



OpenBugger

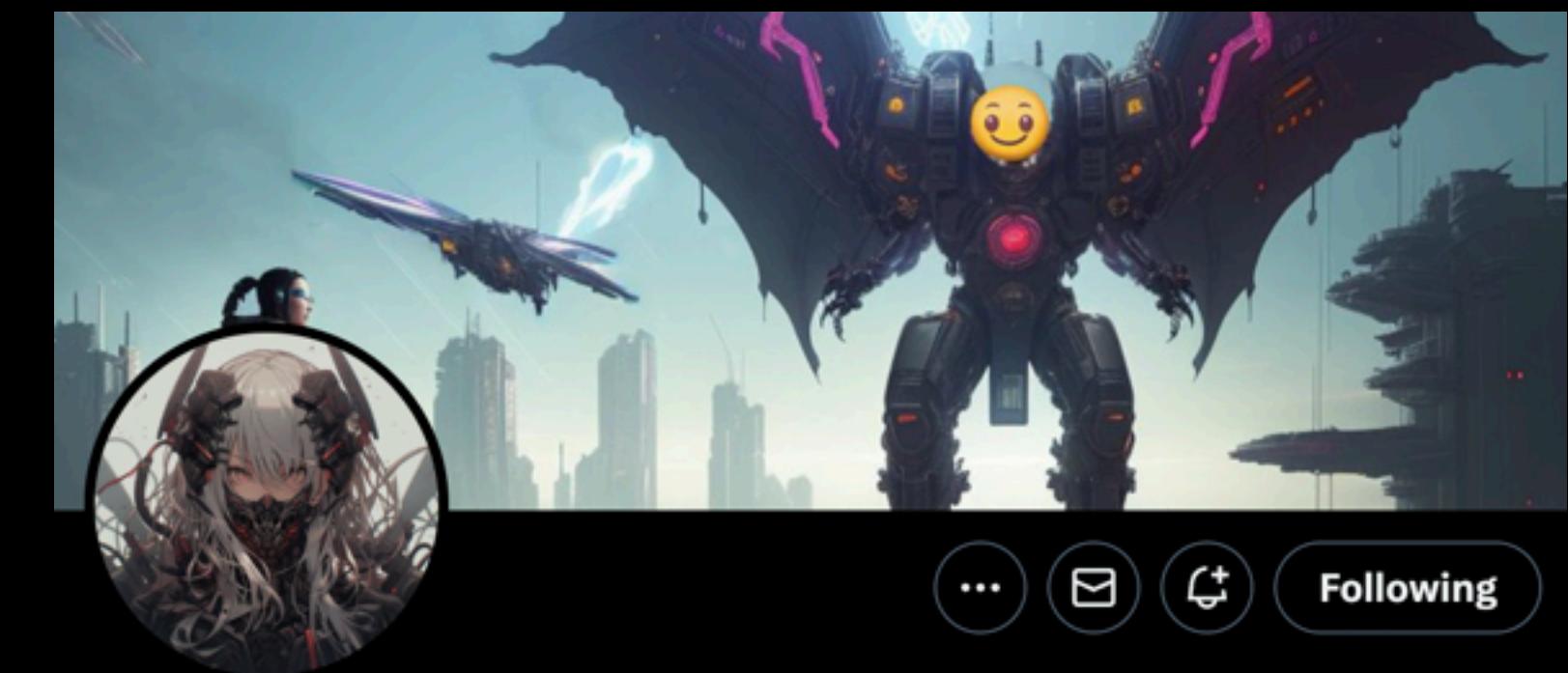
Self-Supervised Data Generation for AI Debuggers

<https://github.com/furlat/OpenBugger>

About me

Senior scientist at HK3 lab and Open Source Machine Learning
<https://github.com/furlat> - @cyndesama

- Phd in Neuroscience @University of Southern California
- AI Research since 2010
- Consulting and Startups since 2018
- Building a new OS community



TITLE	CITED BY	YEAR
Born again neural networks	870	2018
Tensor regression networks	128	2020

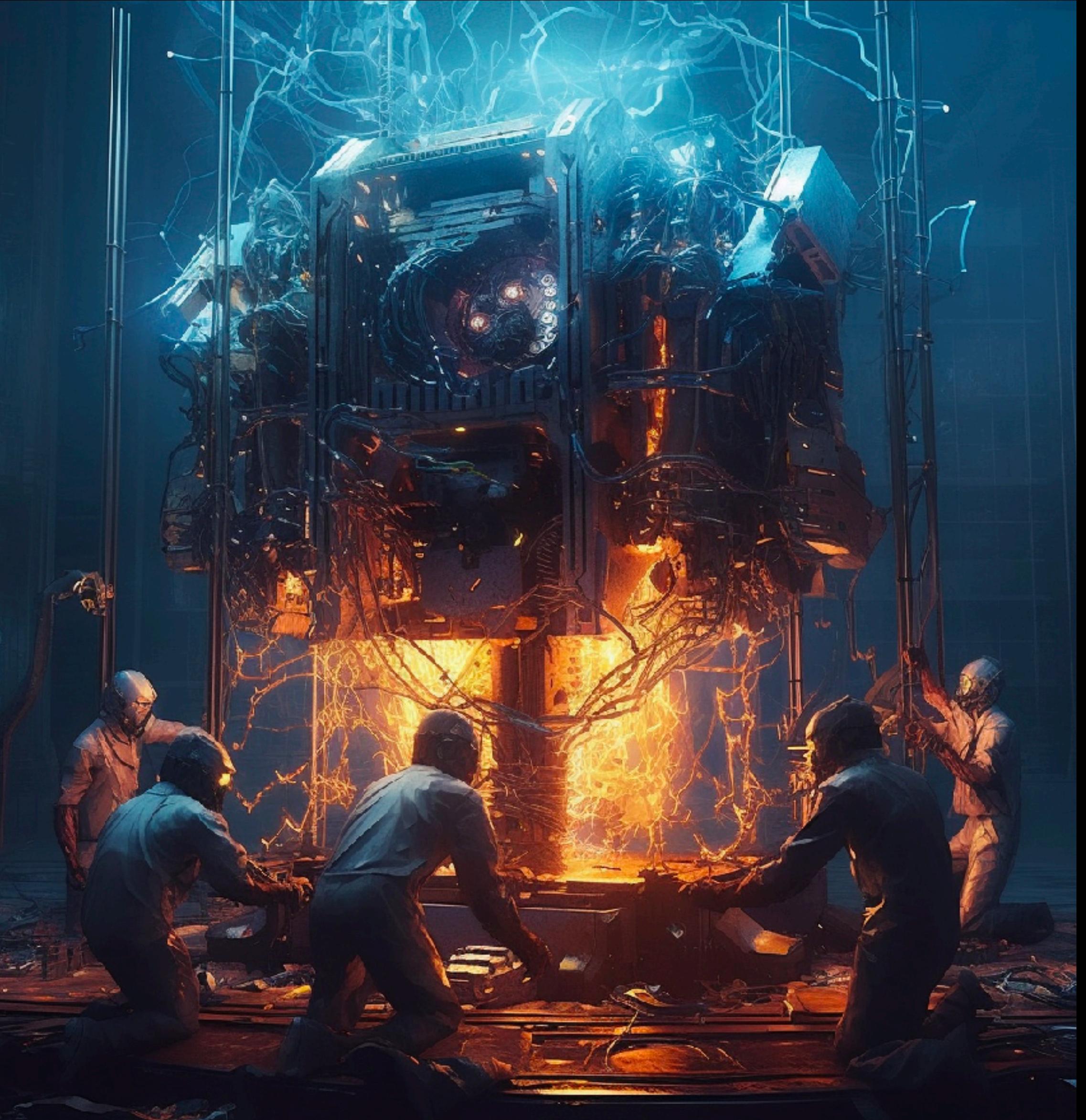
Cynde
@Cyndesama
Building a Neurodragon
{dm github handle for organization invite}

Keep up with my broken syntax

OpenAssistant (Jan)

The OS Language Model Wave

- ChatGPT and Copilot gave consumers access to AI driven code generation via api.
- The open source community was thirsty for access to the underlying models driven by curiosity and privacy concerns.
- The overwhelming participation to OpenAssistant showed that the only missing ingredients are data and compute.
- Lacking the GPUs, hundreds of creative ideas for synthetic data generation were born.



