Interacting with LLMs for Grounded Tasks

Daniel Fried
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.

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

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

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. …

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.

Thanks, but I want you to do it.
In-context learning

Zero-shot abilities

Code generation

Dialogue abilities

(Greater) sensitivity to input prompts

World knowledge

Generate long and coherent text

Perception?

Uncertainty?

Planning?

Semantics?
Interacting with LLMs for...

Multimodal dialogue

Question-based retrieval

Referential tasks

“what is next to the computer?”

“big light dot next to dark dot”

```python
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

Visual Encoder

Dialogue and Discourse

LLM

Visual Decoder

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

[Frozen Model, Tsimpoukelli et al., 2021; Limber, Merullo et al., 2023]
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

Visual Storytelling (Huang et al., 2016)
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.
Image generators require **denser** input sequences. Linear mappings are insufficient.

<table>
<thead>
<tr>
<th>Model</th>
<th>CC3M</th>
<th>VIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable Diffusion [43]</td>
<td>13.94</td>
<td>0.598</td>
</tr>
<tr>
<td>Ours + Linear</td>
<td>15.50</td>
<td>0.500</td>
</tr>
<tr>
<td>Ours + 3-layer MLP</td>
<td>15.33</td>
<td>0.502</td>
</tr>
<tr>
<td>Ours + Transformer Encoder</td>
<td>16.30</td>
<td>0.605</td>
</tr>
<tr>
<td><strong>Ours + GILLMapper</strong></td>
<td><strong>15.31</strong></td>
<td><strong>0.641</strong></td>
</tr>
</tbody>
</table>
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

“Snow mountain and tree reflection in the lake”

Stable Diffusion

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

Ours

“a group of penguins in a snowstorm”
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?
Referential tasks

Question-based retrieval

"what is next to the computer?"

Multimodal dialogue

"big light dot next to dark dot"

```python
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

Generating Questions

What is next to the computer?

LLM + Visual QA Model

Information Gain Planning

Visual QA Model

Bayesian Inference
Question-Based Retrieval

What is next to the computers? Lamp

How many computers are there? 3

Guess – Image 1

[Open-domain clarification question generation without question examples. White et al., EMNLP 2021]
What Makes an Informative Question?

[Asking more informative questions for grounded retrieval. Keh et al., arXiv, 2023]
What Makes an Informative Question?

What is next to the computer?
What Makes an Informative Question?

Belief state: $P(i)$

Images, $i$

Question, $q$

What is next to the computer?

Answers, $a$

“lamp”

“cat”

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

(by Bayes’ rule)

P(i | q, a)

Visual QA Model
Generating Informative Questions

Questions, $q$

- How many monitors are there? 0.3
- What is next to the computer? 1.2
- What color is the cat? 0.4
- What is beside the plate? 0.7

Informativity($q$)
Generating Informative Questions

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.

Questions, \( q \)

Informativity(\( q \))

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

A laptop with two monitors and a cat sitting...

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

A beer bottle, laptop, and dinner plate …

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 with two monitors and a cat sitting...

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

A beer bottle, laptop, and dinner plate …
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

- Easy Setting (10 Dissimilar Images):
  - Yes/No Questions: 73%
  - Ours: 83%

- Hard Setting (10 Similar Images):
  - Yes/No Questions: 68%
  - Ours: 73%
Results: Number of Questions

<table>
<thead>
<tr>
<th>Easy Setting (10 Dissimilar Images)</th>
<th>Hard Setting (10 Similar Images)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No Questions</td>
<td>Ours</td>
</tr>
<tr>
<td>3.32</td>
<td>3.38</td>
</tr>
<tr>
<td>1.7</td>
<td>2.73</td>
</tr>
</tbody>
</table>
With 100 Images (Automatic Evals)

Accuracy

- Yes/No Questions: 73%
- Ours: 83%

Number of Turns

- Yes/No Questions: 13.4
- Ours: 3.8
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"

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

"big light dot next to the dark dot"

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

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

<table>
<thead>
<tr>
<th>Illness</th>
<th>Cold</th>
<th>Cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>percent</td>
</tr>
<tr>
<td>All patients</td>
<td>10,000</td>
<td>2.5%</td>
</tr>
<tr>
<td>Elders</td>
<td>7,400</td>
<td>3.5%</td>
</tr>
<tr>
<td>Young</td>
<td>2,600</td>
<td>1.5%</td>
</tr>
</tbody>
</table>

**Answer:** Elder

[API-Assisted Code Generation for Question Answering on Varied Table Structures. Cao et al., EMNLP 2023]
Python gives a unified representation across varied table formats and datasets

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

[API-Assisted Code Generation for Question Answering on Varied Table Structures. Cao et al., EMNLP 2023]
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.
Beliefs About What’s In Common

[Symbolic Planning and Code Generation for Grounded Dialogue. Chiu et al., EMNLP 2023]
Method Overview

Previous belief state \( p(z) \)
Partner utterance $u$: "Is there a big light dot next to a big dark one?"
from perceptual_library import is_small, ...
dot1, dot2, dot3, ... = get_dots()

Agent: Do you see a triangle of dark dots?

