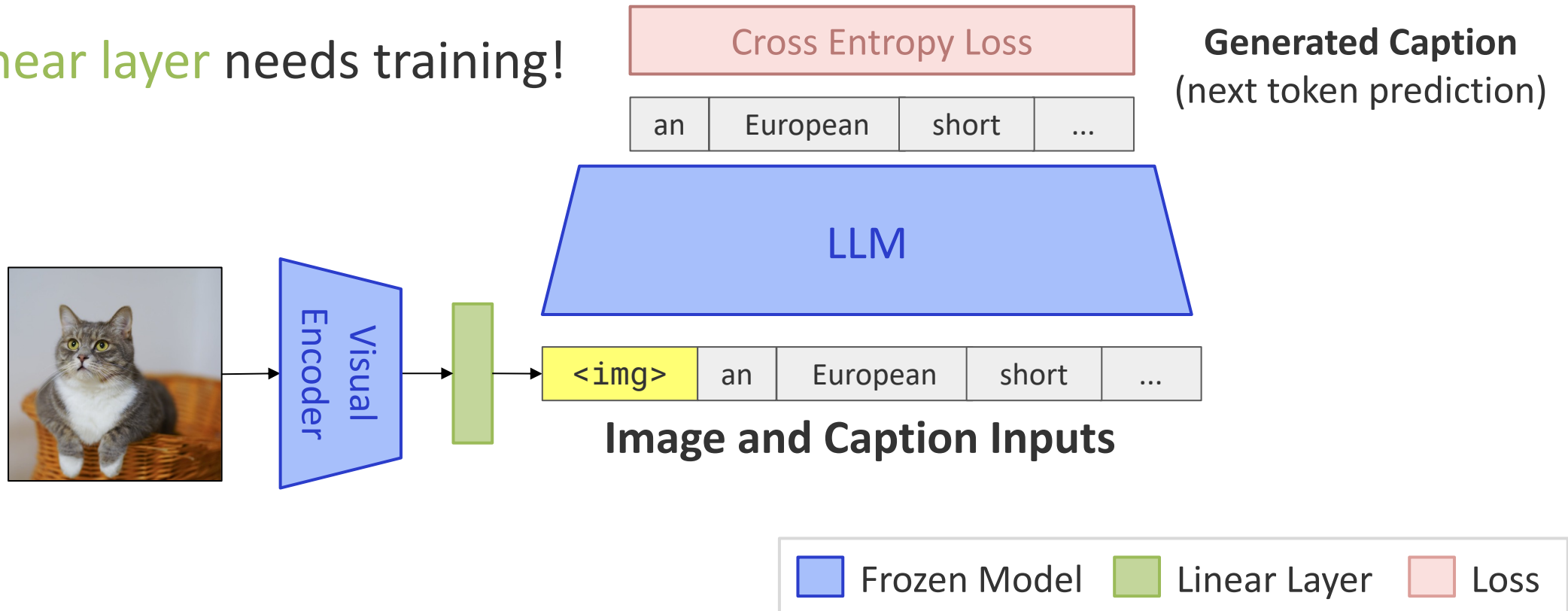


Jing Yu Koh

# Images as Inputs

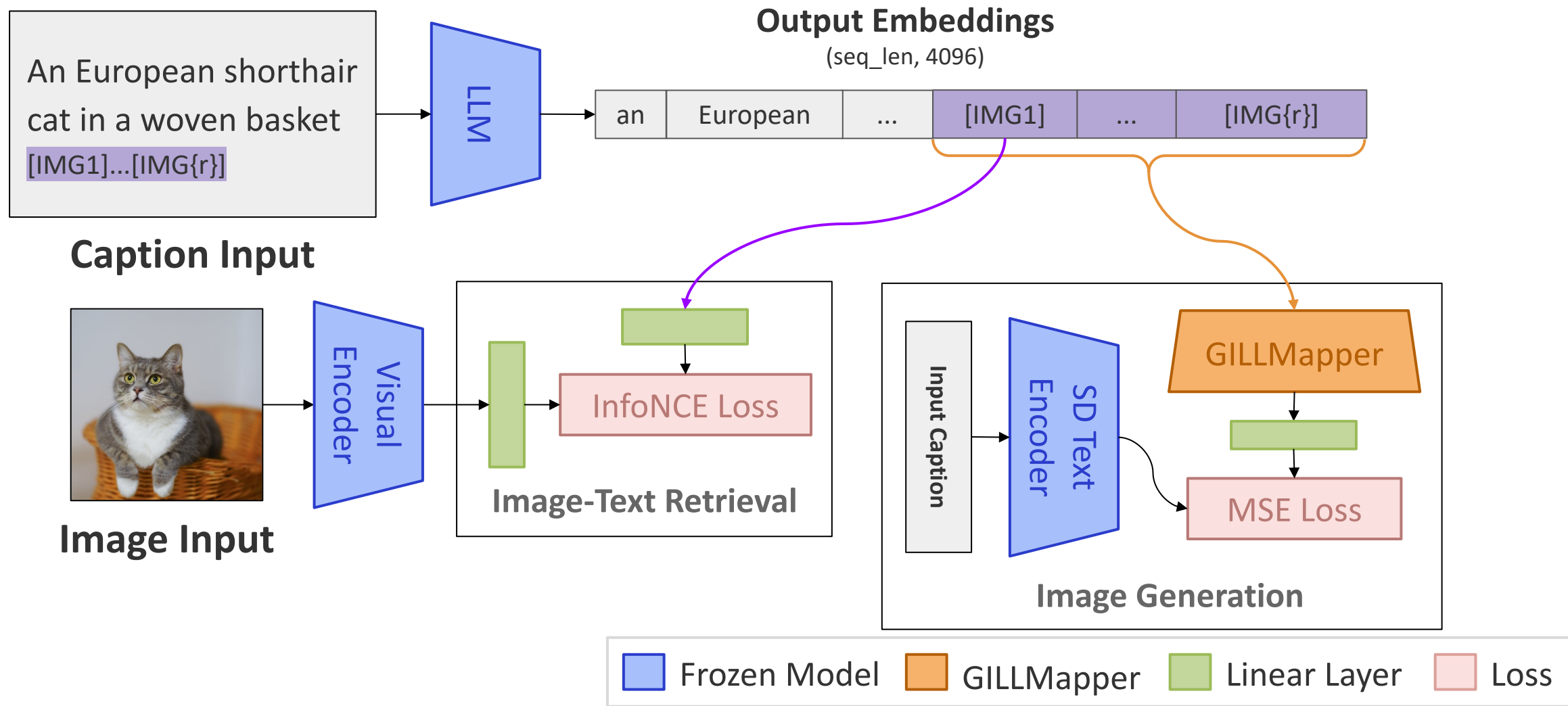
Align *input* representations of an LLM (OPT, Llama2) and *visual encoder outputs (CLIP)* on image captions

Only the **linear layer** needs training!



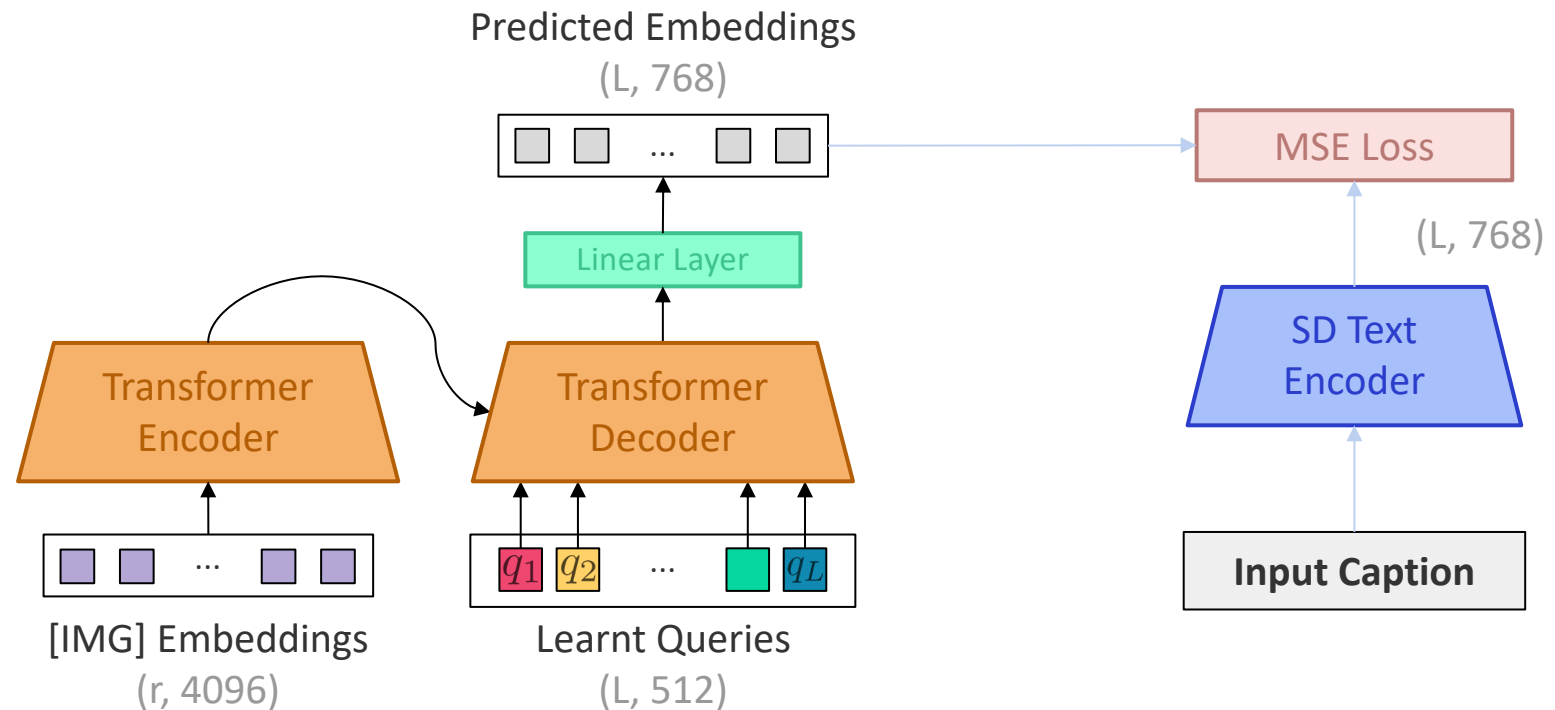
# Images as Outputs

Align **output** representations of an LLM (OPT, Llama2) and **visual models (CLIP, Stable Diffusion)** on image captions



# GILLMapper: An Improved LLM-to-Generator Map

- Previous approaches use linear mappings between LLMs and visual models
- This is insufficient for image generation: decoders require dense information



Multimodal Few-Shot Learning with Frozen Language Models ([Tsimpoukelli et al., 2021](#))

Linearly Mapping from Image to Text Space ([Merullo et al., 2023](#))

Grounding Language Models to Images for Multimodal Inputs and Outputs ([Koh et al., 2023](#))

# Evaluation: Contextual Image Generation

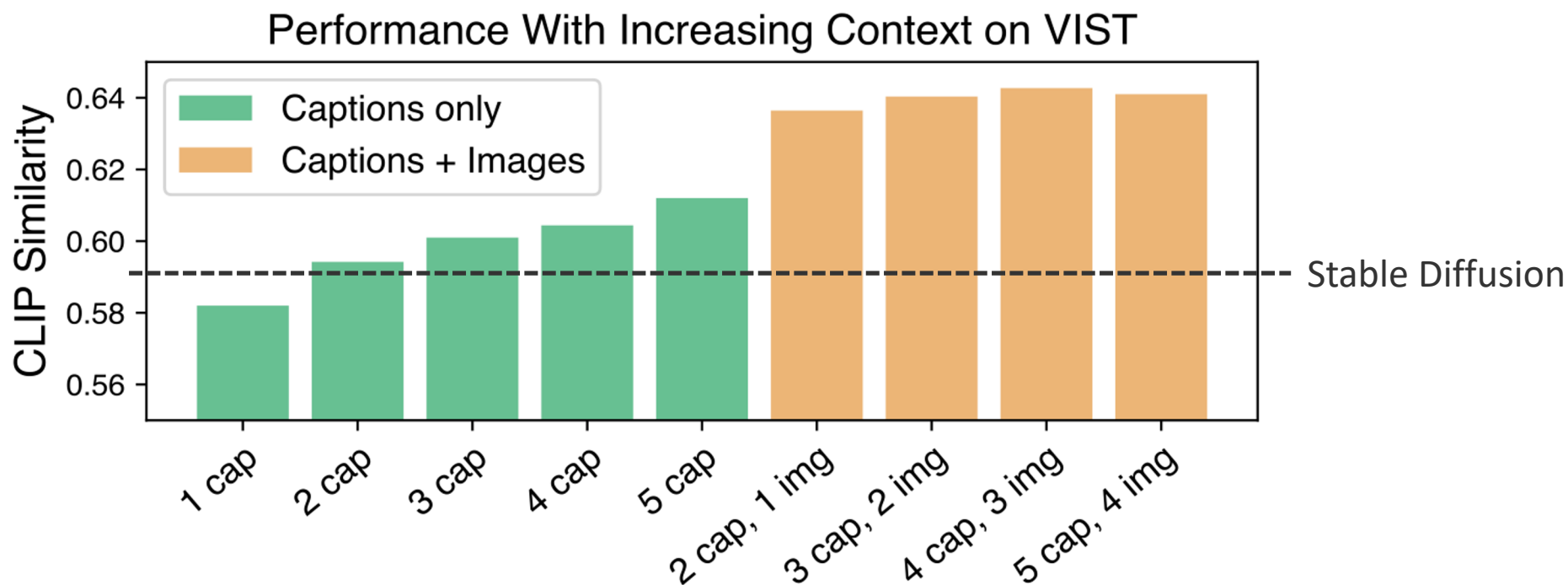
- Given a Visual Story, generate a relevant image
- Need to condition on long, temporally dependent text
- (Optionally) Condition on image inputs interleaved within the text

Once while I was on vacation in this nice brick hotel		I woke up and took my dog Trixie for a walk.		Trixie ran around and enjoyed the fresh air.		We had lots of fun playing fetch together		After a while she got tired and had to take a rest.				
Image and Text Inputs									→	Stable Diffusion	Ours	Groundtruth

# The Effect of Context

Multi-modal context is **worth more** than uni-modal context, producing more relevant generation results.

Our model distills from Stable Diffusion, but outperforms it with multi-modal context.



# GILLMapper: A Stronger LLM-to-Generator Mapping

---

Image generators require **denser** input sequences. Linear mappings are insufficient.