OpenAssistant (now)

The OS Language Model Wave

[https://github.com/LAION-AI/
Open-Assistant](https://github.com/LAION-AI/Open-Assistant) (33.3k stars!!!)

<https://open-assistant.io/chat/>

The screenshot shows the Open Assistant web interface. On the left, there's a sidebar with icons for creating a new chat, presenting a talk, and other features. Below that, there are sponsor links for Hugging Face, Stability AI, Redmond AI, and Weights & Biases. The main area shows a conversation where a user asks for Python code to impress an audience, and the assistant responds with a sample script using Hugging Face transformers. The script is as follows:

```
import torch
from datasets import load_dataset
from transformers import BertTokenizer, BertForSequenceClassification
from sklearn.model_selection import train_test_split

# Load data
train_data = load_dataset('./path/to/your/training/data', split='train')
valid_data = load_dataset('./path/to/your/validation/data', split='validation')

# Tokenize data
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased-whole-word-masking-sentences-mean-pooling-1024')
all_encodings = tokenizer(next(iter(train_data)), truncation=True, padding=True)

# Preprocess data (e.g., one-hot encoding)
X_train = all_encodings['input_ids'].values.astype(torch.float32).unsqueeze(0)
y_train = all_encodings['labels'].values.astype(torch.float32)

x_val = all_encodings['input_ids'].values.astype(torch.float32).unsqueeze(0)
```

On the right, there's a "Chat configuration" sidebar with various settings like Model (OA_SFT_Llama_30B_6), Preset (k50), Temperature (0.75), Max new tokens (1024), Top P (0.95), Repetition Penalty (1.2), Top K (50), and Typical P (disabled). A note at the bottom states: "This Assistant is a demonstration version that does not have internet access. It may generate incorrect or misleading information. It is not suitable for important use cases or for giving advice."

OpenBugger

Python Data Generation via Static Analysis

- From the observation that gpt reaches its limit not when it outputs bugged code, but when it fails at fixing it.
- Starting from the sci-fi idea of an AI bugging and debugging itself to peek at its inner working.
- It is easier to bug good code than debugging bad code.
- We use static analysis to analyse the syntax tree of correct python code and deterministically inject invertible bugs.
- We invert the bugged code to retrieve step-by-step debugging instruction.



@Angantyr

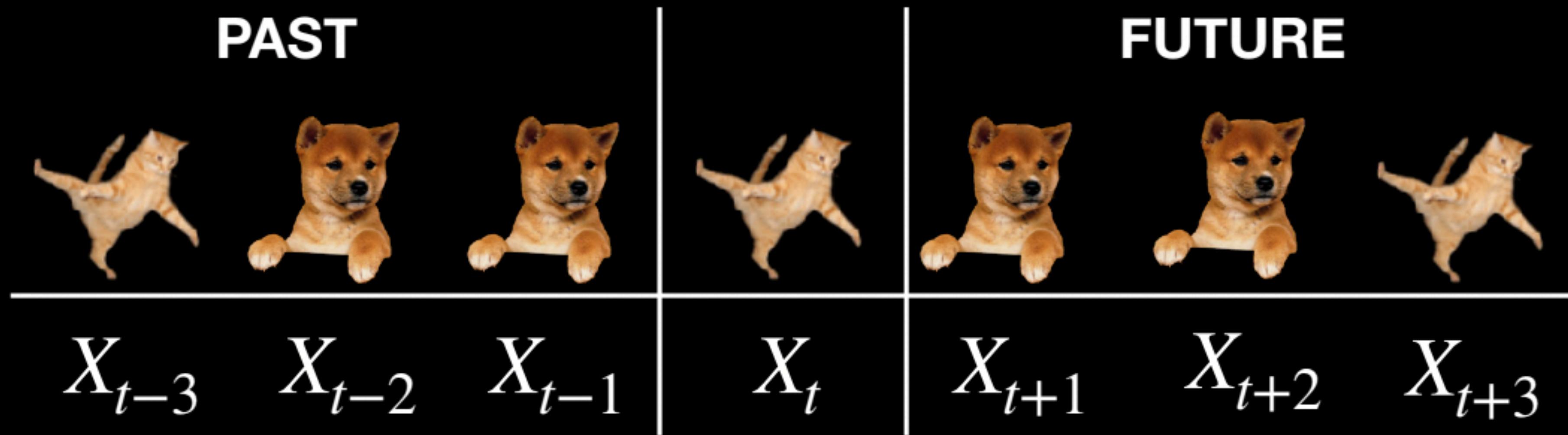
Spoilers

1. Language Models require a LOT of data.
2. Good data is rare and very expensive.
3. OpenBugger cheaply generates good data.