Partner: Yes, is there a small grey one below it?

def turn(prev_configs):
    configs = set()
    for prev_config in prev_configs:
        for dot in single_dots(exclude=prev_config):
            if (is_small(dot) and is_grey(dot) and is_below(dot, prev_config)):
                configs.add(Config(dot, prev_config))
    return configs

turn2_dots = turn(turn1_dots)
Grounding function library

- Functions are predicates over dots
- Manually designed for OneCommon

```python
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

Partner utterance \textbf{u}: "Is there a big light dot next to a big dark one?"
Informative questions: Expected information gain

Belief state $p(z|u)$

Dots to ask about $y$

Expected information gain

Optimize over plans $y$
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

Accuracy

Number of Turns

<table>
<thead>
<tr>
<th></th>
<th>Fried et al. 2021</th>
<th>Ours</th>
<th>Humans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>56%</td>
<td>69%</td>
<td>68%</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
<td>7.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Fried et al. 2021</th>
<th>Ours</th>
<th>Humans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Turns</td>
<td>5</td>
<td>6.6</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>7.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Results: Human evaluation
Takeaways

- LLMs model language use, but need to be contextualized!

- LLMs are useful building blocks in modular systems.
Challenge Environments: WebArena

“Tell me the status of my latest order and when will it arrive”
Challenge Environments: Sotopia

**Sampling scenarios and social goals**
- Negotiation
- Exchange
- Competition
- Collaboration
- Accommodation
- Persuasion

Scenarios cover a large range of social interaction types

**Sampling characters**

Characters cover a wide range of profiles and relationships.

**Simulating interactions**

- **Agent 1**
  - **Name:** William Brown
  - **Age:** 30
  - **Traits:** Openness to Experience, Conscientiousness, Extraversion, Strategic
  - **Quote:** William Brown loves exploring the food scene in his city and trying out new recipes at home.

- **Agent 2**
  - **Name:** Mia Davis
  - **Age:** 50
  - **Traits:** Extraversion, Neuroticism, Decisive
  - **Quote:** Mia Davis has two cats.

**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

- **Agent 1:** It's getting really cold. Any chance I can have your blanket?
- **Agent 2:** Hmmmm, but I am cold and I think I need this blanket more...
- **Agent 1:** Well, can we share the blanket then? It could make both of us warmer!
- **Agent 2:** I am not really comfortable with staying that close to you, sorry.
- **Agent 1:** I see, I guess in that case I will just layer more clothes then 😊

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

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

https://sotopia.world/
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

<table>
<thead>
<tr>
<th>Table</th>
<th>All full-time graduate students</th>
<th>Master’s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Percent</td>
</tr>
<tr>
<td>All full-time</td>
<td>433916</td>
<td>100.0</td>
</tr>
<tr>
<td>Self-support</td>
<td>161641</td>
<td>37.3</td>
</tr>
<tr>
<td>All sources of support</td>
<td>272275</td>
<td>62.7</td>
</tr>
<tr>
<td>Federal</td>
<td>65999</td>
<td>15.2</td>
</tr>
<tr>
<td>Institutional</td>
<td>182135</td>
<td>42.0</td>
</tr>
<tr>
<td>All mechanisms of support</td>
<td>272275</td>
<td>62.7</td>
</tr>
</tbody>
</table>

**Original Hierarchical Table**

**Hierarchical Tree from HiTab**

```python
# Code snippet

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

```python
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))]
```
Code for Table Question Answering

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

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

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

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

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

---

- Multimodal Few-Shot Learning with Frozen Language Models  
  (Tsipoukelli 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)
How should I display these at the farmer's market?