Model	CC3M	VIST
	FID (↓)	CLIP Sim (↑)
Stable Diffusion [43]	<b>13.94</b>	0.598
Ours + Linear	15.50	0.500
Ours + 3-layer MLP	15.33	0.502
Ours + Transformer Encoder	16.30	0.605
Ours + GILLMapper	15.31	<b>0.641</b>



# Qualitative Examples



**Stable Diffusion**

“A dignified beaver wearing glasses, a vest, and colorful neck tie. He stands next to a tall stack of books in a library.”



**Ours**



**Stable Diffusion**

“Snow mountain and tree reflection in the lake”



**Ours**



**Stable Diffusion**

“A drop-top sports car coming around a bend in the road”



**Ours**



**Stable Diffusion**

“a group of penguins in a snowstorm”



**Ours**

# Qualitative Examples

I want to paint a landscape with mountains and a river. Can you help me find some inspiration?



I'm not sure



Can you show me some with horses?

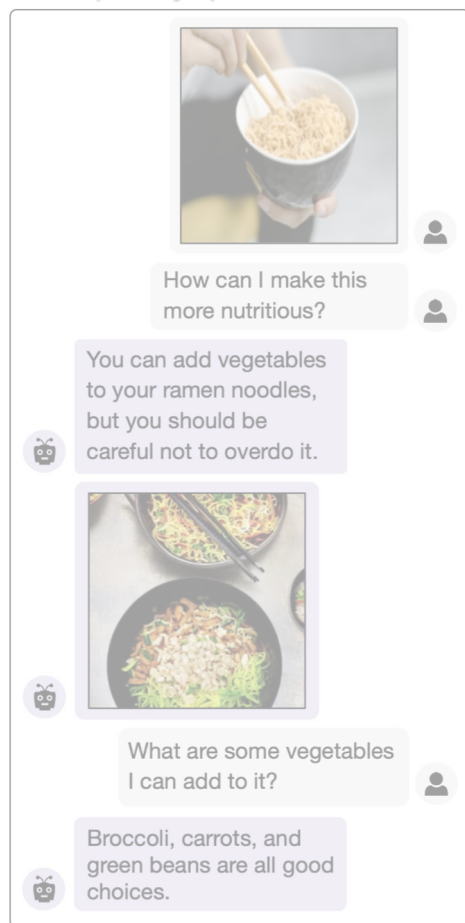


What about something more abstract?



# Interacting with LLMs for...

## Multimodal dialogue



How can I make this more nutritious?

You can add vegetables to your ramen noodles, but you should be careful not to overdo it.

What are some vegetables I can add to it?

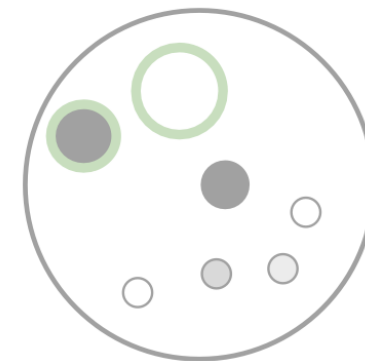
Broccoli, carrots, and green beans are all good choices.

## Question-based retrieval



*"what is next to the computer?"*

## Referential tasks



*"big light dot next to dark dot"*

```
def is_light(n, ctx):  
    return ctx[x, -1] > 0.3  
  
def are_close(x, y, ctx):  
    dist = np.linalg.norm(  
        ctx[x, :-2] - ctx[y, :-2]  
    )  
    return dist < 0.3
```

# Question-Based Retrieval

Tracking  
User Goals

Reasoning About  
Informativity



Visual QA  
Model

Bayesian Inference

Generating  
Questions

*What is next to  
the computer?*

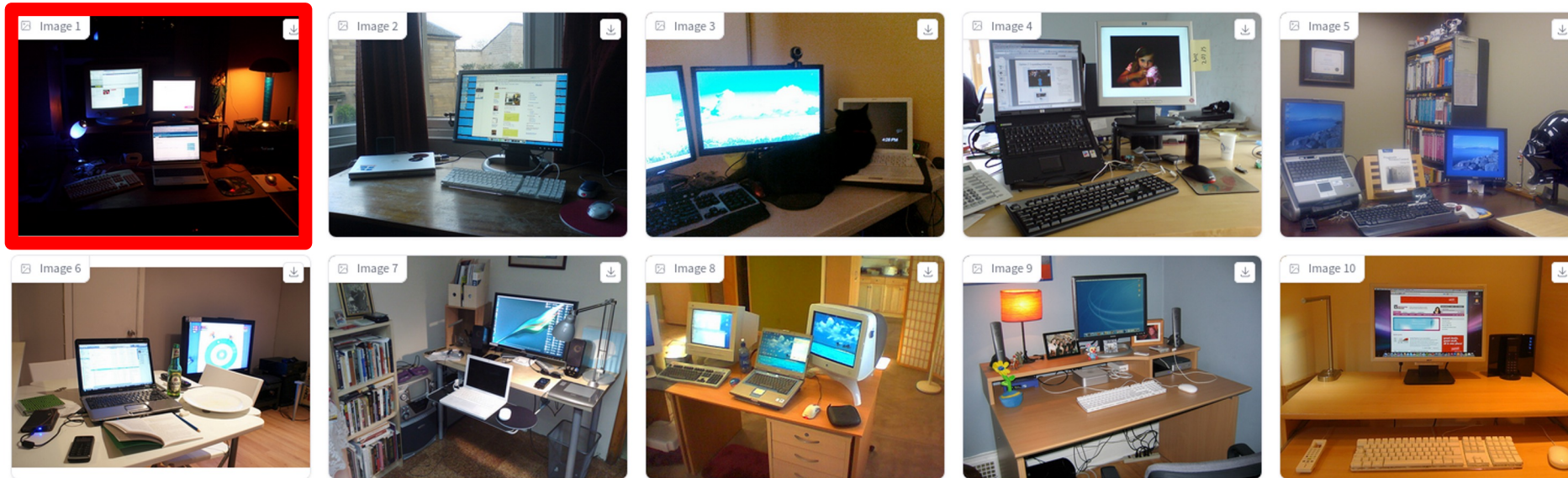
LLM


Visual QA  
Model

Information Gain Planning



# Question-Based Retrieval



 : What is next to the computers?

 : Lamp

 : How many computers are there?

 : 3

 : Guess – Image 1



Sedrick Keh

# What Makes an Informative Question?

---



*Is there a computer in the image?* X

*Is there a cat in the image?* X

*What is next to the computer?* ✓

# What Makes an Informative Question?

---



*What is next to  
the computer?*



# What Makes an Informative Question?

Images,  $i$   
Belief state:  $P(i)$



Question,  $q$   
*What is next to  
the computer?*

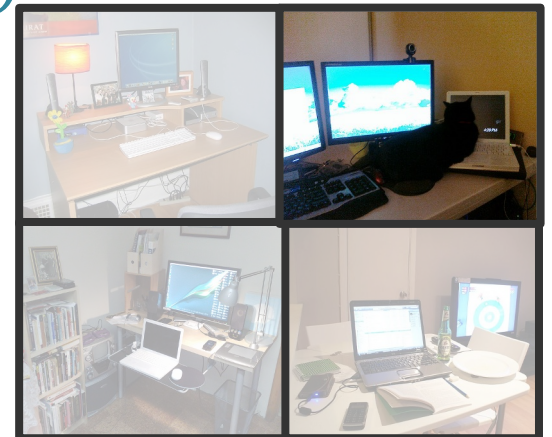
Answers,  $a$   
"lamp"

Visual QA  
Model  
 $P(a | q, i)$

"cat"



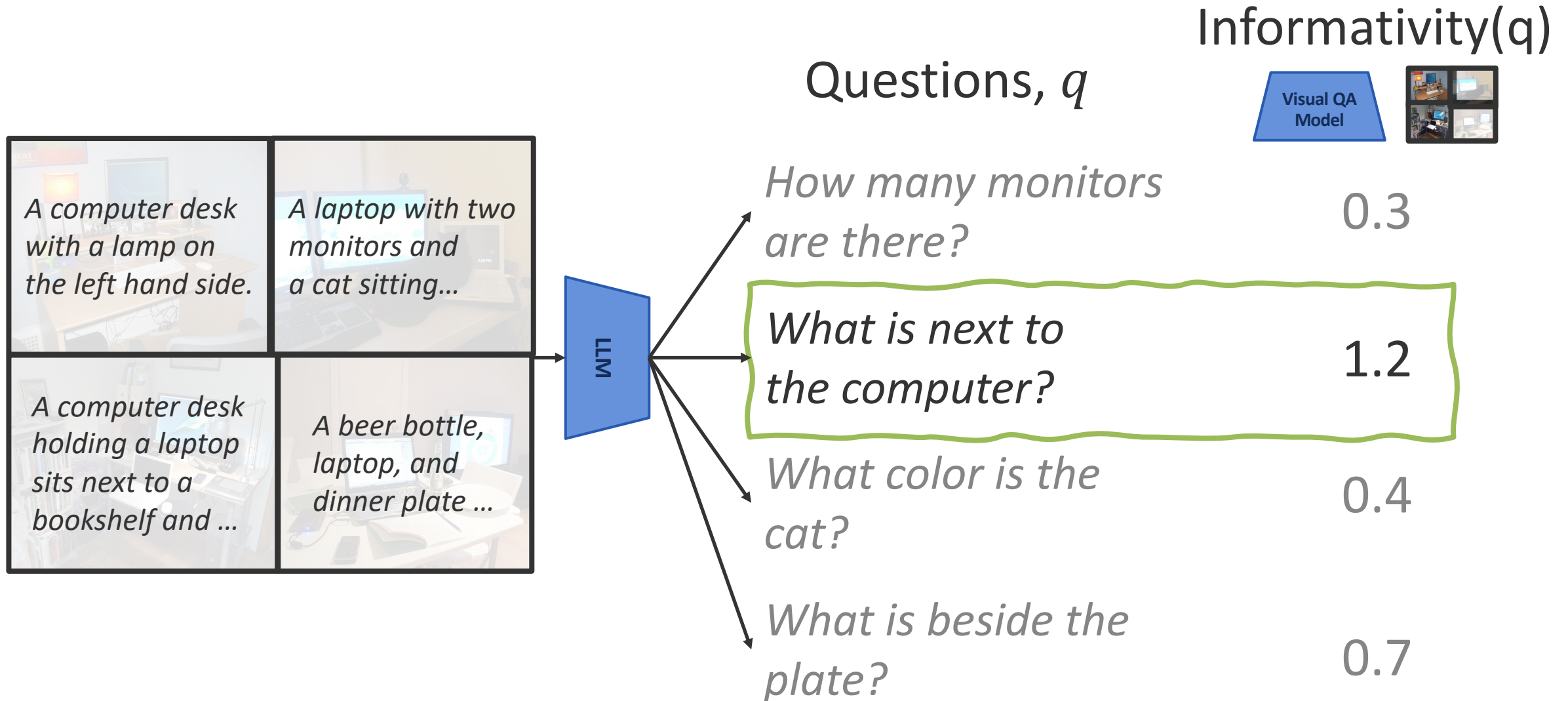
$P(i | q, a)$   
(by Bayes' rule)



$$\text{Informativity}(q) = H[i] - \mathbb{E}_{p(a|q)} H[i|q, a]$$

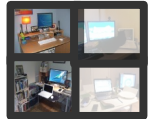
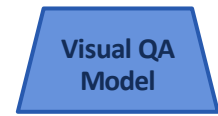


# Generating Informative Questions



# Generating Informative Questions

Informativity( $q$ )



**GPT-4 Prompt**

**You are tasked to produce reasonable questions from a given caption.**  
The questions you ask must be answerable only using visual information.  
As such, never ask questions that involve exact measurement such as 'How tall', 'How big', or 'How far', since these cannot be easily inferred from just looking at an object.  
... (More detailed examples here.) ...  
**For each caption, please generate 3-5 reasonable questions.**

A computer desk with a lamp on the left hand side.

A laptop monitor and a cat sitting on the desk.

A computer desk holding a laptop sits next to a bookshelf and ...

A bee on a laptop next to a dinner plate.

... questions,  $q$

any monitors  
re?

next to  
computer?

color is the

beside the  
plate?