What Are Language Models

Statistical Models of Tokens Sequences

$$\mathbb{P}(X_t, \dots, X_\infty | X_{-\infty}, \dots, X_{t-1})$$

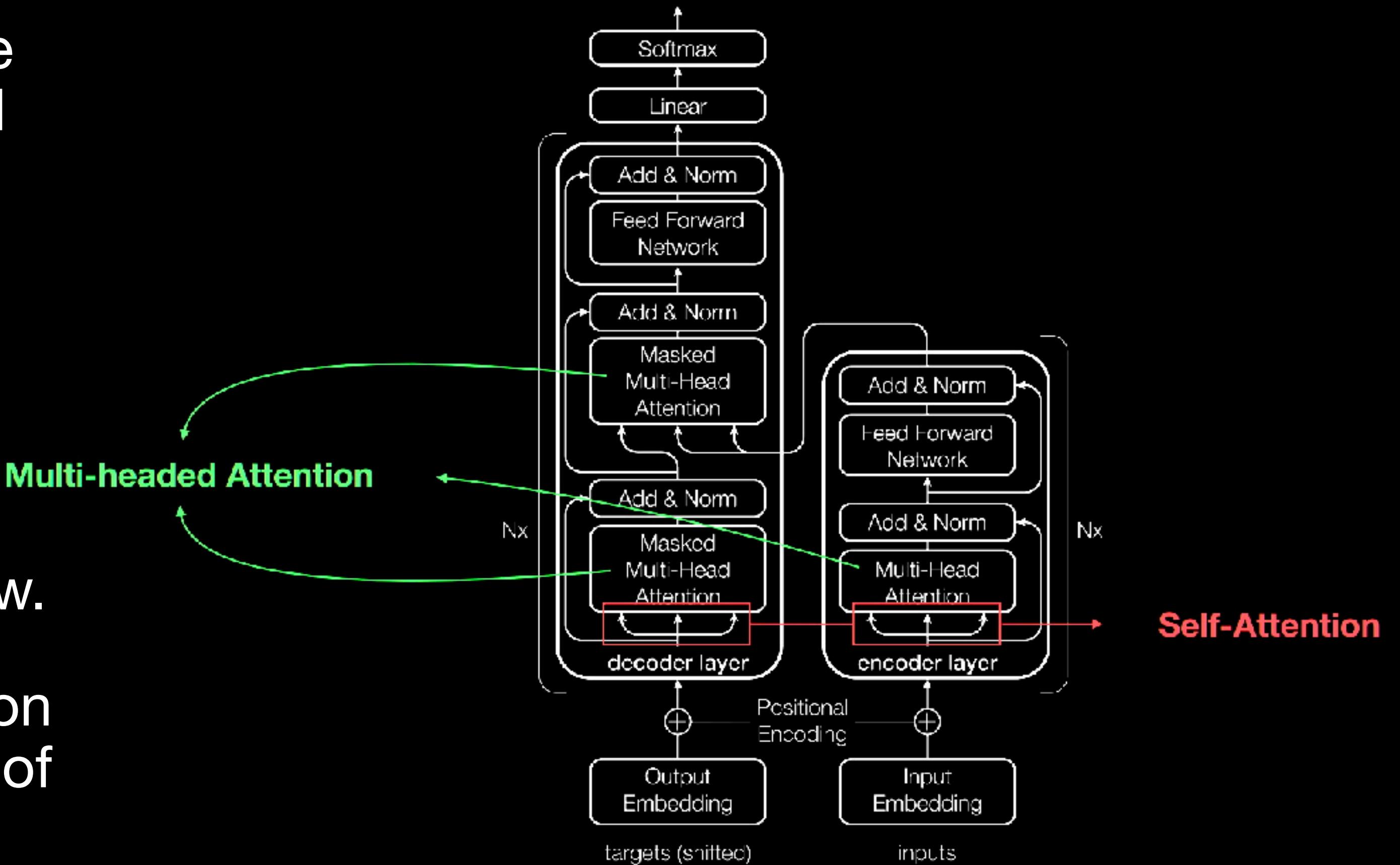


1. Introduced by Claude Shannon in 1951
2. Describe future sub-sequences with a finite discrete alphabet as conditional probabilities over past sub-sequence
3. Heavily suffer from the Curse of Dimensionality

What Are Large Language Models

GPT and the Transformers Revolution

- Vaswani et al 2017 during a Google internship (a couple of years ahead of me at USC !)
- Revolutionary architecture for sequence predictions.
- Removes all the sequential processing of recurrent network in exchange of a fixed context window.
- Enable massive model parallelisation of the architecture over thousands of nodes enabling “scaling”.