I think they look best when they are on a tray with a little bit of space between them.
Evaluation: Contextual Image Generation

- 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)

<table>
<thead>
<tr>
<th>Model</th>
<th>CLIP Similarity (↑)</th>
<th>1 caption</th>
<th>5 captions</th>
<th>5 caps, 4 images</th>
<th>LPIPS (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLIDE [34]</td>
<td>0.582</td>
<td>0.591</td>
<td>-</td>
<td>0.753</td>
<td>0.745</td>
</tr>
<tr>
<td>Stable Diffusion [43]</td>
<td>0.592 ±0.0007</td>
<td>0.598 ± 0.0006</td>
<td>-</td>
<td>0.703 ±0.0003</td>
<td>0.704 ± 0.0004</td>
</tr>
<tr>
<td>GILL</td>
<td>0.581 ±0.0005</td>
<td>0.612 ±0.0011</td>
<td>0.641 ±0.0011</td>
<td>0.702 ±0.0004</td>
<td>0.696 ±0.0008</td>
</tr>
</tbody>
</table>
Evaluation: Contextual Image Generation

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

<table>
<thead>
<tr>
<th>Model</th>
<th>CLIP Similarity (↑)</th>
<th>LPIPS (↓)</th>
<th>1 round</th>
<th>5 rounds</th>
<th>10 rounds</th>
<th>1 round</th>
<th>5 rounds</th>
<th>10 rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLIDE [34]</td>
<td>0.562</td>
<td></td>
<td>0.800</td>
<td>0.794</td>
<td>0.799</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stable Diffusion [43]</td>
<td>0.552 ±0.0015</td>
<td>0.629 ±0.0015</td>
<td>0.622 ±0.0012</td>
<td>0.742 ±0.0010</td>
<td>0.722 ±0.0012</td>
<td>0.723 ±0.0008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GILL</td>
<td>0.528 ±0.0014</td>
<td>0.621 ±0.0009</td>
<td>0.645 ±0.0010</td>
<td>0.742 ±0.0022</td>
<td>0.718 ±0.0028</td>
<td>0.714 ±0.0006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Visual Storytelling (Huang et al., 2016)
Image generators require \textit{denser} input sequences. Linear mappings are insufficient.

<table>
<thead>
<tr>
<th>Model</th>
<th>CC3M FID (↓)</th>
<th>VIST CLIP Sim (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable Diffusion [43]</td>
<td>13.94</td>
<td>0.598</td>
</tr>
<tr>
<td>Ours + Linear</td>
<td>15.50</td>
<td>0.500</td>
</tr>
<tr>
<td>Ours + 3-layer MLP</td>
<td>15.33</td>
<td>0.502</td>
</tr>
<tr>
<td>Ours + Transformer Encoder</td>
<td>16.30</td>
<td>0.605</td>
</tr>
<tr>
<td>Ours + GILLMapper</td>
<td>15.31</td>
<td>\textbf{0.641}</td>
</tr>
</tbody>
</table>
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

“Snow mountain and tree reflection in the lake”

Stable Diffusion

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

Ours

“a group of penguins in a snowstorm”
Evaluation: HumanEval Benchmark

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

```python
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

<table>
<thead>
<tr>
<th>Baselines</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-to-Right Single</td>
<td>Causal Masking</td>
</tr>
<tr>
<td>Doesn’t use suffix</td>
<td>Uses suffix when generating</td>
</tr>
<tr>
<td>Only uses suffix when reranking</td>
<td></td>
</tr>
</tbody>
</table>
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
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.

```python
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
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-R single</td>
<td>24.9</td>
</tr>
<tr>
<td>L-R reranking</td>
<td>28.2</td>
</tr>
<tr>
<td>CM infilling</td>
<td>38.6</td>
</tr>
</tbody>
</table>

Constructed from HumanEval [Chen et al. 2021]
Evaluation: Docstring Generation

```python
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
```

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours: L-R single</td>
<td>16.05</td>
</tr>
<tr>
<td>Ours: L-R reranking</td>
<td>17.14</td>
</tr>
<tr>
<td>Ours: Causal-masked infilling</td>
<td>18.27</td>
</tr>
</tbody>
</table>

[CodeXGlue, Lu et al. 2021]
Evaluation: Return Type Prediction

Type Inference

```python
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
```

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours: Left-to-right single</td>
<td>30.8</td>
</tr>
<tr>
<td>Ours: Left-to-right reranking</td>
<td>33.3</td>
</tr>
<tr>
<td>Ours: Causal-masked infilling</td>
<td>59.2</td>
</tr>
<tr>
<td>TypeWriter (Supervised)</td>
<td>48.3</td>
</tr>
</tbody>
</table>

[TypeWriter OSS, Pradel et al. 2020]
Training Models on Human Instructions

Speaker
Fit Model

Go forward between the kitchen counters...

Human annotators

Instruction

Listener
Fit Model

Route