0.3

1.2

0.4

0.7

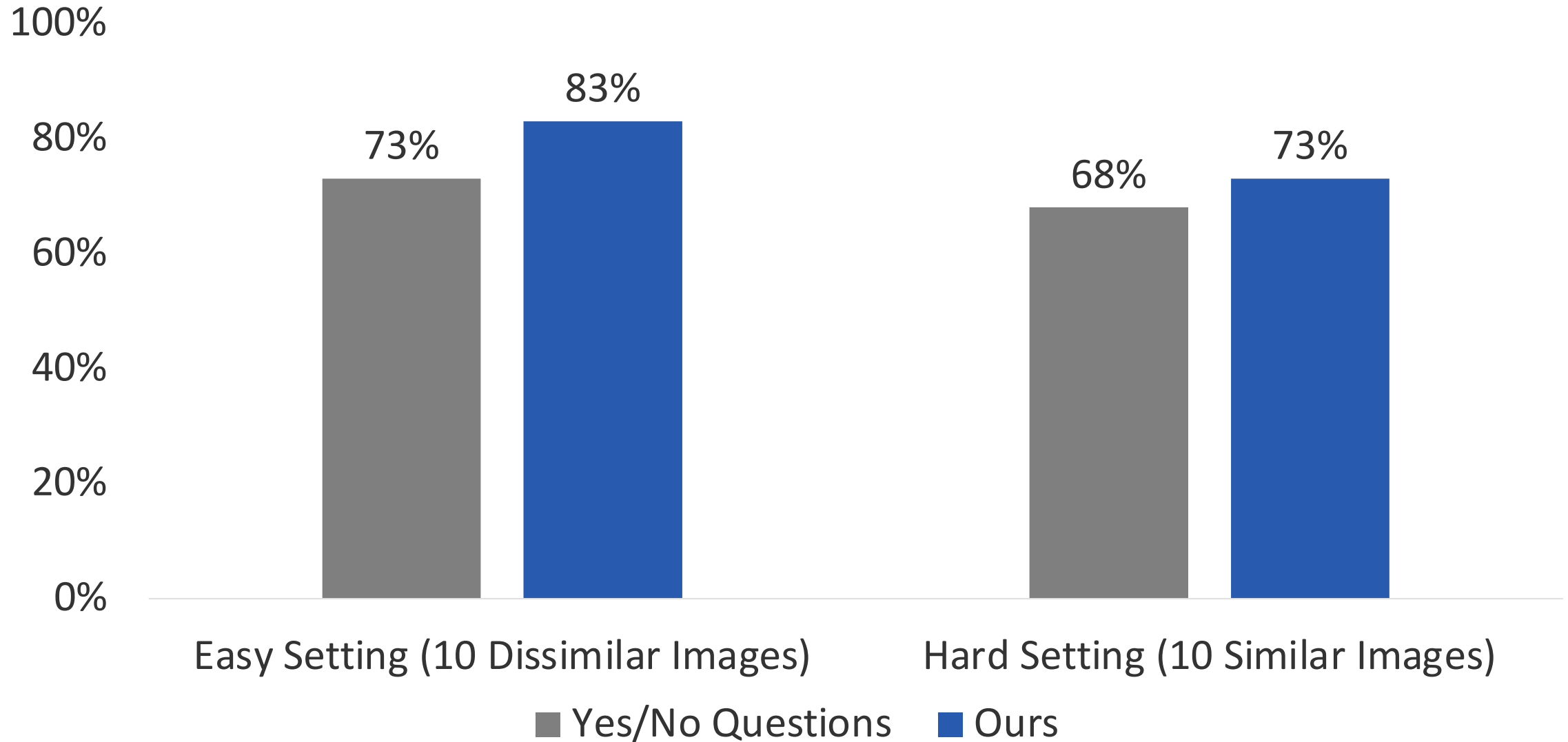
# Experiments

---

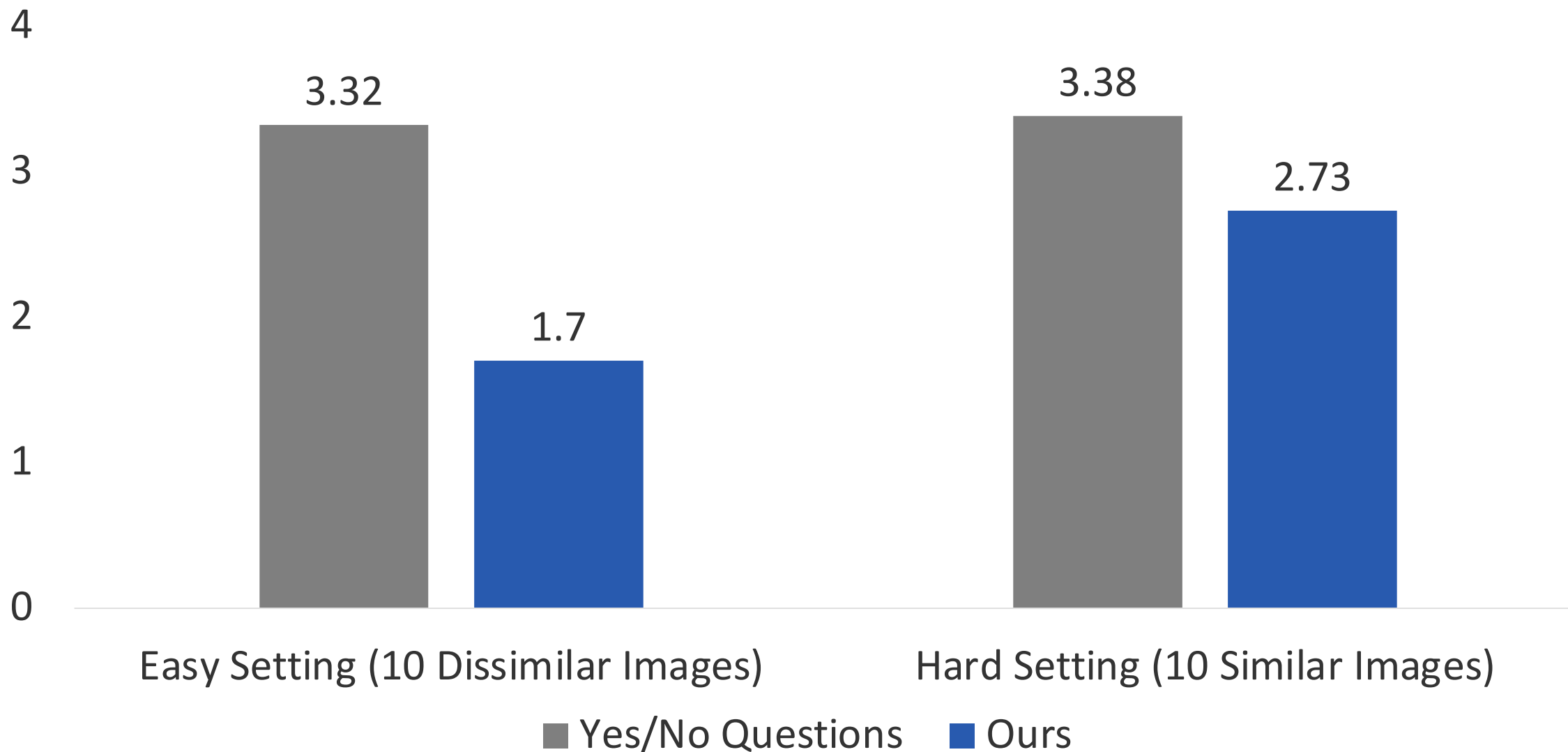
- ▶ Play with people using sets of 10 MSCOCO images.
- ▶ Compare against a Yes/No question method from past work [White et al. 2021].
- ▶ Evaluate accuracy and number of questions asked.
- ▶ Also need to avoid presupposition errors in the VQA models – see the paper!

# Results: Accuracy ↑

---

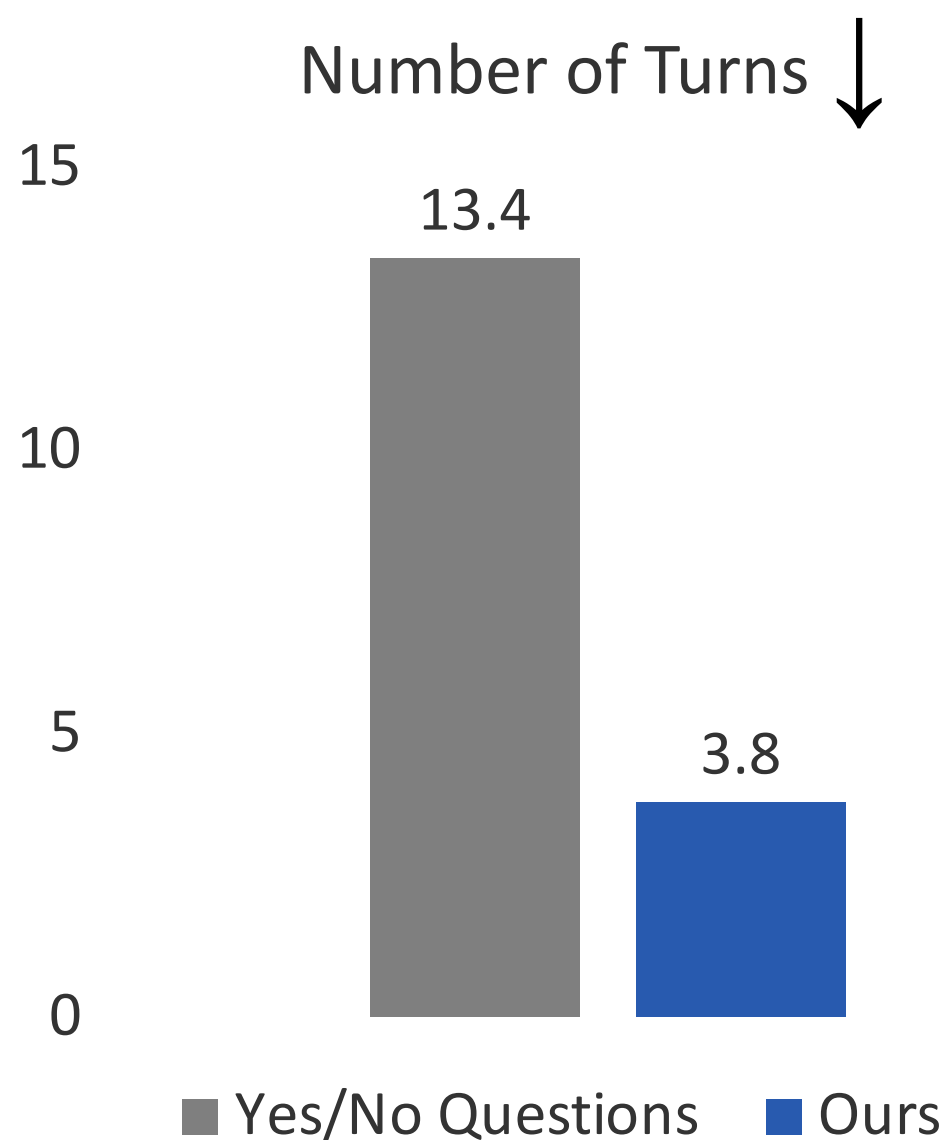
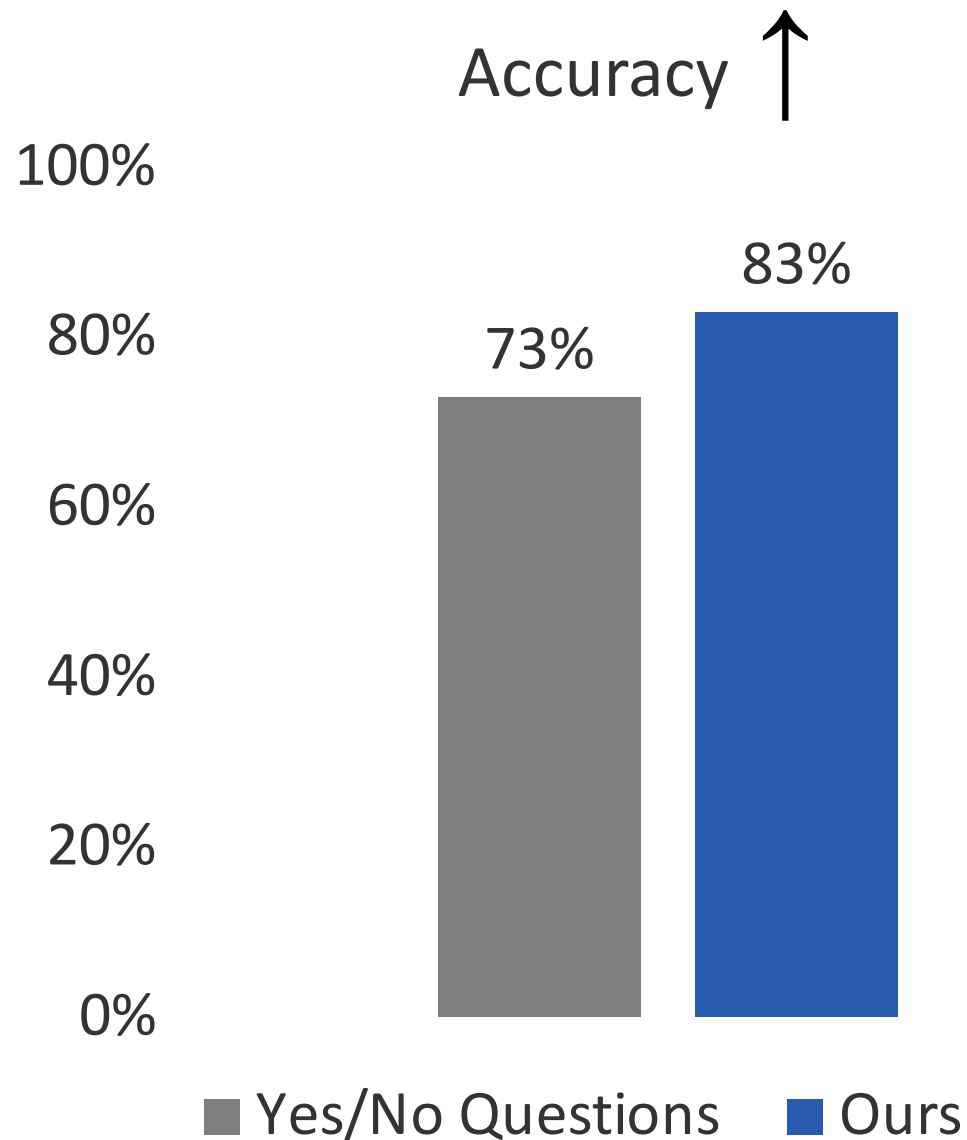


# Results: Number of Questions ↓



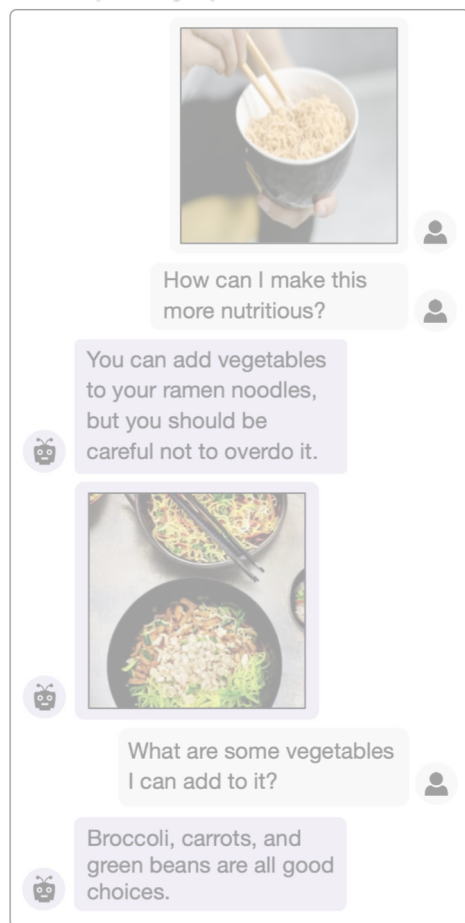
# With 100 Images (Automatic Evals)

---



# Interacting with LLMs for...

## Multimodal dialogue



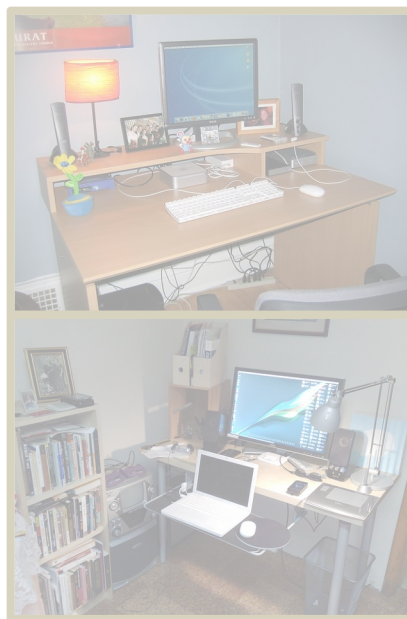
How can I make this more nutritious?

You can add vegetables to your ramen noodles, but you should be careful not to overdo it.

What are some vegetables I can add to it?