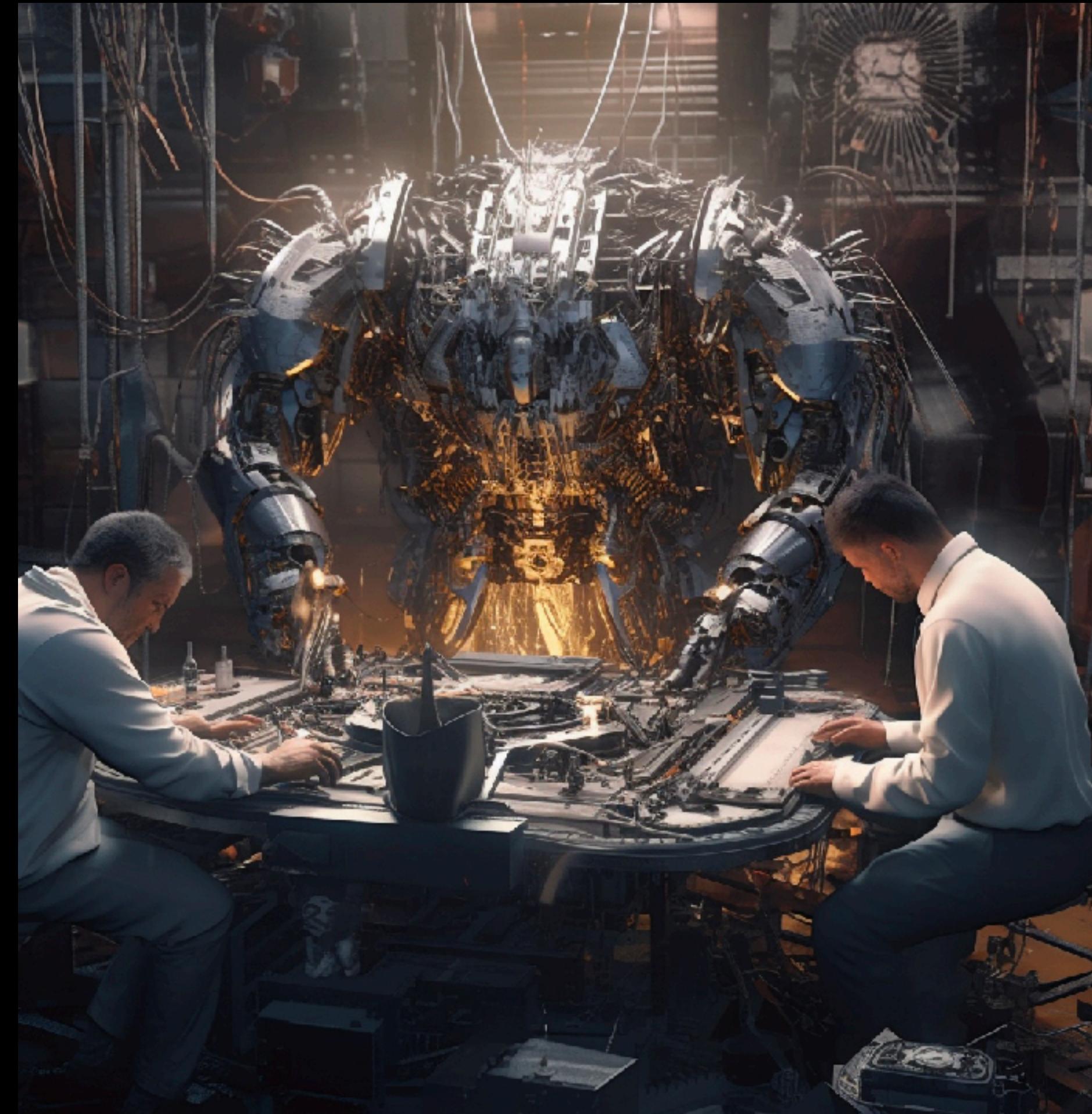


What You Need for a LLM Assistant

A simplified cookbook of LLMs chatbot training pipeline

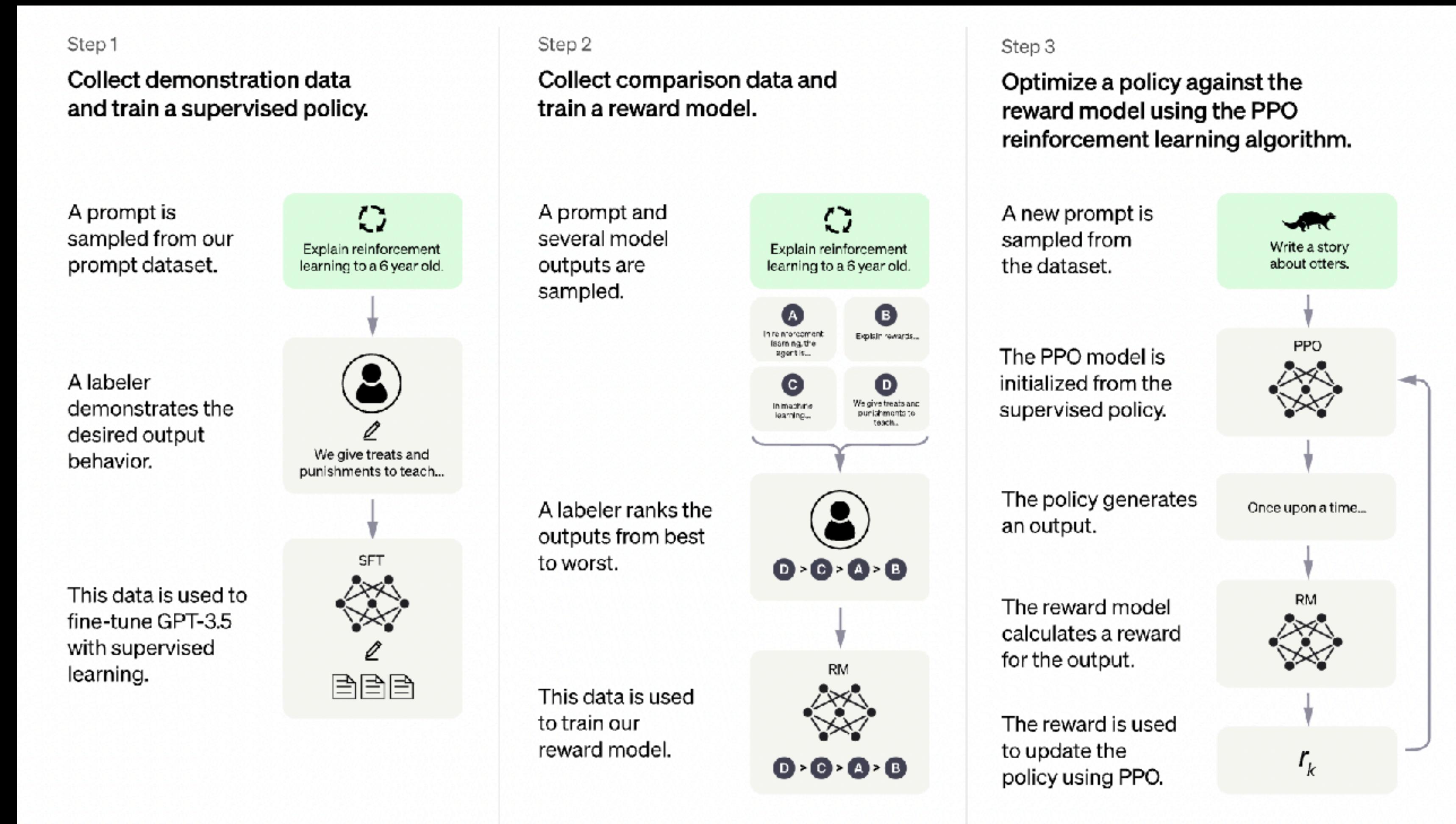
OpenAI huge contribution

- Blindly believing in Google's transformers and large language models.
- Scaling pre-training to a large percentage of the internet.
- Recycling their reinforcement learning expertise from Dota and Robotics efforts in Language Models
- High quality human feedbacks through third party suppliers.



What You Need for a LLM Assistant

A simplified cookbook of LLMs chatbot training pipeline



What You Need for a LLM Assistant

A simplified cookbook of LLMs chatbot training pipeline

1. A very large pre-training dataset [SSL Pretraining]

1. The model is fed billions of sequences of text tokens as input and is asked to predict the following ones.
2. Here is where the model is thought to learn its “factual” knowledge and its ability to understand and produce language by completing the input context.

2. A medium sized instruction tuning dataset [Instruction Tuning]

1. The models is fed millions of {instruction, input, output} examples.
2. Here the model learns to perform tasks from instructions.

3. A small sized dataset of human preferences [RLHF]

1. The model is fed thousands of tuples {instruction, input, generated_output, human_evaluation} and is trained with reinforcement learning to favour positively evaluated answers.

What You Need for a LLM Assistant

A simplified cookbook of LLMs chatbot training pipeline

OpenBugger uses Python syntax tree mutation for automatic data generation

2. A medium sized instruction tuning dataset [Instruction Tuning]

1. The models is fed millions of {instruction, input, output} examples.
2. Here the model learns to perform tasks from instructions.



What You Need for a LLM Assistant

A simplified cookbook of LLMs chatbot training pipeline

1. A very large pre-training dataset [SSL Pretraining]

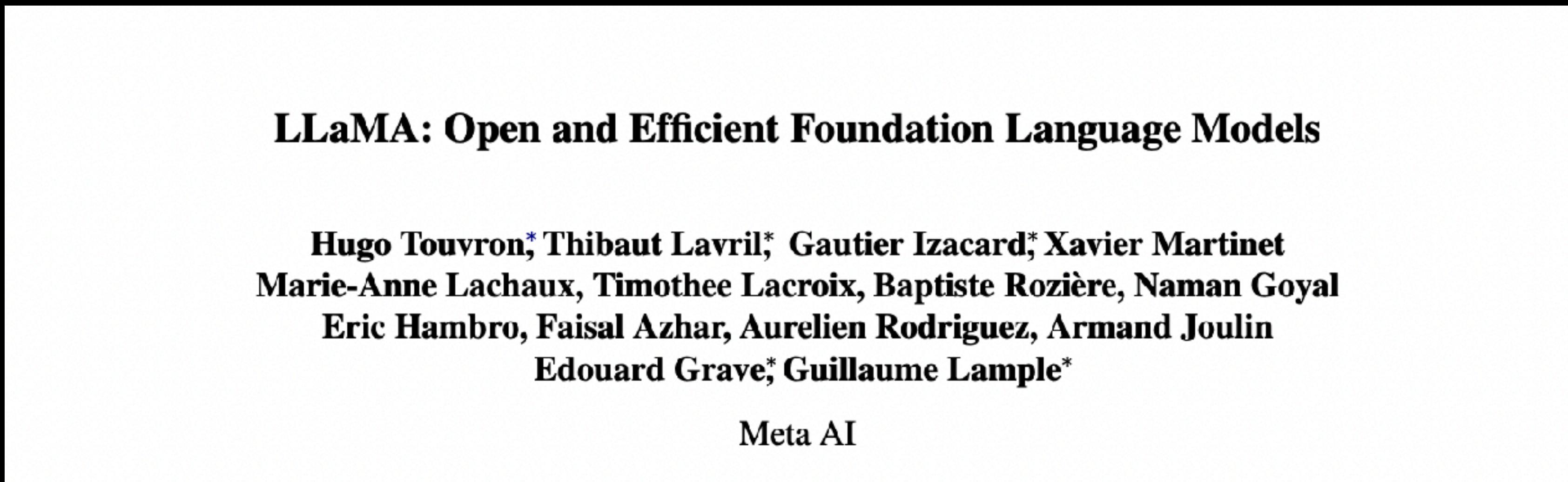
1. The model is fed billions of sequences of text tokens as input and is asked to predict the following ones.
2. Here is where the model is thought to learn its “factual” knowledge and its ability to understand and produce language by completing the input context.