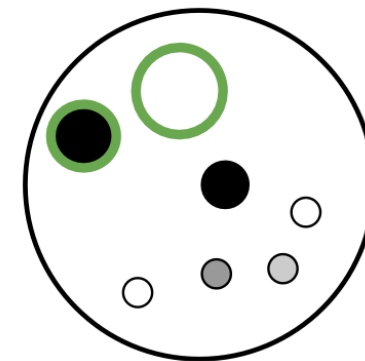
Broccoli, carrots, and green beans are all good choices.

## Question-based retrieval



*"what is next to the computer?"*

## Referential tasks



*"big light dot next to dark dot"*

```
def is_light(n, ctx):  
    return ctx[x, -1] > 0.3
```

```
def are_close(x, y, ctx):  
    dist = np.linalg.norm(  
        ctx[x, :-2] - ctx[y, :-2]  
    )  
    return dist < 0.3
```

# Referential Dialogue and QA

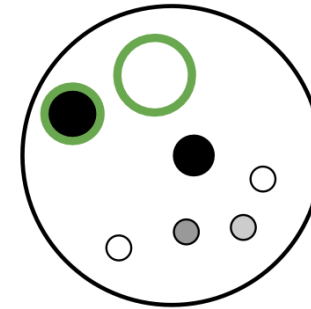
Semantic  
Parsing

Code  
Generation

```
is_big(x) and is_light(x)  
and is_dark(y) and  
are_close(x, y)
```

LLM

*"big light dot next  
to the dark dot"*



Python  
Interpreter

```
def is_light(n, ctx):  
    return ctx[x, -1] > 0.3  
  
def are_close(x, y, ctx):  
    dist = np.linalg.norm(  
        ctx[x, :-2]-ctx[y, :-2]  
    )  
    return dist < 0.3
```





Yihan Cao Shuyi Chen Ryan Liu Zora Wang

# Code for Table QA

**Question:**

Who is more likely to have cancer, *the elder* or *the young*?

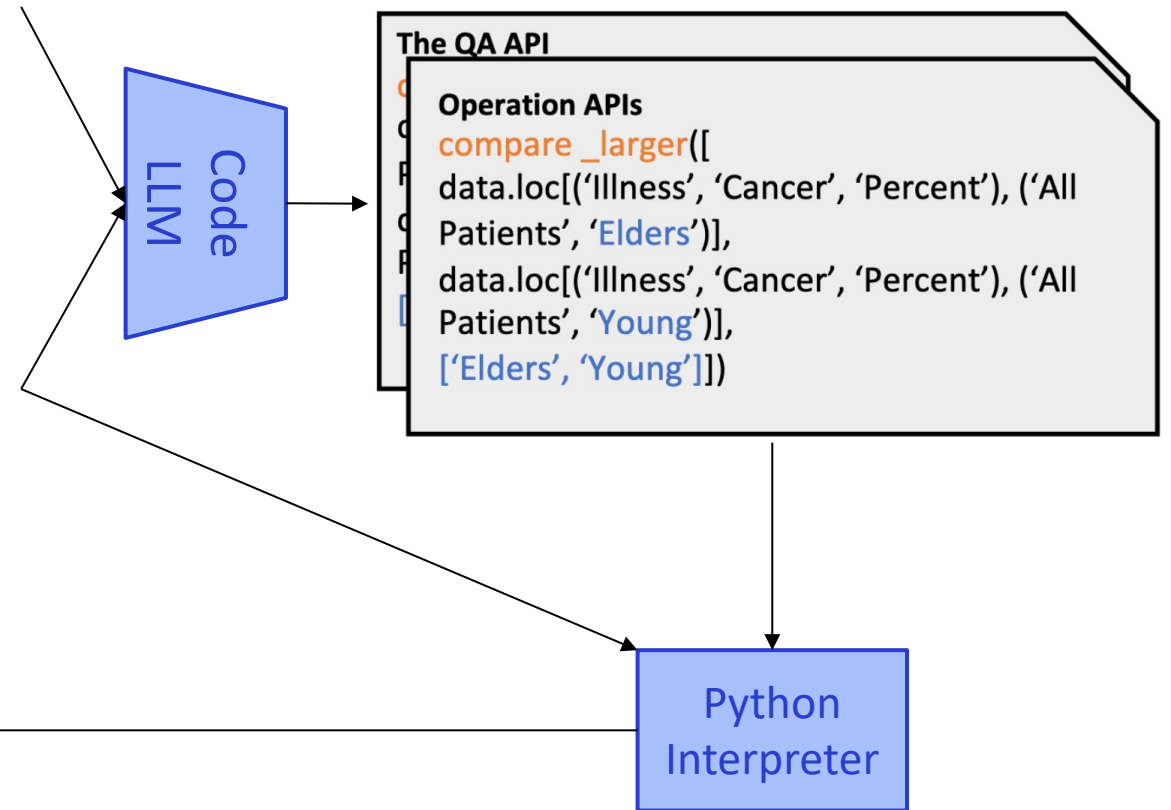
**Table:**

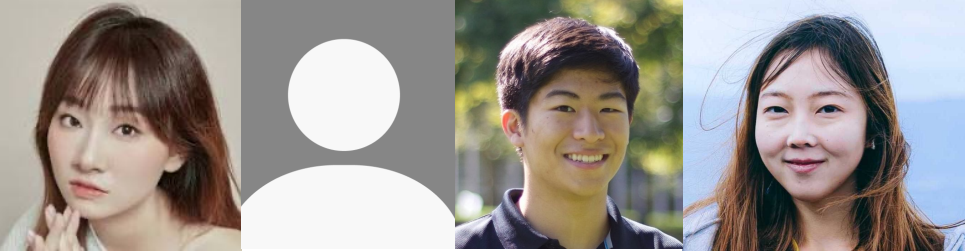
**Title:** Number and percentage of people who are interviewed who have or have had illnesses.

Illnese	Cold		Cancer	
	total	percent	total	percent
All patients	10,000	2.5%	200	0.3%
Elders	7,400	3.5%	126	0.9%
Young	2,600	1.5%	74	0.2%

**Answer:**

Elder





# Code for Table QA

Yihan Cao   Shuyi Chen   Ryan Liu   Zora Wang

Python gives a unified representation across varied table formats and datasets

<b>Dataset</b>	<b>HiTab</b>	<b>Spider</b>	<b>AIT-QA</b>	<b>WikiTQ</b>
Baseline	MAPO 40.7	DIN-SQL 61.5	RCI 51.8	BINDER <b>54.8</b>
Codex	59.6	61.2	77.8	41.7
<b>w/ API (Ours)</b>	<b>69.3</b>	<b>63.8</b>	<b>78.0</b>	42.4

# Grounded Collaborative Dialogue

---



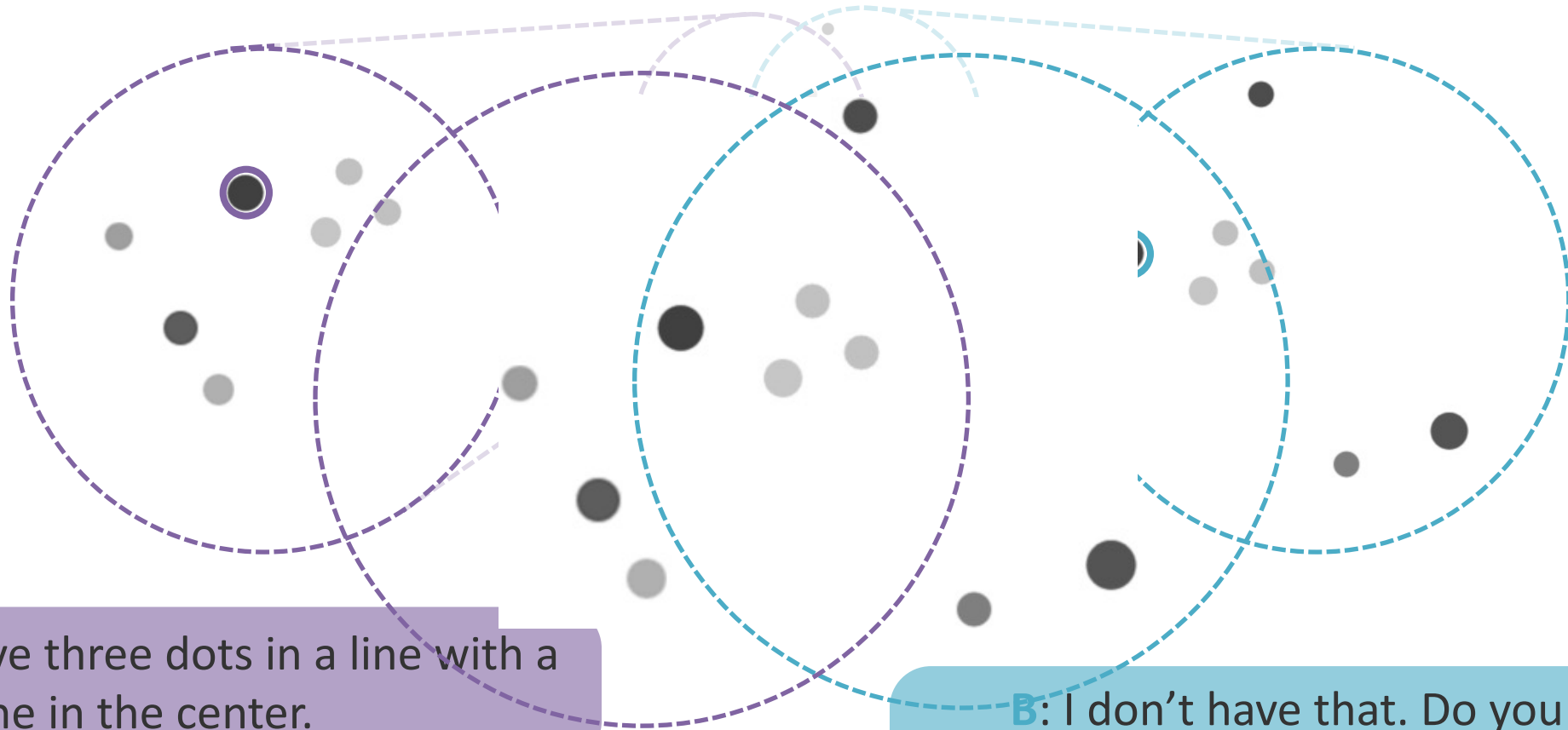
**A:** I have three dots in a line with a dark one in the center.

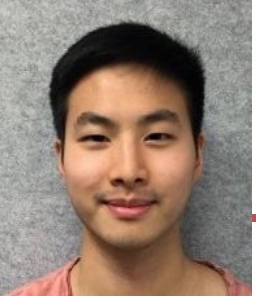
**A:** Is there a large black dot to the left of the three grey dots?



**B:** I don't have that. Do you have a cluster of three grey dots in a triangle?

**B:** Yes, let's select the black one.

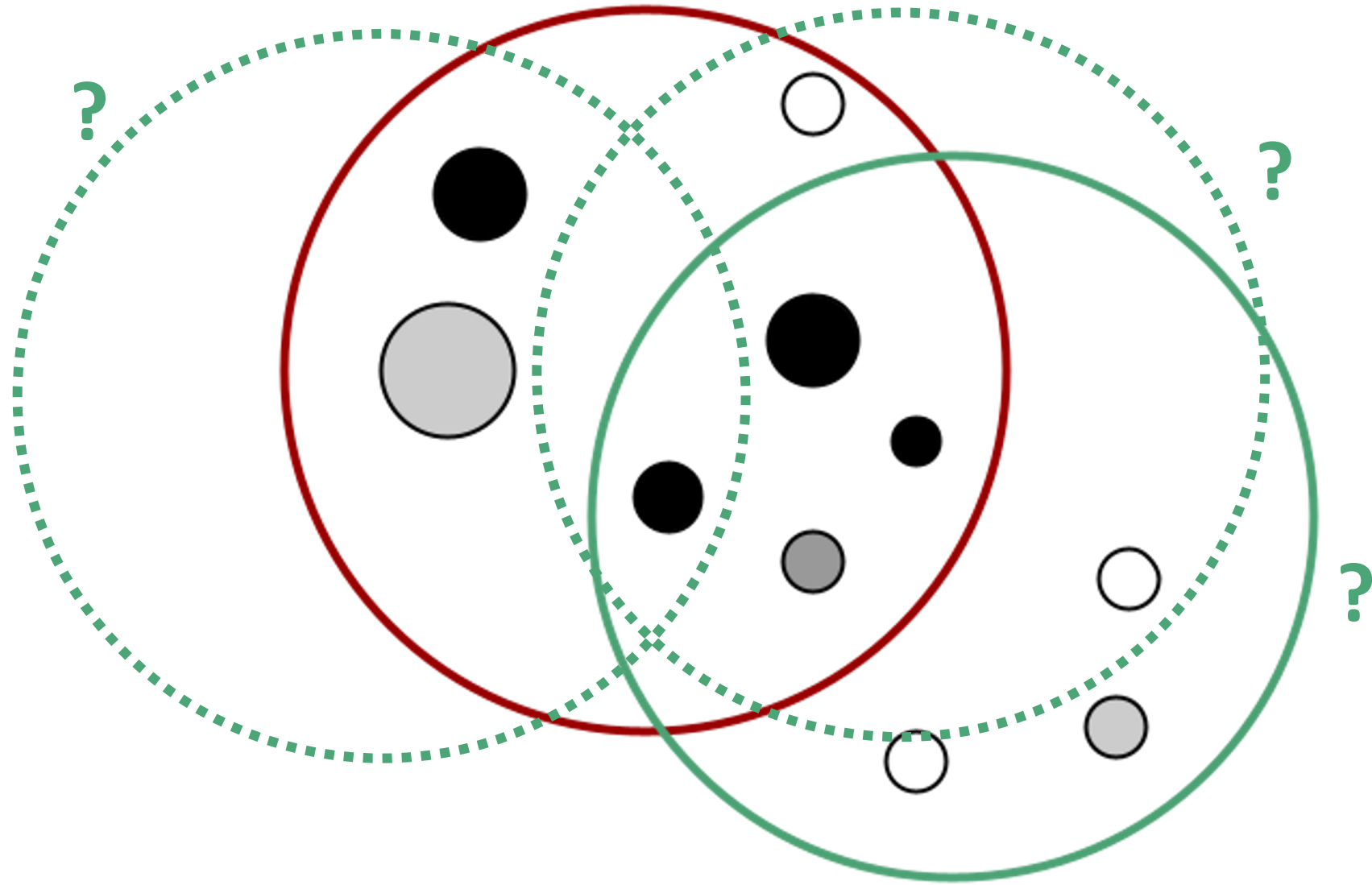




Justin Chiu

# Beliefs About What's In Common

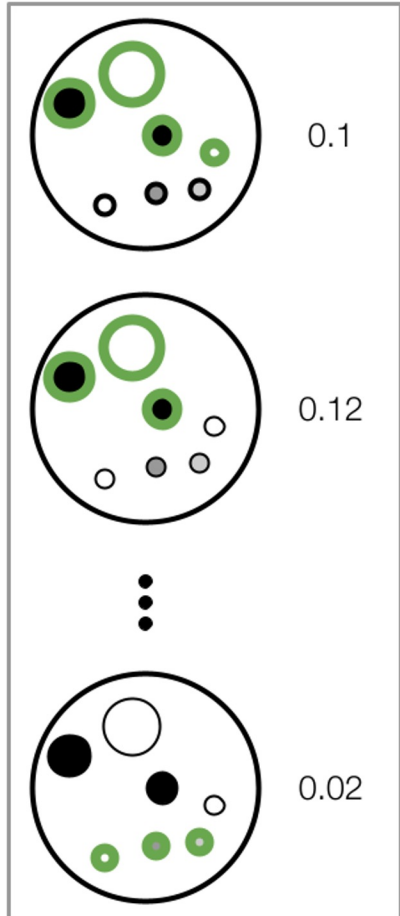
---



# Method Overview

---

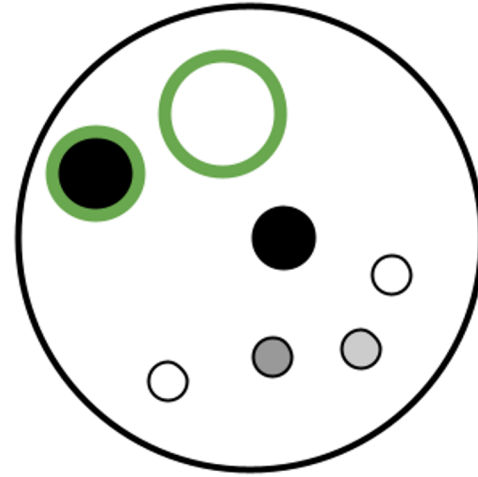
Previous belief state  $\mathbf{p}(\mathbf{z})$



# Reading

---

Dots mentioned  $p(x|u)$



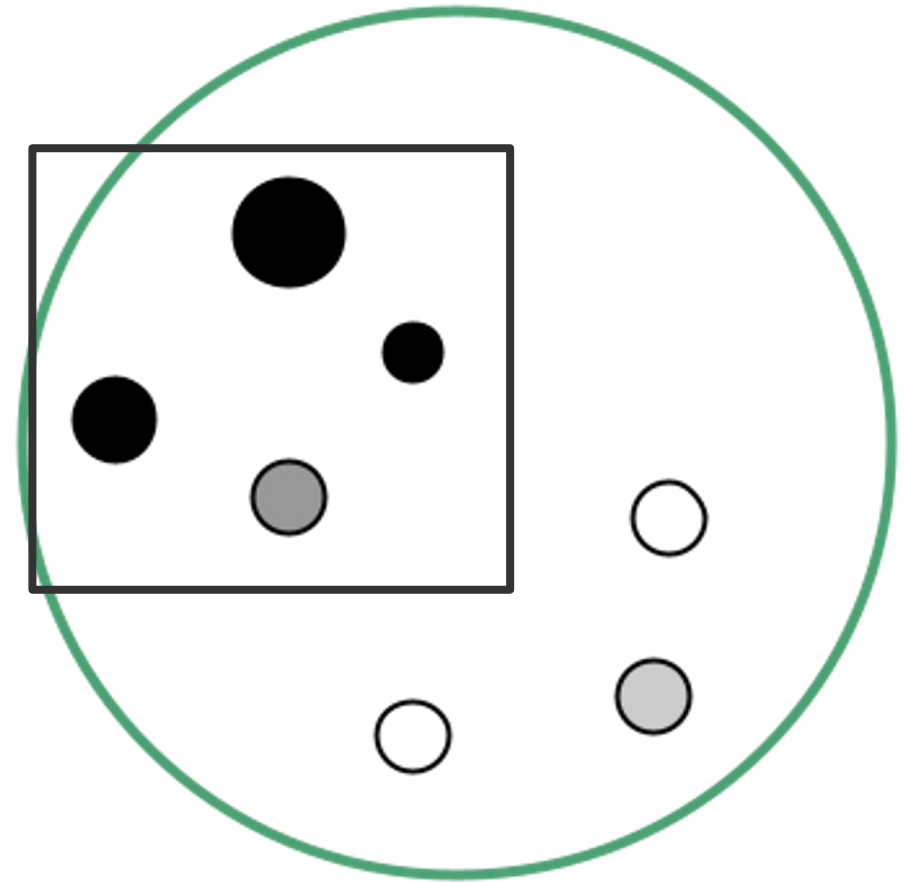
Read

Partner utterance  $u$ : "Is there a big light dot next to a big dark one?"

# Reading via a Code LLM

---

```
from perceptual_library import is_small, ...  
dot1, dot2, dot3, ... = get_dots()
```



# Grounding function library

---

- ▶ Functions are predicates over dots
- ▶ Manually designed for OneCommon

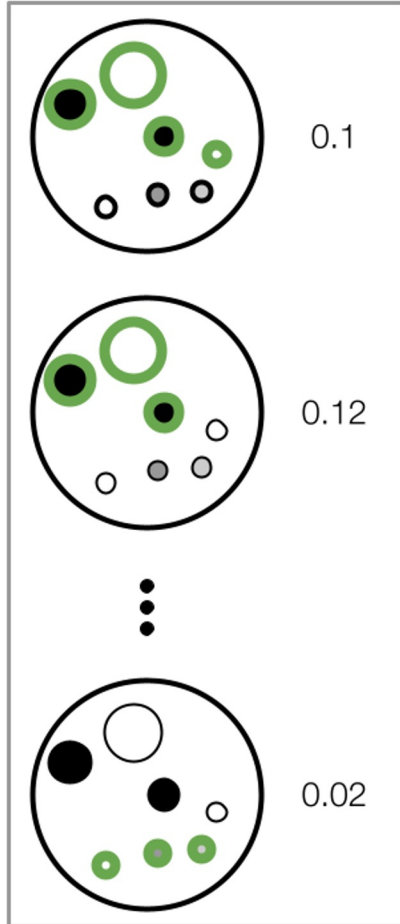
```
def is_light(n, ctx):  
    return ctx[x, -1] > 0.3
```

```
def are_close(x, y, ctx):  
    dist = np.linalg.norm(  
        ctx[x, :-2] - ctx[y, :-2]  
    )  
    return dist < 0.3
```



# Belief update

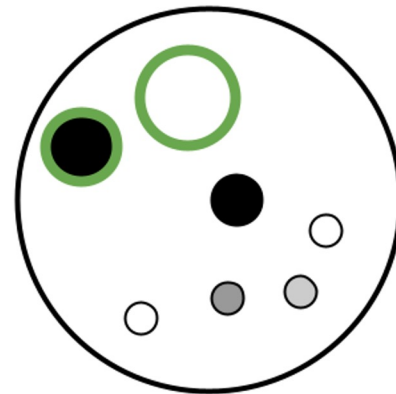
Previous belief state  $p(\mathbf{z})$



Belief update



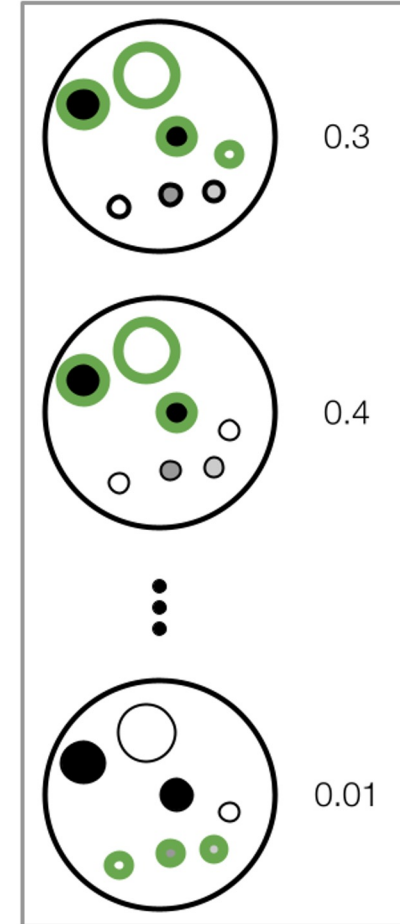
Dots mentioned  $p(\mathbf{x}|\mathbf{u})$



Read



Belief state  $p(\mathbf{z}|\mathbf{u})$



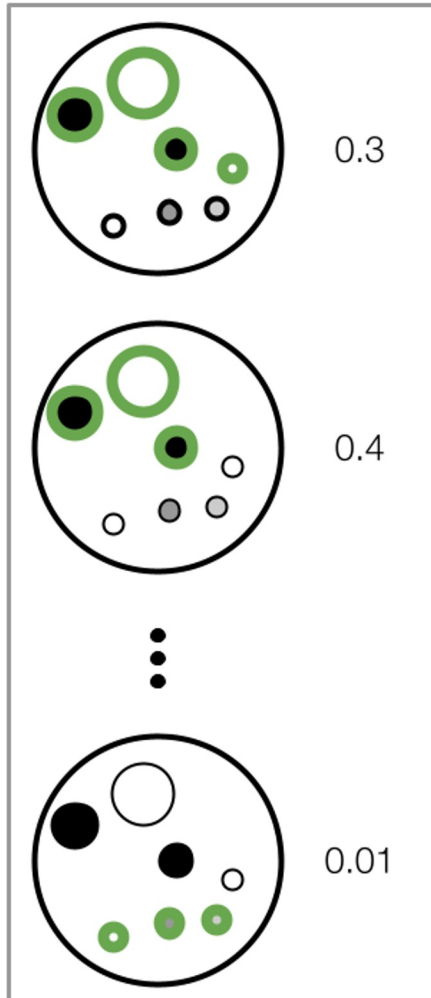
Partner utterance  $\mathbf{u}$ : "Is there a big light dot next to a big dark one?"

# Informative questions: Expected information gain

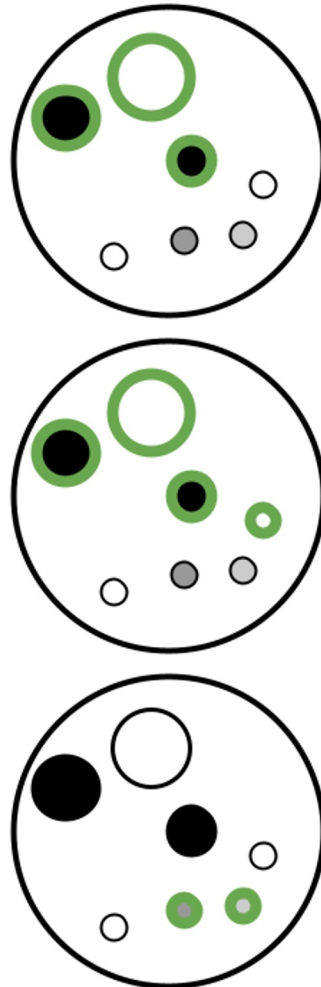
---

Optimize over plans  $y$

Belief state  $p(\mathbf{z}|\mathbf{u})$



Dots to ask about  $y$



Expected information gain

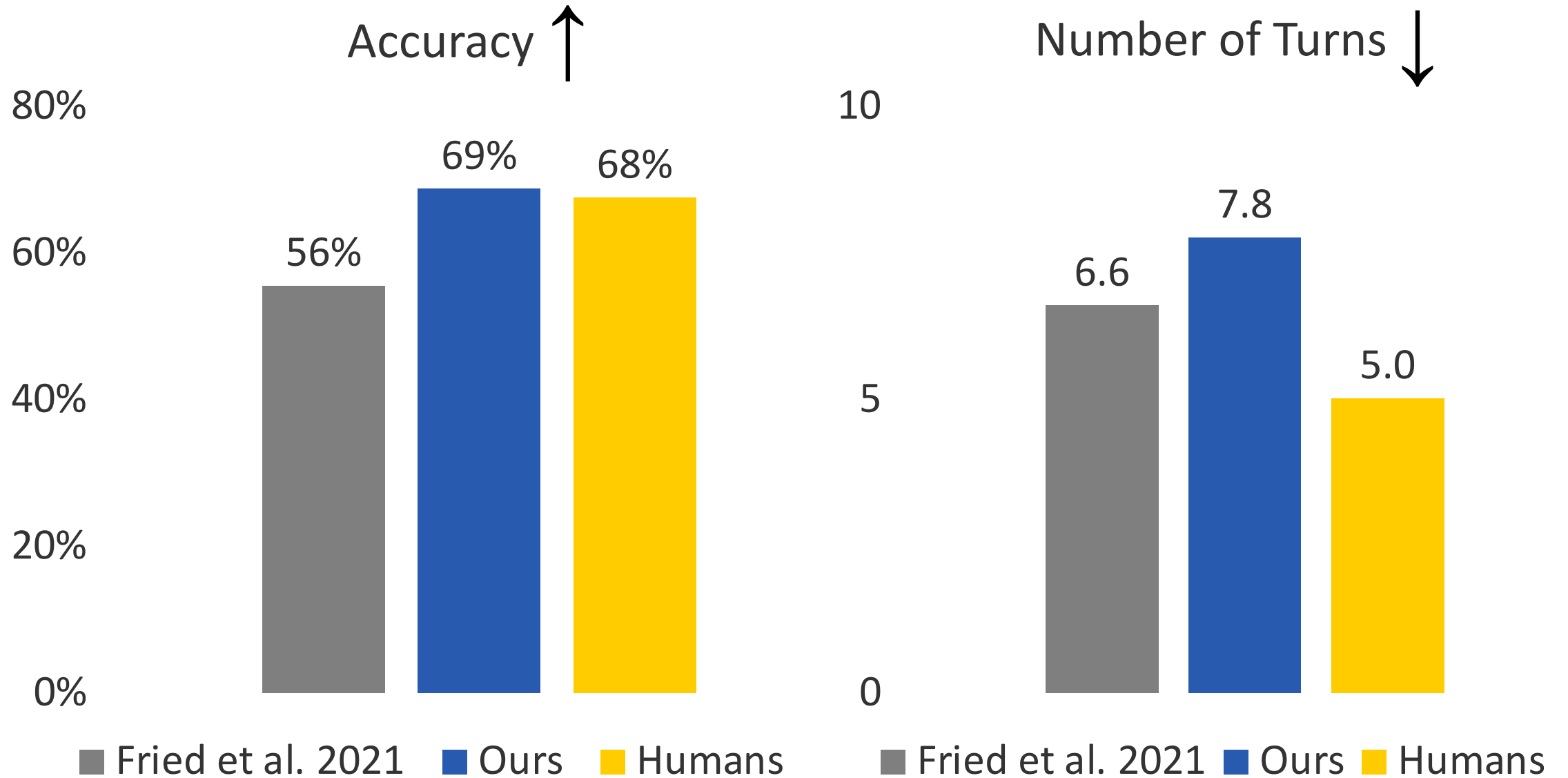
# Evaluation

---

- ▶ Play the game with humans recruited on Mechanical Turk; evaluate success rate.
- ▶ Agents compared:
  - ▷ LSTM with Neural CRF Reference Resolver (Fried, Chiu, Klein, 2021)
  - ▷ Ours
  - ▷ Humans