The Animal Farm to the Rescue

A simplified cookbook of LLMs chatbot training pipeline

Open Source LLAMA from Meta AI

1. Suite of gpt-style pre-trained language models from 7b to 65b parameters trained over 1 trillion tokens.



The Animal Farm to the Rescue

Corporate shots fired

Open Source LLAMA from Meta AI

2. Trained on public data but data have not been released.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO ₂ eq)
OPT-175B	400W	809,472	356 MWh	137
BLOOM-175B	400W	1,082,880	475 MWh	183
LLaMA-7B	400W	82,432	36 MWh	14
LLaMA-13B	400W	135,168	59 MWh	23
LLaMA-33B	400W	530,432	233 MWh	90
LLaMA-65B	400W	1,022,362	449 MWh	173

The Animal Farm to the Rescue

A few days later

Open Source LLAMA from Meta AI

3. Non-commercial license and form to request weights
github.com/facebookresearch/llama

Save bandwidth by using a torrent to distribute more efficiently #73

[Open](#) ChristopherKing... wants to merge 3 commits into [facebookresearch:main](#) from [ChristopherKing42:patch-1](#)

Conversation 440 Commits 3 Checks 1 Files changed 1 +1 -1

Changes from all commits File filter Conversations Jump to ... 0 / 1 files viewed Review in codespace Review changes

... README.md @@ -1,7 +1,7 @@ # LLaMA This repository is intended as a minimal, hackable and readable example to load [LLaMA](<https://ai.facebook.com/blog/large-language-model-llama-meta-ai/>) ([arXiv](<https://arxiv.org/abs/2302.13971v1>)) models and run inference. In order to download the checkpoints and tokenizer, fill this [google form](<https://forms.gle/jk851eBVbX1m5TAv5>) or if you want to save our bandwidth use this BitTorrent link: "[magnet:?xt=urn:btih:ZXXDAUWYLRUXXBHUYEMS6Q5CE5WA3LVA&dn=LLaMA]".



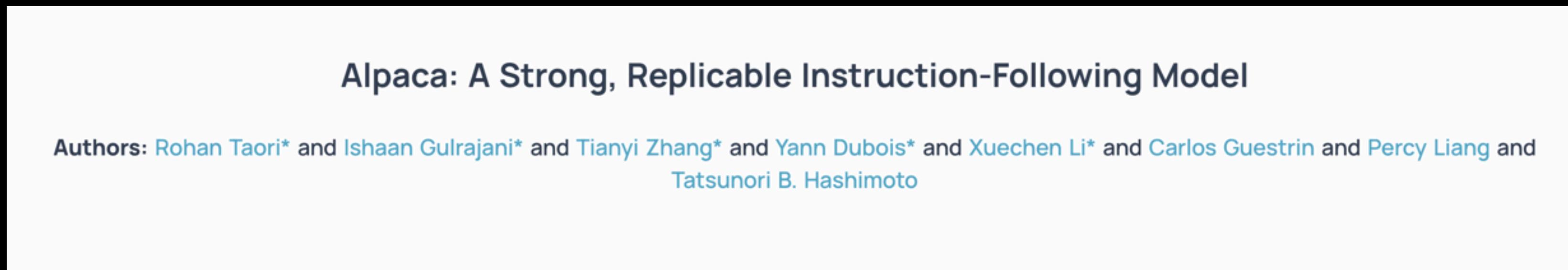


The Animal Farm to the Rescue

A few weeks later

Stanford ignores Open AI TOS and releases Alpaca

1. LLAMA fine-tuned to imitate chat-gpt output over 50k samples
2. Exhibits chat-like abilities.
3. Very awkward licensing.



The Animal Farm to the Rescue

A month later

RedPajama os replication of the LLAMA dataset

1. 1.5 Trillion tokens
2. Wikipedia\Github and much more from Common Crawl
3. Apache 2.0
4. <https://github.com/togethercomputer/RedPajama-Data>



Open Source base LLMs are
solving themselves.
What about instruction tuning?

OpenBugger

Python Data Generation via Static Analysis

- Leverage good code to generate a large amount of debugging instructions.
- Faster to implement a finite number of bugs than to debug infinite bad code.
- Exploit LibCST for precise injection of bugs through Concrete Syntax Tree manipulation.
- Bugs are deterministic, can be chained, and reversed.
- Precise knowledge of bug type, location and how to remove it.



LibCST and Concrete Syntax Trees

Lossless Syntactical Representations of Python Code

- Abstract Syntax Trees (AST) are trees based data structure that describe a string of source code in terms of the operations implemented.
- Concrete Syntax Trees (CST) contains further information like spaces, new-lines, and comments, as such they are amenable to be used in automatic code generation and refactoring.
- LibCST is a Python library developed by Instagram, allowing parsing, analyzing, and modifying Python source code using CSTs. <https://github.com/Instagram/LibCST>

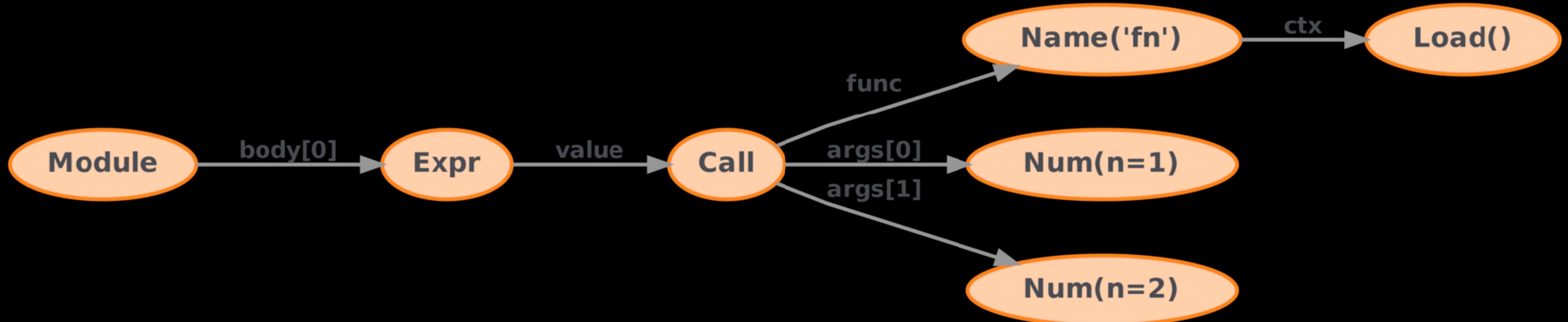


LibCST and Concrete Syntax Trees

Lossless Syntactical Representations of Python Code

```
fn(1, 2) # calls fn
```

→ Abstract Syntax Trees (AST)