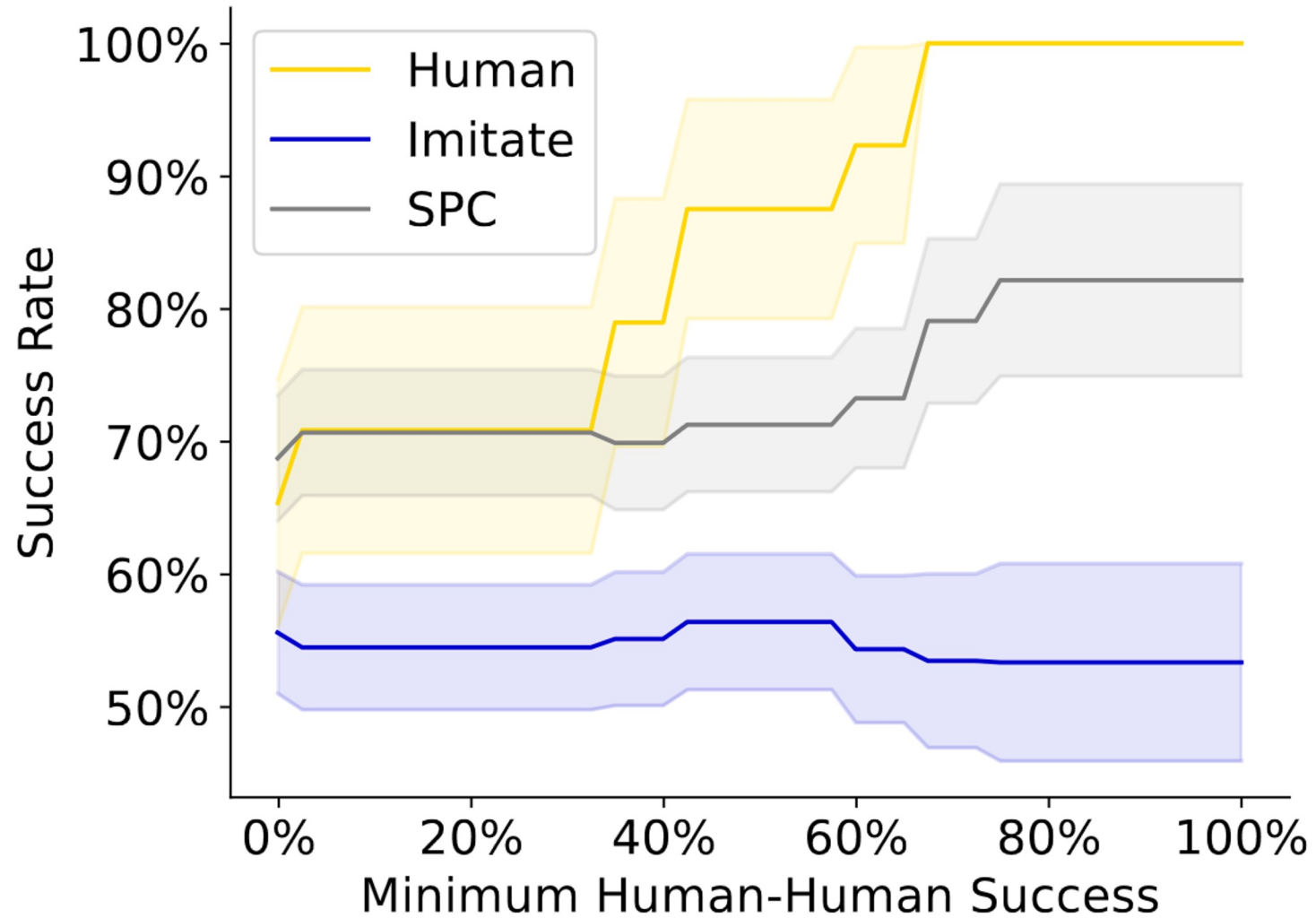
# Results

---



# Results: Human evaluation

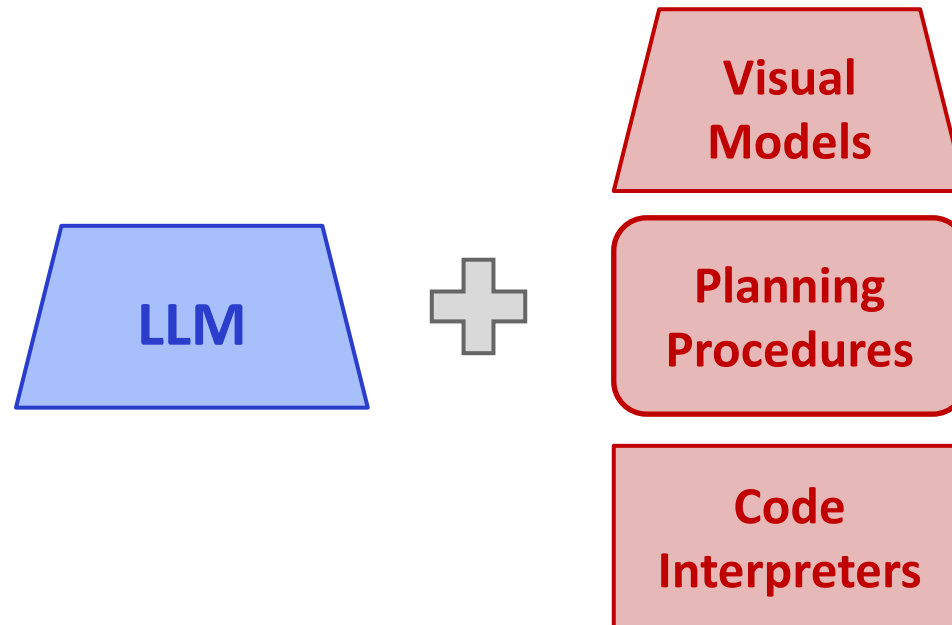
---

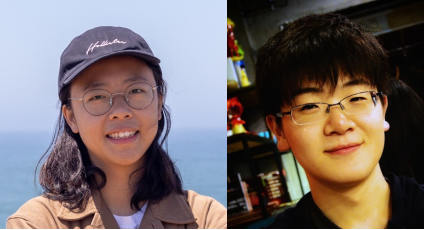


# Takeaways

---

- ▶ LLMs model language use, but need to be contextualized!
- ▶ LLMs are useful building blocks in modular systems.





# Challenge Environments: WebArena

Shuyan  
Zhou

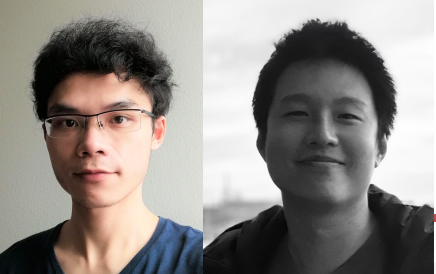
Frank  
Xu

*"Tell me the status of my latest order and when will it arrive"*

```
(web) [~/workshop]$ python web_agent.py
What can I do for you?
User Intent: Tell me the status of my latest order and when will it arrive
Start completing the task ...
```

The screenshot shows a web browser window with the URL `metis.lti.cs.cmu.edu:7770`. The page title is "One Stop Market". The browser's address bar shows "Not Secure" and "Incognito" mode. The website has a search bar with the text "Search entire store here..." and a shopping cart icon. Below the search bar, there are several category links: "Beauty & Personal Care", "Sports & Outdoors", "Clothing, Shoes & Jewelry", "Home & Kitchen", "Office Products", "Tools & Home Improvement", "Health & Household", "Patio, Lawn & Garden", "Electronics", "Cell Phones & Accessories", "Video Games", and "Grocery & Gourmet Food". The main content area is titled "One Stop Market" and "Product Showcases". There are five product images: a large brown container, a pack of Energy drinks, a bag of Elmwood Inn Orange Vanilla Espresso, a pile of popcorn, and a tub of So Delicious Coco Whip.



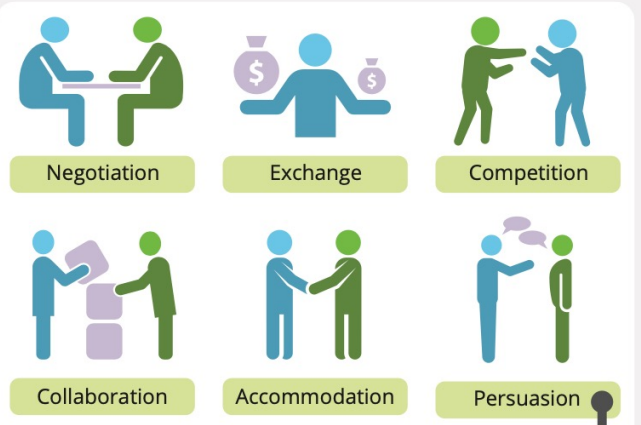


Xuhui  
Zhou

Hao  
Zhu

# Challenge Environments: Sotopia

## Sampling scenarios and social goals



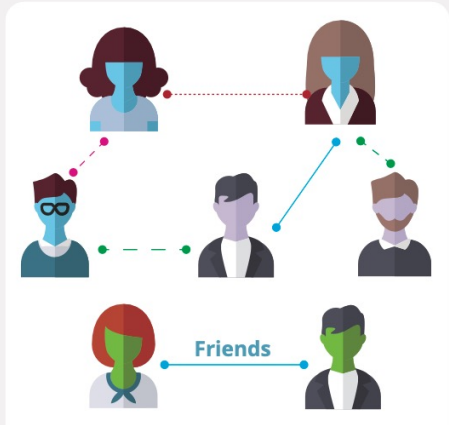
Scenarios cover a large range of social interaction types

**Scenario**  
Two friends are camping in the wilderness and the temperature drops significantly at night

🎯 **Goal (for Agent 1):** Keep the one blanket you have just for yourself

🎯 **Goal (for Agent 2):** Convince your friend to share the blanket with you

## Sampling characters



Characters cover a wide range of profiles and relationships.

**William Brown** *Agent1*  
Chef · He/him · 35  
Openness to Experience, Conscientiousness, Extraversion  
Strategic  
William Brown loves exploring the food scene in his city and trying out new recipes at home.

**Mia Davis** *Agent2*  
High School Principal · She/her · 50  
Extraversion, Neuroticism  
Decisive  
Mia Davis has two cats.

🔒 Part of a rebellious punk rock band in her youth

## Simulating interactions

It's getting really cold. Any chance I can have your blanket?

hmmm, but I am cold and I think I need this blanket more...

Well, can we share the blanket then? It could make both of us warmer!

I am not really comfortable with staying that close to you, sorry.

I see, I guess in that case I will just layer more clothes then 😞

Put more clothes on and move away from William. (Interaction ends)

**SOTOPIA-EVAL** ⭐⭐⭐⭐

Mia did not achieve her social goals in the end, and their relationship seems to be worse ...



# Collaborators

---



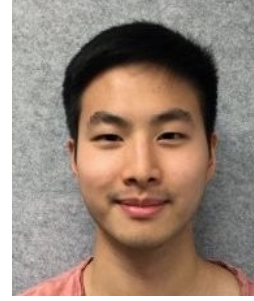
Yihan  
Cao



Derek  
Chen



Shuyi  
Chen



Justin  
Chiu



Sedrick  
Keh



Jing Yu  
Koh



Ryan  
Liu



Sasha  
Rush



Russ  
Salakhutdinov



Saujas  
Vaduguru



Zora  
Wang



Wenting  
Zhao

# Thanks!

dfried@cs.cmu.edu  
<http://dpfried.github.io>

FROMAGE: <https://jykoh.com/fromage>

GILL: <https://jykoh.com/gill>

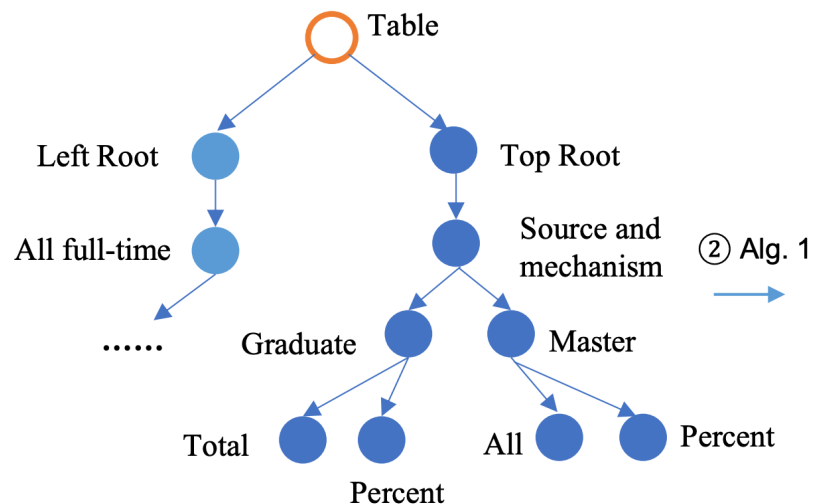
# Other Projects

# Backup Slides

# Code for Table Question Answering

Source and mechanism	All full-time graduate students		Master's	
	Total	Percent	All	Percent
All full-time	433916	100.0	209221	100.0
Self-support	161641	37.3	139373	66.0
All sources of support	272275	62.7	69848	33.4
Federal	65999	15.2	10736	5.1
Institutional	182135	42.0	52319	25.0
All mechanisms of support	272275	62.7	69848	33.4

Original Hierarchical Table



Hierarchical Tree from HiTab

② Alg. 1

```

top_root: [
  (Source..., full-time..., total),
  (Source..., full-time..., percent),
  (Source..., Master's, All),
  (Source..., Master's, Percent)
]

left_root: [
  (full-time, self-support, None),
  (full-time, All sources of
  support, None),
  ...
]
  
```

Multi-index Object

```

df = pd.DataFrame(
  data_matrix,
  index= top_root,
  columns= left_root
)
  
```

# Code for Table Question Answering

---

Who is more likely to have cancer,  
*the elder* or *the young*?

```
def compare_larger(values: list[float], args: list[str]) -> str:  
    """Return the argument associated with the larger value."""  
    return args[values.index(max(values))]
```

```
def compare_smaller(values: list[float], args: list[str]) -> str:  
    """Return the argument associated with the smaller value."""  
    return args[values.index(min(values))]
```

## The QA API

### Operation APIs

```
compare_larger([  
    data.loc[('Illness', 'Cancer', 'Percent'), ('All  
Patients', 'Elders')],  
    data.loc[('Illness', 'Cancer', 'Percent'), ('All  
Patients', 'Young')],  
    ['Elders', 'Young'])
```

# Code for Table Question Answering

---

- ▶ Code gives a unified representation across varied table formats and datasets

<b>Dataset</b>	<b>HiTab</b>	<b>Spider</b>	<b>AIT-QA</b>	<b>WikiTQ</b>
Baseline	MAPO 40.7	DIN-SQL 61.5	RCI 51.8	BINDER <b>54.8</b>
Codex	59.6	61.2	77.8	41.7
<b>w/ API (Ours)</b>	<b>69.3</b>	<b>63.8</b>	<b>78.0</b>	42.4



# GILL: A More General Multimodal LM

User prompts    Retrieved    Generated

I am thinking of an animal. It is brown and furry, and has a tail. What is it?


A squirrel.

No, the animal I am thinking of lives in the river.


A beaver.

Show me an illustration of a beaver


A beaver.



Yes, what about a pencil drawing of one?



What about a comic illustration of one?



## Ours

Retrieval and generation multimodal LM.  
Decides when to retrieve or generate.

I am thinking of an animal. It is brown and furry, and has a tail. What is it?

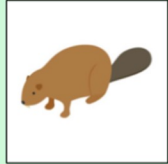
A squirrel.

No, the animal I am thinking of lives in the river.


A beaver.

Show me an illustration of a beaver


A beaver.



Yes, what about a pencil drawing of one?



What about a comic illustration of one?



## FROMAGe

Retrieval only multimodal language model. Image outputs limited to the candidate retrieval set.

I am thinking of an animal. It is brown and furry, and has a tail. What is it?


A squirrel.

No, the animal I am thinking of lives in the river.

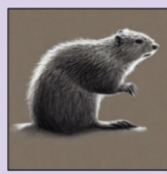
A beaver.

Show me an illustration of a beaver


A beaver.



Yes, what about a pencil drawing of one?



What about a comic illustration of one?



## Stable Diffusion

Generation only text-to-image model.  
Less sensitive to longer text inputs (such as dialogue).



# GILL: A More General Multimodal LM

- [Frozen](#) (Tsimpoukelli et al., 2021)
- [Flamingo](#) (Alayrac et al., 2022)
- [BLIP-2](#) (Li et al., 2023)
  - Process **image** + **text**, generate **text** only
- [FROMAGe](#) (Koh et al., 2023)
  - Process **image** + **text**, generate **text** + **retrieve images**
- **GILL** (this work)
  - Process **image** + **text**, generate **text** + **retrieve images** + **generate images**
  - Decides whether to retrieve images or generate from scratch
  - Resource efficient: trained on 2 GPUs for 2 days

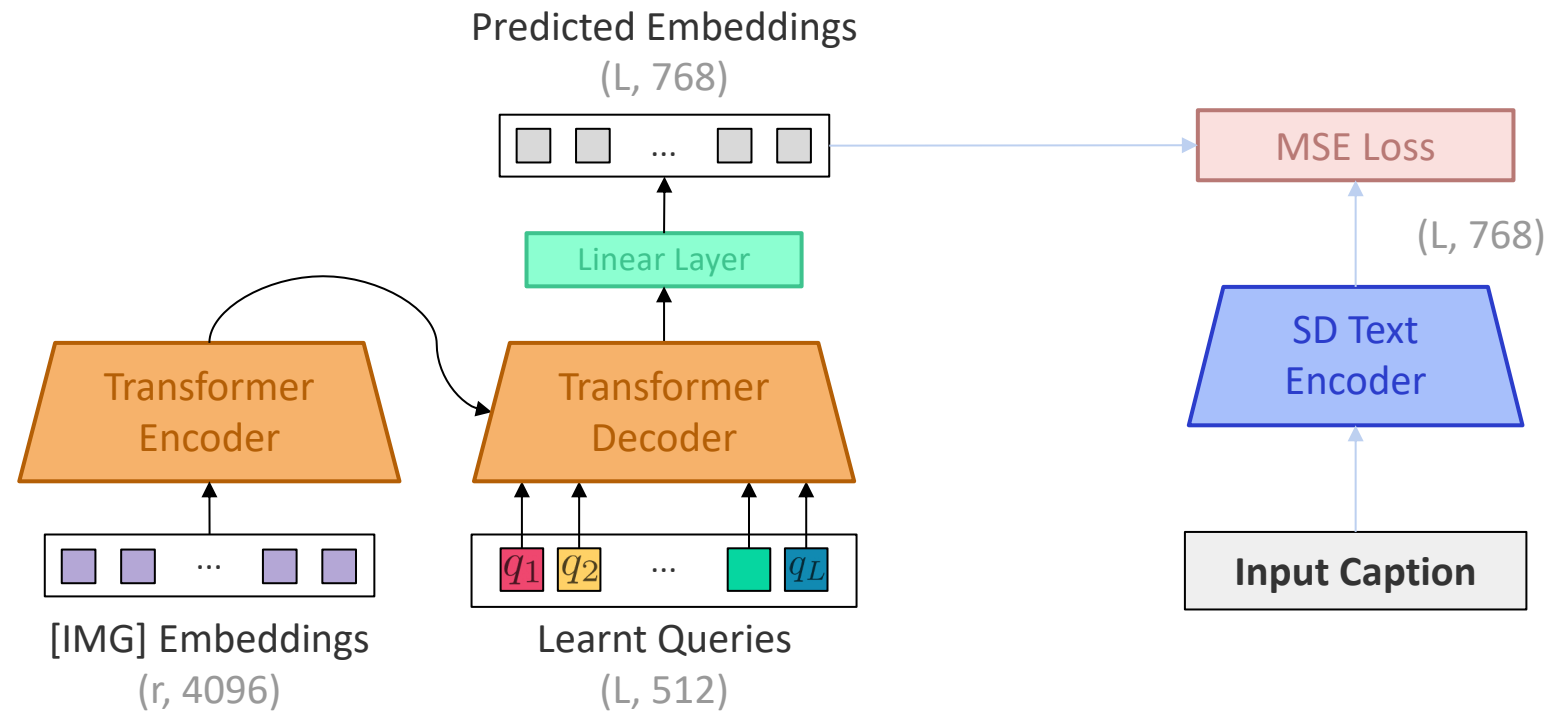
# GILL

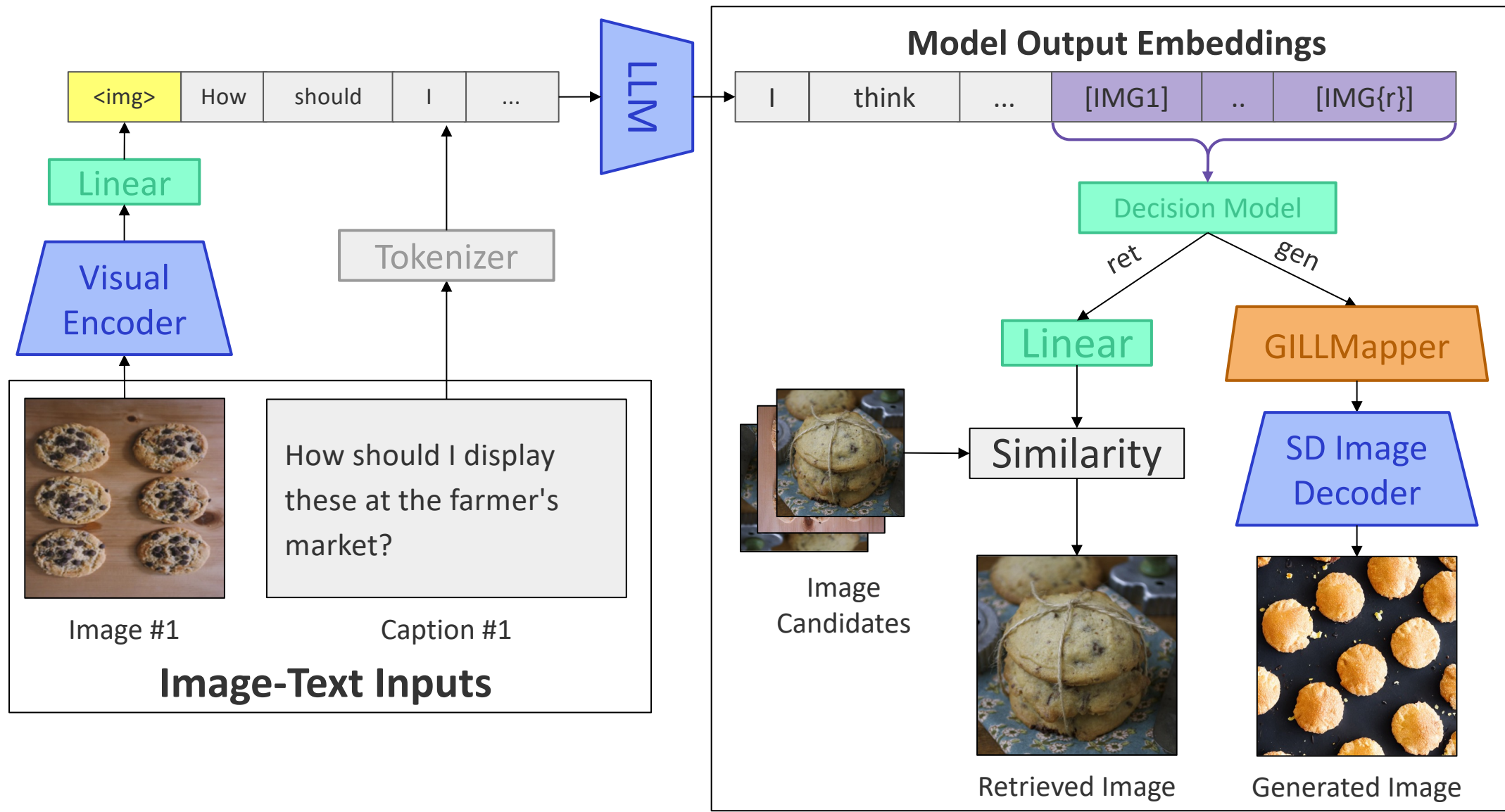
## Generating Images with Large Language Models

- **Capable of retrieving images, generating images, and generating text**
  - Can condition on arbitrarily interleaved image + text inputs
  - Generate text, generate images, and retrieve images as part of the output
- **Leverage the learnt abilities of pre-trained text-only LLMs**
  - In-context learning
  - Sensitivity to input prompts
  - Generate long and coherent dialogue
- **Model agnostic**
  - We use a 7B LLM, the CLIP encoder, and the Stable Diffusion image generator
  - Likely benefits from using larger and stronger LLMs in the future
  - Can be applied with other visual models (e.g., OCR) to introduce new abilities

# GILLMapper: An Improved LLM-to-Generator Map

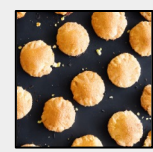
- Previous approaches use linear mappings between LLMs and visual models
- This is insufficient for image generation: decoders require dense information





**Final Model Outputs:**

I think they look best when they are on a tray with a little bit of space between them.









# Evaluation: Contextual Image Generation

Model	CLIP Similarity ( $\uparrow$ )			LPIPS ( $\downarrow$ )		
	1 caption	5 captions	5 caps, 4 images	1 caption	5 captions	5 caps, 4 images
GLIDE [34]	0.582	0.591	-	0.753	0.745	-
Stable Diffusion [43]	<b>0.592</b> $\pm 0.0007$	0.598 $\pm 0.0006$	-	0.703 $\pm 0.0003$	0.704 $\pm 0.0004$	-
GILL	0.581 $\pm 0.0005$	<b>0.612</b> $\pm 0.0011$	<b>0.641</b> $\pm 0.0011$	<b>0.702</b> $\pm 0.0004$	<b>0.696</b> $\pm 0.0008$	<b>0.693</b> $\pm 0.0008$

- Our model outperforms Stable Diffusion on longer input contexts
- This is despite GILL (essentially) distilling from SD!
- GILL benefits from the abilities of the LLM (sensitivity to longer inputs, word orderings, in-context learning)

# Evaluation: Contextual Image Generation

- Given a Visual Dialogue, generate a relevant image
- Need to condition on long dialogue-like text (OOD with finetuning data)

Q: is the man alone? A: yes, the man is alone 1	Q: is it sunny outside? A: no, it is not sunny outside 2	Q: what color is the snowboard? A: the snowboard is grey in color 3	Q: is the man wearing a cap? A: the man is wearing a black cap 4	...	Q: what color are the glasses? A: the glasses are white in color 8	Q: can you see the sky? A: no it's totally dark 9	Q: does it look like he's having fun? A: he seems to be enjoying 10			
<b>VisDial Inputs</b>										
Q: what color are the dogs? A: 1 of the dog is white and the other dog is light brown 1	Q: can you tell what breed they are? A: i can't really tell what breed they are, perhaps german shepherd 2	Q: are they both wearing a hat? A: only 1 is wearing a hat 3	...	Q: are they standing in grass? A: no, they are standing on dirt 8	Q: are they looking at each other? A: no, they are facing away from each other 9	Q: do they seem like they like each other? A: can't tell 10				
<b>VisDial Inputs</b>										
								<b>Stable Diffusion</b>	<b>Ours</b>	<b>Groundtruth</b>

# Evaluation: Contextual Image Generation

---

Model	CLIP Similarity ( $\uparrow$ )			LPIPS ( $\downarrow$ )		
	1 round	5 rounds	10 rounds	1 round	5 rounds	10 rounds
GLIDE [34]	<b>0.562</b>	0.595	0.587	0.800	0.794	0.799
Stable Diffusion [43]	0.552 $\pm$ 0.0015	<b>0.629</b> $\pm$ 0.0015	0.622 $\pm$ 0.0012	<b>0.742</b> $\pm$ 0.0010	0.722 $\pm$ 0.0012	0.723 $\pm$ 0.0008
GILL	0.528 $\pm$ 0.0014	0.621 $\pm$ 0.0009	<b>0.645</b> $\pm$ 0.0010	<b>0.742</b> $\pm$ 0.0022	<b>0.718</b> $\pm$ 0.0028	<b>0.714</b> $\pm$ 0.0006

# GILLMapper: A Stronger LLM-to-Generator Mapping

---

Image generators require **denser** input sequences. Linear mappings are insufficient.

<b>Model</b>	<b>CC3M</b>	<b>VIST</b>
	<b>FID (↓)</b>	<b>CLIP Sim (↑)</b>
Stable Diffusion [43]	<b>13.94</b>	0.598
Ours + Linear	15.50	0.500
Ours + 3-layer MLP	15.33	0.502
Ours + Transformer Encoder	16.30	0.605
Ours + GILLMapper	15.31	<b>0.641</b>



# Other Abilities: Text-to-Image Generation

---



**Stable Diffusion**

“A dignified beaver wearing glasses, a vest, and colorful neck tie. He stands next to a tall stack of books in a library.”