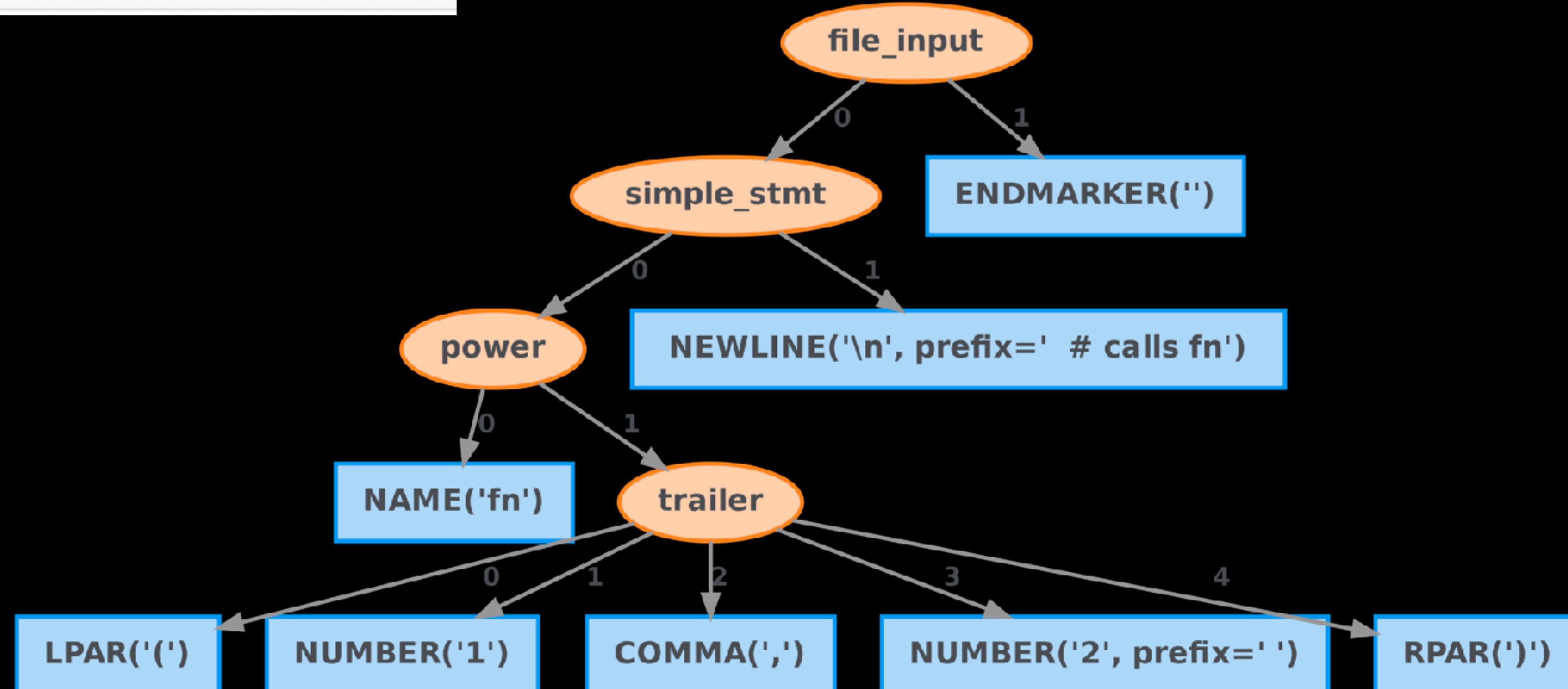


LibCST and Concrete Syntax Trees

Lossless Syntactical Representations of Python Code

```
fn(1, 2) # calls fn
```

→ Concrete Syntax Trees (lib2to3 CST)



LibCST and Concrete Syntax Trees

Lossless Syntactical Representations of Python Code

```
fn(1, 2) # calls fn
```

→ Concrete Syntax Trees (LibCST)



Openbugger CST Bugs Factory

- LibCST matchers to identify candidate targets in the source code.
- The code is modified by a node change operation over the syntax tree.
- LibCST is used to generate the code for the new CST.
- Invertability can be checked empirically by “debugging” the generated code and checking for identity with respect to the source.



Openbugger CST Bugs Factory

Controlflow

- ForgettingToUpdateVariable
- InfiniteWhile
- OffByKIndex,
- IncorrectExceptionHandler
- MissingArgument
- ReturningEarly

Type

- IncorrectType
- NonExistingMethod
- SwapForTransformer

Data

- IncorrectVariableInitialization
- VariableNameTypo
- MutableDefaultArgument
- UseBeforeDefinition

Logical

- ComparisonTarget
- ComparisonSwap
- SwapForTransformer

Bug Example

InfiniteWhile

Control flow bug that sets all while to be always True.

- Targets cst.While nodes
- Modifies the “test” attribute, that is the condition evaluated by the While operator, to be always “True”

```
class InfiniteWhileTransformer(ContextAwareTransformer):  
    METADATA_DEPENDENCIES = (PositionProvider,)  
    def __init__(self, context: CodemodContext):  
        super().__init__(context)  
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"  
    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:  
        self.reverse=reverse  
        return self.transform_module(tree)  
    def leave_While(self, original_node: cst.While, updated_node: cst.While) -> None:  
        meta_pos = self.get_metadata(PositionProvider, original_node)  
        #only updates nodes that are not already in the scratch  
        already_modified = is_modified(original_node,meta_pos,self.context)  
        if not already_modified:  
            updated_node = cst.While(  
                test=cst.Name("True"),  
                body=original_node.body  
            )  
            save_modified(self.context,meta_pos,original_node,updated_node,self.id)  
        return updated_node
```

Bug Example

InfiniteWhile

Control flow bug that sets all while to be always True.

- Targets cst.While nodes
- Modifies the “test” attribute, that is the condition evaluated by the While operator, to be always “True”

```
class InfiniteWhileTransformer(ContextAwareTransformer):  
    METADATA_DEPENDENCIES = (PositionProvider,)  
    def __init__(self, context: CodemodContext):  
        super().__init__(context)  
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"  
    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:  
        self.reverse=reverse  
        return self.transform_module(tree)  
    def leave_While(self, original_node: cst.While, updated_node: cst.While) -> None:  
        meta_pos = self.get_metadata(PositionProvider, original_node)  
        #only updates nodes that are not already in the scratch  
        already_modified = is_modified(original_node,meta_pos,self.context)  
        if not already_modified:  
            updated_node = cst.While(  
                test=cst.Name("True"),  
                body=original_node.body  
            )  
            save_modified(self.context,meta_pos,original_node,updated_node,self.id)  
        return updated_node
```

Bug Example

InfiniteWhile

Control flow bug that sets all while to be always True.

- Targets cst.While nodes
- Modifies the “test” attribute, that is the condition evaluated by the While operator, to be always “True”

```
class InfiniteWhileTransformer(ContextAwareTransformer):  
    METADATA_DEPENDENCIES = (PositionProvider,)  
    def __init__(self, context: CodemodContext):  
        super().__init__(context)  
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"  
    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:  
        self.reverse=reverse  
        return self.transform_module(tree)  
    def leave_While(self, original_node: cst.While, updated_node: cst.While) -> None:  
        meta_pos = self.get_metadata(PositionProvider, original_node)  
        #only updates nodes that are not already in the scratch  
        already_modified = is_modified(original_node,meta_pos,self.context)  
        if not already_modified:  
            updated_node = cst.While(  
                test=cst.Name("True"),  
                body=original_node.body  
            )  
            save_modified(self.context,meta_pos,original_node,updated_node,self.id)  
        return updated_node
```

Bug Example

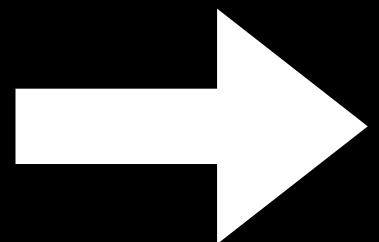
InfiniteWhile

```
while x == 1 + 2 == 3 + 2 != 3 + 4 > 3:  
    y = 1 + 2
```

Bug Example

InfiniteWhile

```
while x == 1 + 2 == 3 + 2 != 3 + 4 > 3:  
    y = 1 + 2
```



```
while True:  
    y = 1 + 2
```

The result of `deep_equals` between the concrete syntax tree of the original and the bugged code is `False`
Checking for bugs...

The following Node has a bug of type `InfiniteWhileTransformer-e885` starting at line 1, column 0 and ending at line 2, column 11.
The bug can be fixed by substituting the bugged code-string `<while True:`

```
y = 1 + 2  
> with the following code-string <while x == 1 + 2 == 3 + 2 != 3 + 4 > 3:  
    y = 1 + 2  
>  
Debugging...  
clean_code  
while x == 1 + 2 == 3 + 2 != 3 + 4 > 3:  
    y = 1 + 2
```

Checking if the debugged code is equal to the original code..

The result of `deep_equals` between the concrete syntax tree of the original and debugged code is `True`

Bug Example

NonExistingMethod

Type bug that calls a wrong class method

- Targets `cst.Call` nodes
- Checks if the call is from an attribute.
- Checks if the attribute is a target attribute.
- Substitutes the attribute with that of another class from the dictionary.

```
class NonExistingMethodTransformer(ContextAwareTransformer):
    METADATA_DEPENDENCIES = (PositionProvider,)

    # Define a mapping from existing methods to non-existing methods
    METHOD_TRANSFORM_MAP: Dict[str, str] = {
        "append": "update", # list to dictionary
        "add": "extend", # set to list
        "update": "add", # dictionary to set
        "extend": "append" # list to list
    }

    def __init__(self, context):
        super().__init__(context)
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"
        self.mutated = False

    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:
        return self.transform_module(tree)

    def leave_Call(self, original_node: cst.Call, updated_node: cst.Call) -> cst.Call:
        if m.matches(updated_node, m.Call(func=m.Attribute())):
            attribute = updated_node.func
            if isinstance(attribute.attr, cst.Name):
                attr_name = attribute.attr.value
                if attr_name in self.METHOD_TRANSFORM_MAP:
                    updated_node = updated_node.with_changes(func=attribute.with_changes(attr=cst.Name(self.METHOD_TRANSFORM_MAP[attr_name])))

        save_modified(self.context, self.get_metadata(PositionProvider, original_node), original_node, updated_node, self.id)
        return updated_node
```

Bug Example

NonExistingMethod

Type bug that calls a wrong class method

- Targets cst.Call nodes
- Checks if the call is from an attribute.
- Checks if the attribute is a target attribute.
- Substitutes the attribute with that of another class from the dictionary.

```
class NonExistingMethodTransformer(ContextAwareTransformer):
    METADATA_DEPENDENCIES = (PositionProvider,)

    # Define a mapping from existing methods to non-existing methods
    METHOD_TRANSFORM_MAP: Dict[str, str] = {
        "append": "update", # list to dictionary
        "add": "extend", # set to list
        "update": "add", # dictionary to set
        "extend": "append" # list to list
    }

    def __init__(self, context):
        super().__init__(context)
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"
        self.mutated = False

    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:
        return self.transform_module(tree)

    def leave_Call(self, original_node: cst.Call, updated_node: cst.Call) -> cst.Call:
        if m.matches(updated_node, m.Call(func=m.Attribute())):
            attribute = updated_node.func
            if isinstance(attribute.attr, cst.Name):
                attr_name = attribute.attr.value
                if attr_name in self.METHOD_TRANSFORM_MAP:
                    updated_node = updated_node.with_changes(func=attribute.with_changes(attr=cst.Name(self.METHOD_TRANSFORM_MAP[attr_name])))

        save_modified(self.context, self.get_metadata(PositionProvider, original_node), original_node, updated_node, self.id)
        return updated_node
```

Bug Example

NonExistingMethod

Type bug that calls a wrong class method

- Targets cst.Call nodes
- Checks if the call is from an attribute.
- Checks if the attribute is a target attribute.
- Substitutes the attribute with that of another class from the dictionary.

```
class NonExistingMethodTransformer(ContextAwareTransformer):  
    METADATA_DEPENDENCIES = (PositionProvider,)  
  
    # Define a mapping from existing methods to non-existing methods  
    METHOD_TRANSFORM_MAP: Dict[str, str] = {  
        "append": "update", # list to dictionary  
        "add": "extend", # set to list  
        "update": "add", # dictionary to set  
        "extend": "append" # list to list  
    }  
  
    def __init__(self, context):  
        super().__init__(context)  
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"  
        self.mutated = False  
  
    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:  
        return self.transform_module(tree)  
  
    def leave_Call(self, original_node: cst.Call, updated_node: cst.Call) -> cst.Call:  
        if m.matches(updated_node, m.Call(func=m.Attribute())):  
            attribute = updated_node.func  
            if isinstance(attribute.attr, cst.Name):  
                attr_name = attribute.attr.value  
                if attr_name in self.METHOD_TRANSFORM_MAP:  
                    updated_node = updated_node.with_changes(func=attribute.with_changes(attr=cst.Name(self.METHOD_TRANSFORM_MAP[attr_name])))  
                    save_modified(self.context, self.get_metadata(PositionProvider, original_node), original_node, updated_node, self.id)  
            return updated_node
```

Bug Example

NonExistingMethod

Type bug that calls a wrong class method

- Targets `cst.Call` nodes
- Checks if the call is from an attribute.
- Checks if the attribute is a target attribute.
- Substitutes the attribute with that of another class from the dictionary.

```
class NonExistingMethodTransformer(ContextAwareTransformer):
    METADATA_DEPENDENCIES = (PositionProvider,)

    # Define a mapping from existing methods to non-existing methods
    METHOD_TRANSFORM_MAP: Dict[str, str] = {
        "append": "update", # list to dictionary
        "add": "extend", # set to list
        "update": "add", # dictionary to set
        "extend": "append" # list to list
    }

    def __init__(self, context):
        super().__init__(context)
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"
        self.mutated = False

    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:
        return self.transform_module(tree)

    def leave_Call(self, original_node: cst.Call, updated_node: cst.Call) -> cst.Call:
        if m.matches(updated_node, m.Call(func=m.Attribute())):
            attribute = updated_node.func
            if isinstance(attribute.attr, cst.Name):
                attr_name = attribute.attr.value
                if attr_name in self.METHOD_TRANSFORM_MAP:
                    updated_node = updated_node.with_changes(func=attribute.with_changes(attr=cst.Name(self.METHOD_TRANSFORM_MAP[attr_name])))

        save_modified(self.context, self.get_metadata(PositionProvider, original_node), original_node, updated_node, self.id)
        return updated_node
```

Bug Example

NonExistingMethod

Type bug that calls a wrong class method

- Targets cst.Call nodes
- Checks if the call is from an attribute.
- Checks if the attribute is a target attribute.
- Substitutes the attribute with that of another class from the dictionary.

```
class NonExistingMethodTransformer(ContextAwareTransformer):  
    METADATA_DEPENDENCIES = (PositionProvider,)  
  
    # Define a mapping from existing methods to non-existing methods  
    METHOD_TRANSFORM_MAP: Dict[str, str] = {  
        "append": "update", # list to dictionary  
        "add": "extend", # set to list  
        "update": "add", # dictionary to set  
        "extend": "append" # list to list  
    }  
  
    def __init__(self, context):  
        super().__init__(context)  
        self.id = f"{self.__class__.__name__}-{uuid.uuid4().hex[:4]}"  
        self.mutated = False  
  
    def mutate(self, tree: cst.Module, reverse: bool = False) -> cst.Module:  
        return self.transform_module(tree)  
  
    def leave_Call(self, original_node: cst.Call, updated_node: cst.Call) -> cst.Call:  
        if m.matches(updated_node, m.Call(func=m.Attribute())):  
            attribute = updated_node.func  
            if isinstance(attribute.attr, cst.Name):  
                attr_name = attribute.attr.value  
                if attr_name in self.METHOD_TRANSFORM_MAP:  
                    updated_node = updated_node.with_changes(func=attribute.with_changes(attr=cst.Name(self.METHOD_TRANSFORM_MAP[attr_name])))  
                    save_modified(self.context, self.get_metadata(PositionProvider, original_node), original_node, updated_node, self.id)  
        return updated_node
```

Bug Example

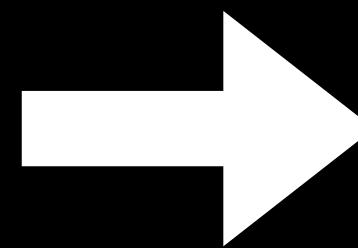
NonExistingMethod

```
my_list = [1, 2, 3]
my_list.append(4)
```

Bug Example

NonExistingMethod

```
my_list = [1, 2, 3]
my_list.append(4)
```



```
my_list = [1, 2, 3]
my_list.update(4)
```

```
The result of deep_equals between the concrete syntax tree of the original and the bugged code is False
Checking for bugs...
```

```
The following Node has a bug of type NonExistingMethodTransformer-7ff6 starting at line 2, column 0 and ending at line 2, column
The bug can be fixed by substituting the bugged code-string <my_list.update(4)> with the following code-string <my_list.append(4>
Debugging...
```

```
clean_code
```

```
my_list = [1, 2, 3]
my_list.append(4)
```

```
Checking if the debugged code is equal to the original code..
```

```
The result of deep_equals between the concrete syntax tree of the original and debugged code is True
```