**Ours**

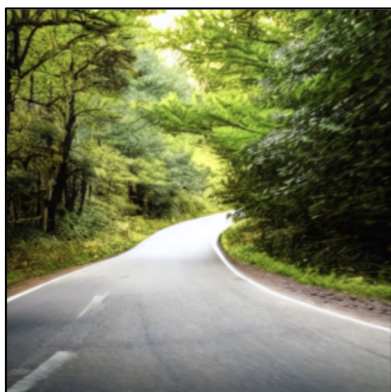


**Stable Diffusion**

“Snow mountain and tree reflection in the lake”



**Ours**



**Stable Diffusion**

“A drop-top sports car coming around a bend in the road”



**Ours**



**Stable Diffusion**

“a group of penguins in a snowstorm”



**Ours**

# Evaluation: HumanEval Benchmark

---

Constructed by authors of Codex paper; programming puzzle/simple contest problems. Evaluated using unit tests.

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
    than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```

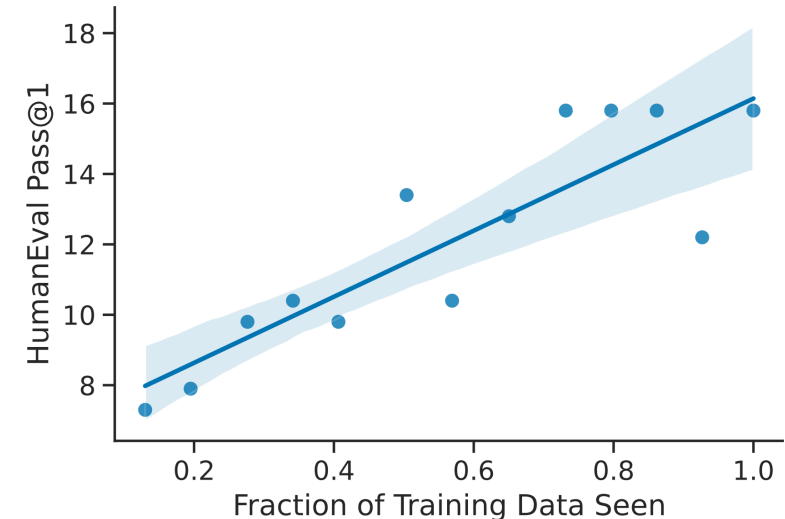
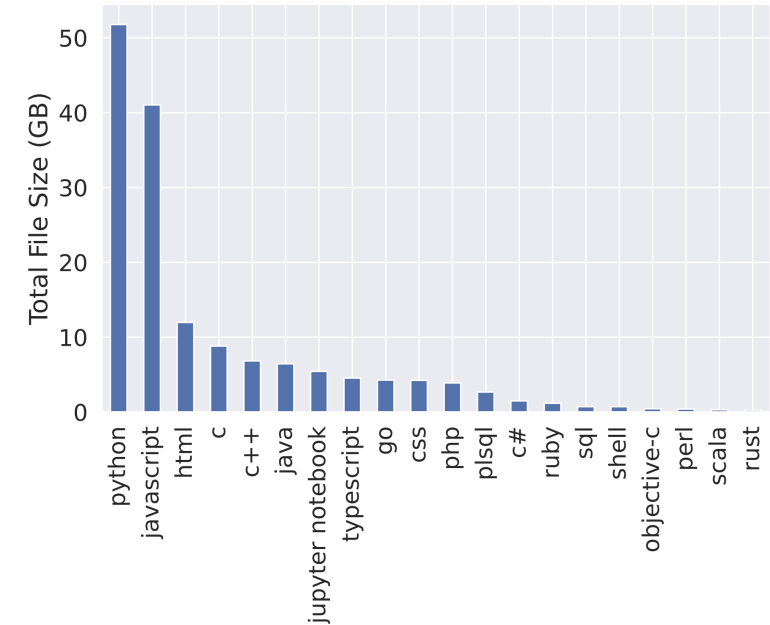
# Model Training

## ▶ Training Data

- ▶ 600K permissively-licensed repositories from GitHub & GitLab. ~150GB total
- ▶ StackOverflow: questions, answers, comments. ~50GB

## ▶ Models

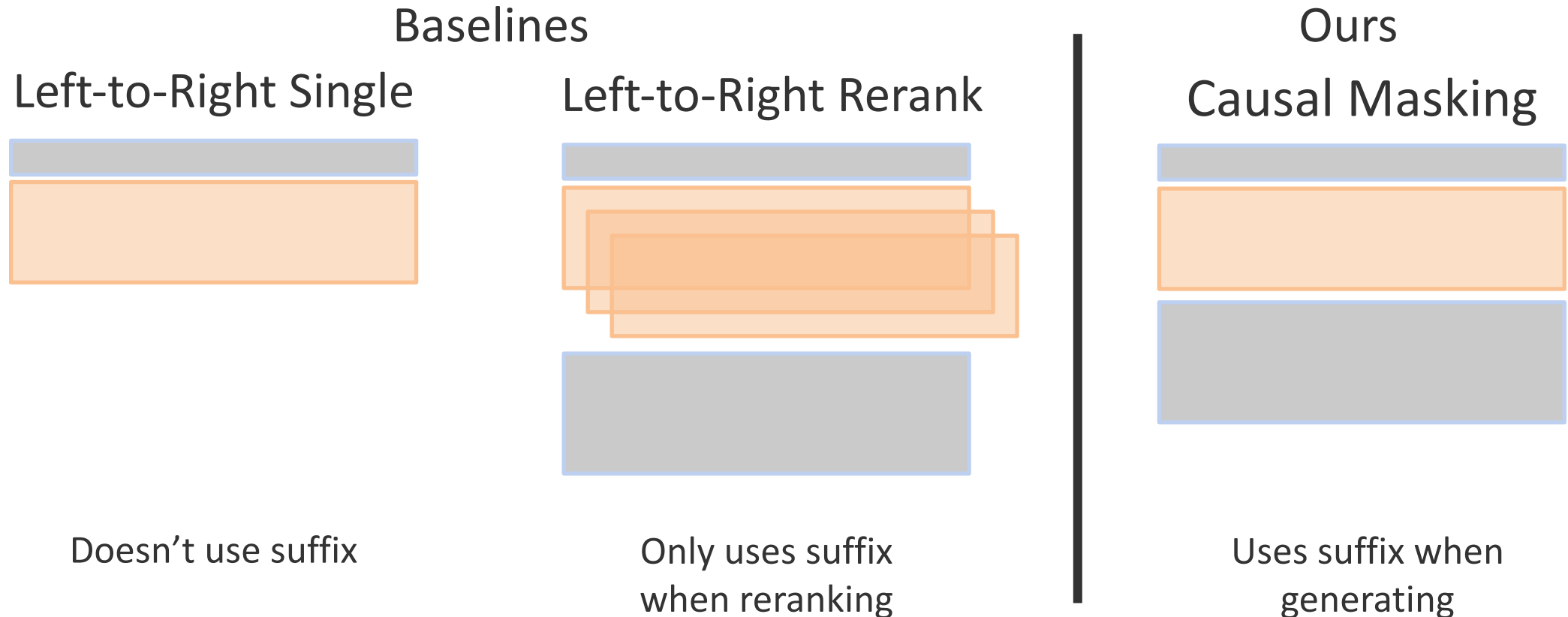
- ▶ Standard transformer LM
- ▶ 1B model: ~1 week on 128 V100s
- ▶ 6B model: ~3 weeks on 240 V100s



# Evaluation

---

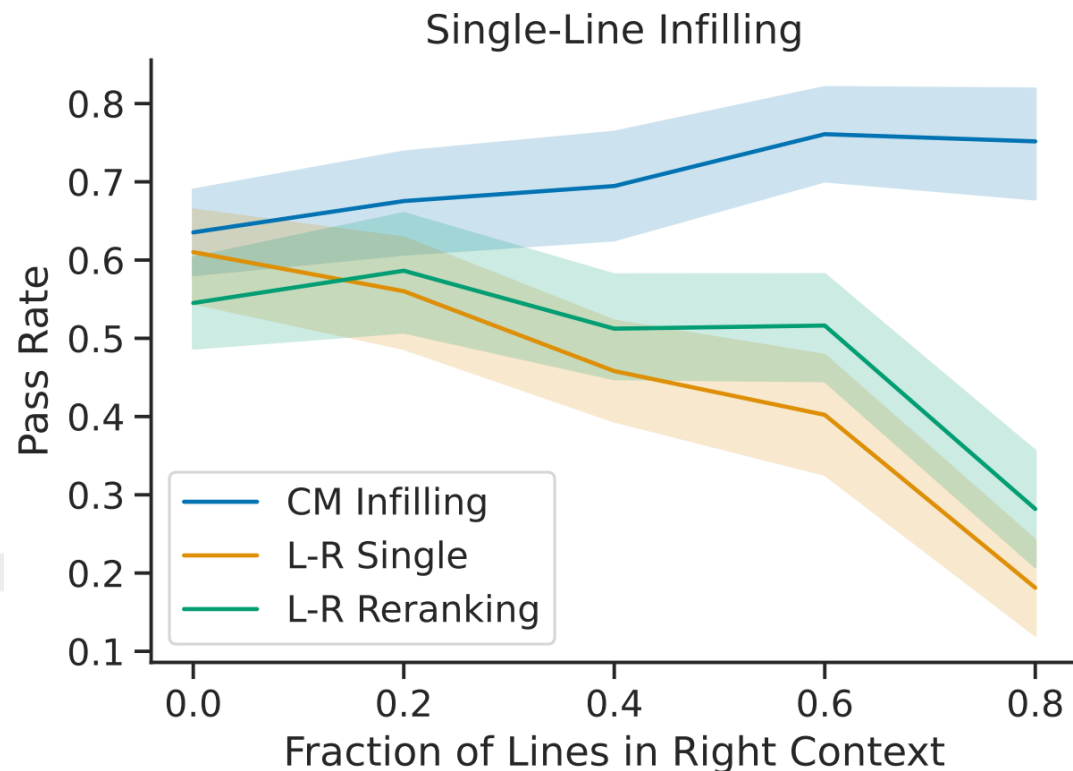
- ▶ Zero-shot evaluation on several software development-inspired code infilling tasks (we'll show two).
- ▶ Compare the model in three different modes to evaluate benefits of suffix context



# Evaluation: Function Completion

```
from typing import List

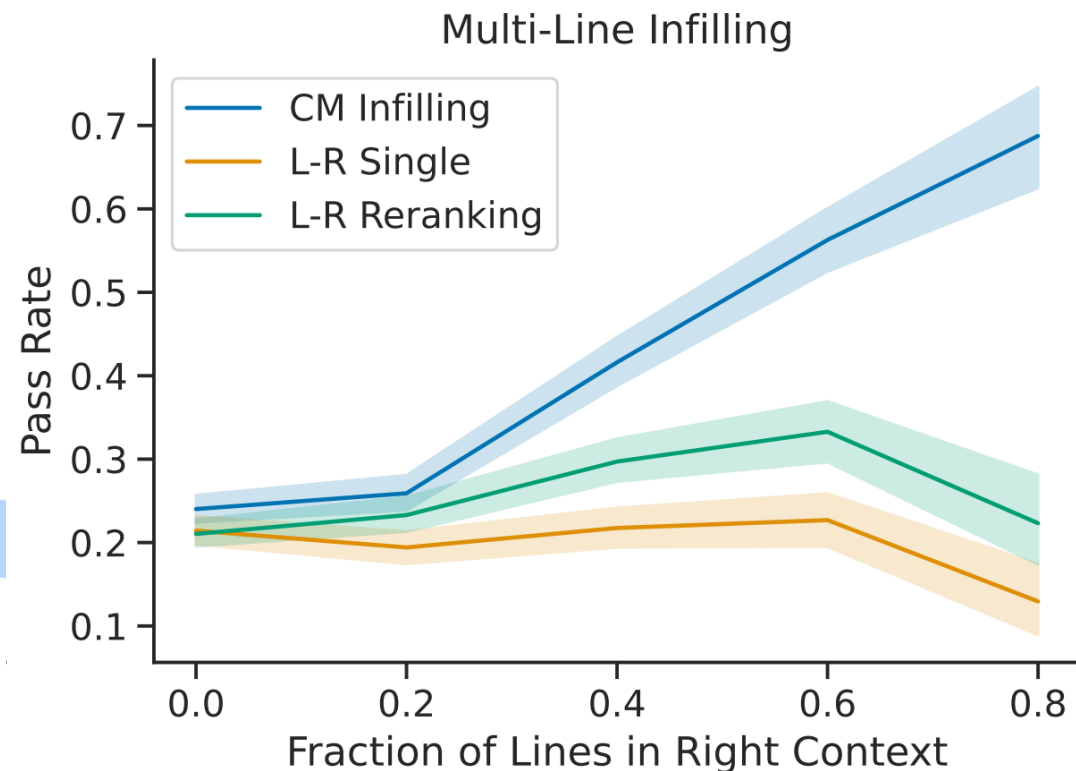
def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
    than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```



# Evaluation: Function Completion

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
    than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```



# Evaluation: Function Completion

---

Fill in one or more lines of a function; evaluate with unit tests.

```
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
    than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```

Method	Pass Rate
L-R single	24.9
L-R reranking	28.2
CM infilling	38.6



# Evaluation: Docstring Generation

---

```
def count_words(filename: str) -> Dict[str, int]:  
    """  
    Counts the number of occurrences of each word in the given file.  
  
    :param filename: The name of the file to count.  
    :return: A dictionary mapping words to the number of occurrences.  
    """  
    with open(filename, 'r') as f:  
        word_counts = {}  
        for line in f:  
            for word in line.split():  
                if word in word_counts:  
                    word_counts[word] += 1  
                else:  
                    word_counts[word] = 1  
    return word_counts
```

Method	BLEU
Ours: L-R single	16.05
Ours: L-R reranking	17.14
Ours: Causal-masked infilling	18.27

# Evaluation: Return Type Prediction

## Type Inference

```
def count_words(filename: str) -> Dict[str, int]:  
    """Count the number of occurrences of each word in the file."""  
    with open(filename, 'r') as f:  
        word_counts = {}  
        for line in f:  
            for word in line.split():  
                if word in word_counts:  
                    word_counts[word] += 1  
                else:  
                    word_counts[word] = 1  
    return word_counts
```

Method	F1
Ours: Left-to-right single	30.8
Ours: Left-to-right reranking	33.3
Ours: Causal-masked infilling	<b>59.2</b>
TypeWriter (Supervised)	48.3

# Training Models on Human Instructions

---