Automatic Data Generation with OpenBugger

Python Debugging Instruction Tuning

2K LeetCode python exercises with solutions

LibCST parsing and filtering of non syntactic code

20k Invertible Bugs and Debugging Traces

OpenBugger injections on the parsed trees

Question and Answer generation via GPT API

20K Examples of Debugging Operations to train LLM

Automatic Data Generation with OpenBugger

Python Debugging Instruction Tuning

As a beginner, I am having difficulties understanding the code. Can you explain what this piece of code is trying to achieve? Also, what does the variable "gapgap" represent? Lastly, how does this code determine the maximum points for the mice to achieve?

```
def maxPoints(reward1, reward2, k):
    gapgap = sorted([(reward2[i] - reward1[i], i) for i in range(len(reward1))], reverse=True)
    firstMousePoints = sum(reward1[gapgap[i][1]] for i in range(k))
    secondMousePoints = sum(reward2[gapgap[i][1]] for i in range(k, len(reward1)))
    return firstMousePoints + secondMousePoints
```

Automatic Data Generation with OpenBugger

Python Debugging Instruction Tuning

The code is trying to solve a problem where there are two mice and a certain number of cheeses. Each cheese has a reward value associated with it if either mouse eats it. The task is to maximize the total reward points that can be achieved by the mice if the first mouse eats exactly k cheeses. The variable "gappgap" represents a mistake in the code, which is supposed to be "gap". The variable "gap" is a list of tuples where each tuple represents the difference between the reward points of the second mouse and the reward points of the first mouse for a certain cheese and its index. The code then sorts the tuples in descending order and uses the first k tuples to calculate the total reward points for the first mouse. The remaining tuples are used to calculate the total reward points for the second mouse. The sum of these two values is returned as the maximum reward points that can be achieved by the mice.

Bugs:

The variable "gappgap" should be "gap".

The variable "firstMousePointsfirstMousePoints" should be "firstMousePoints".

The variable "secondMousePointsgap" should be "secondMousePoints".

Fixed Code:

```
def maxPoints(reward1, reward2, k):
    gap = sorted([(reward2[i] - reward1[i], i) for i in range(len(reward1))], reverse=True)
    firstMousePoints = sum(reward1[gap[i][1]] for i in range(k))
    secondMousePoints = sum(reward2[gap[i][1]] for i in range(k, len(reward1)))
    return firstMousePoints + secondMousePoints
```

What We Learned

1. Language Models require a LOT of data.
2. Good data is rare and very expensive.
3. OpenBugger cheaply generates good data.

HK3lab

Research Lab

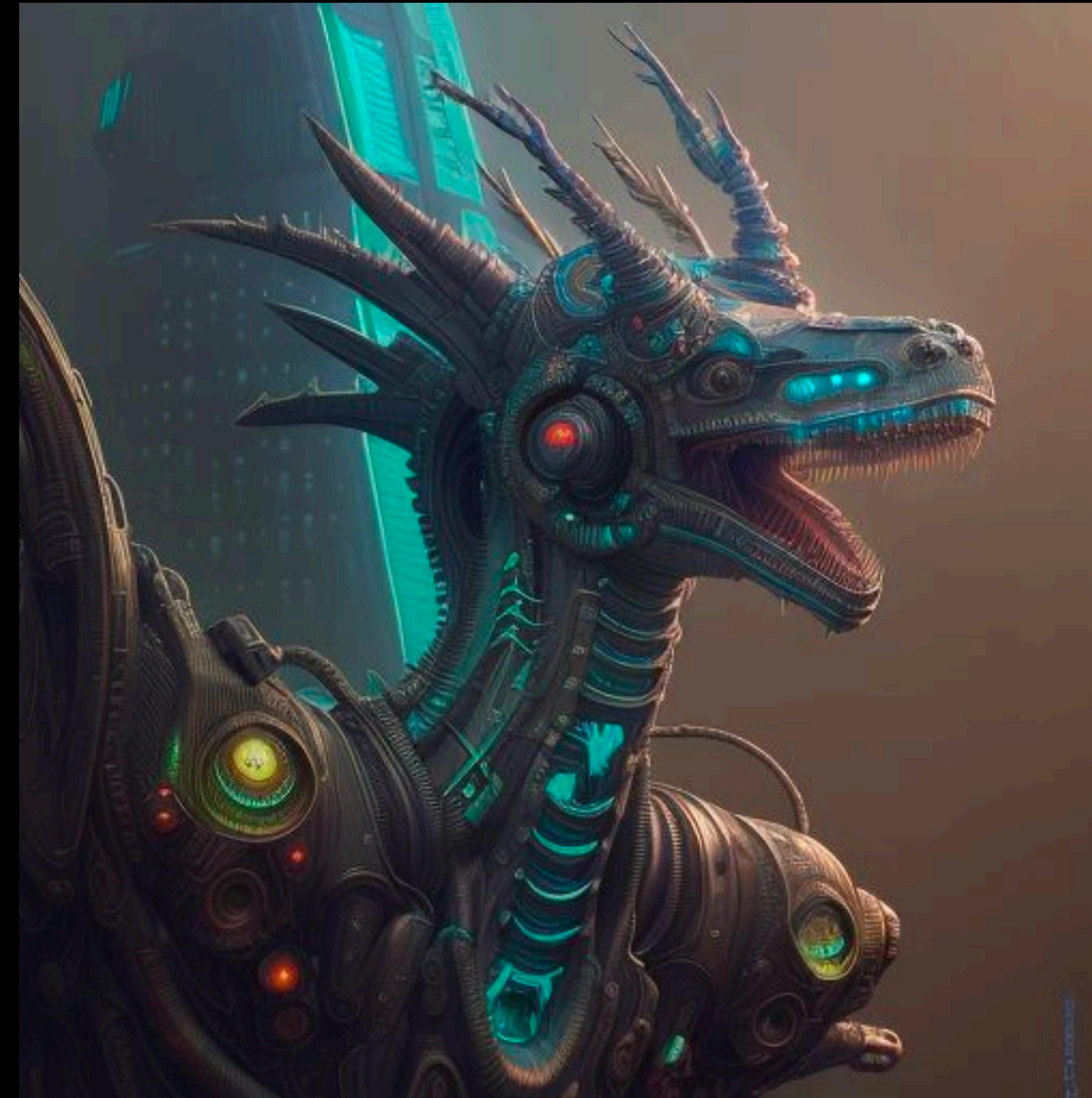
- AI Research Startup in Rovereto (TN)
- Funded by Horizon2020. We are hiring —> tf@hk3lab.ai
 - Metabolic Time Series Forecasting
 - GixaPixel computer vision on histological images



Neurodragon.ai

Open Source Community

- <https://github.com/Neural-Dragon-AI>
- <https://discord.gg/txVHtvDS>
- **BabyDragon:** long term memory management for coding chatbots.
- **Infinipy Engine:** ai first rpg PyGame engine.
- **DragonCodex:** digitalisation of Acadian and Sumerian tablets.



Neurodragon.ai

Open Source Community

- <https://github.com/Neural-Dragon-AI>
- <https://discord.gg/txVHtvDS>
- **Infinipy Engine:** AI first rpg PyGame engine.
- **BabyDragon:** long-term memory management for coding chatbots.
- **DragonCodex:** digitalisation of Acadian and Sumerian tablets.

