香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP 4901B

Large Language Models

# Language Model Bias and Safety

Junxian He

Nov 14, 2025

Part of slides are adapted from CMU 11711

# Recap: Learning of LLM Agents

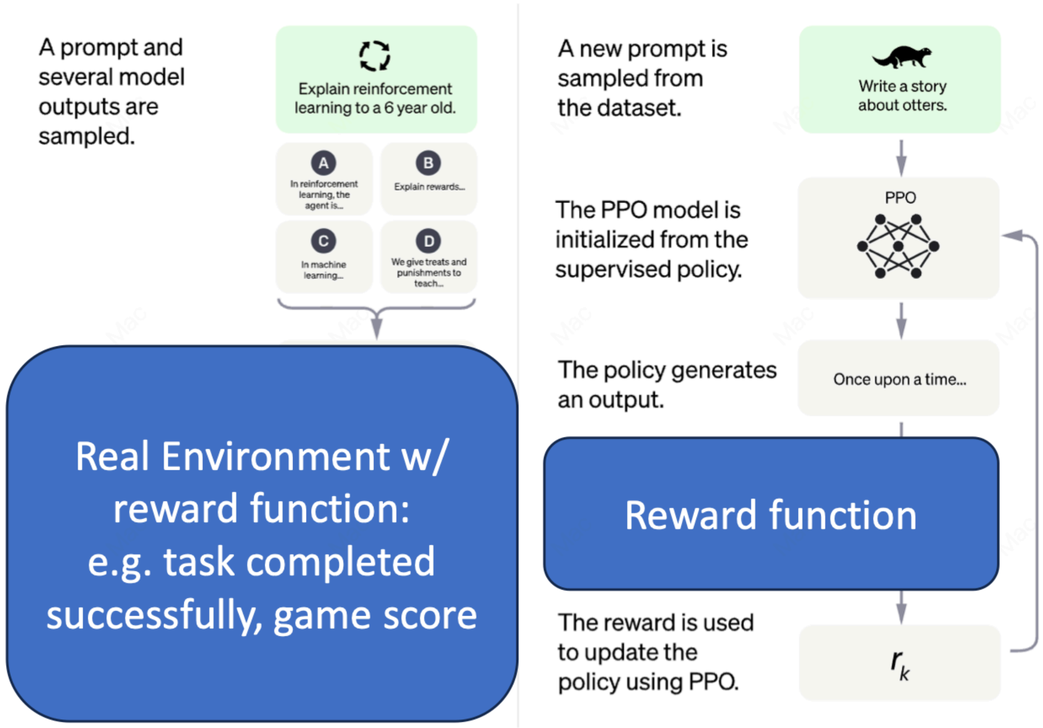
- In-Context Learning – Learning from few-shot exemplars
- Supervised Finetuning – Learning From *Experts*
- Reinforcement Learning – Learning from *Environment*

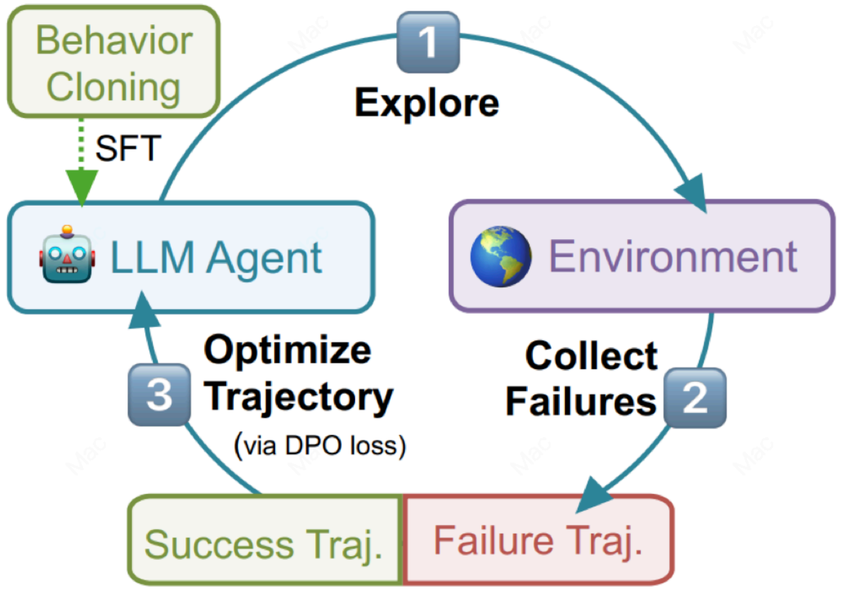
# Recap: Reinforcement Learning

Compared to RLHF:

Given environment, *reward function*

*(trajectory, reward)* pairs without human



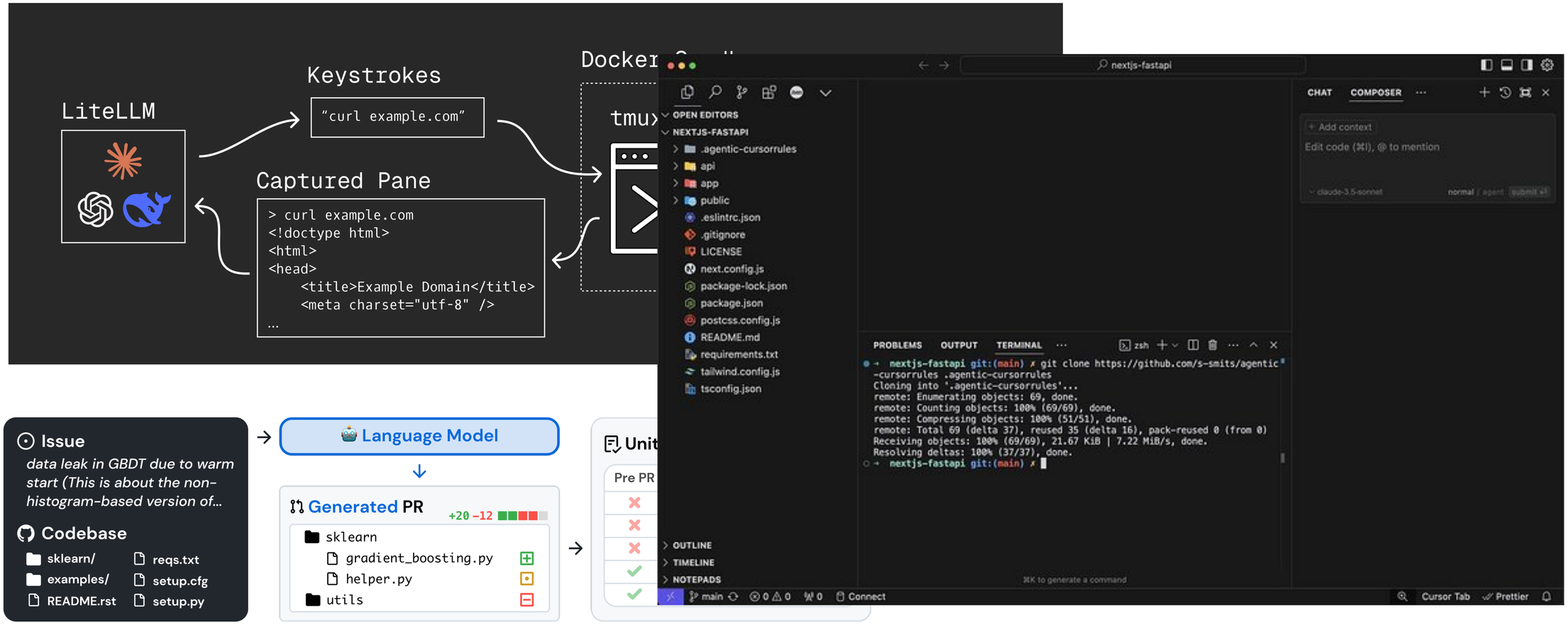

Trial and Error (Song et al. 24')

# Reinforcement Learning

- Closed loop, interactive environment
- Need good reward functions
  - What if the task success/fail is not easy to automatically assess?
- Need good initial models
  - Has decent basic knowledge ability, sparse rewards
- Scalability
  - The environment takes 10 seconds to env.step()
  - The reward function takes 100 seconds to get a scalar reward

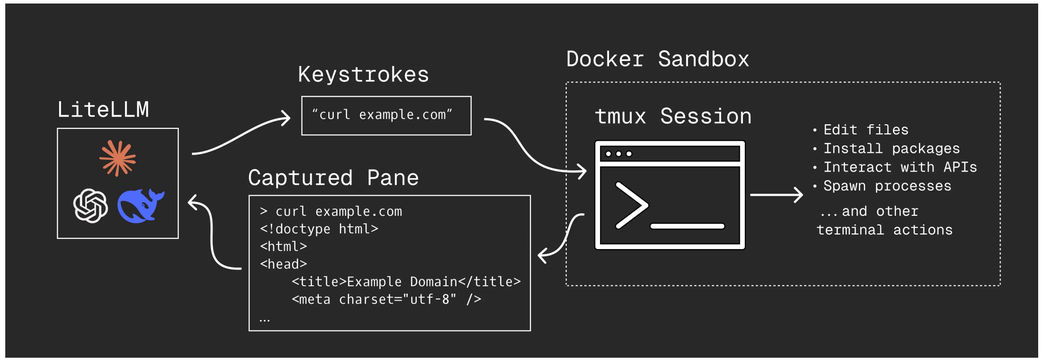
# RL Environments

## Environments and benchmarks typically come together




Research and Products are really close nowadays, and we can directly RL in real, product-level environments

# A Holistic Framework of Agents




Environment

System prompt

Memory management

Default tools

Agent Loop

Query

Env states

Data

# Implement a Live Example to Understand Actual Model Input/Output

```
tokenizer = AutoTokenizer.from_pretrained("Qwen/Qwen3-Coder-30B-A3B-Instruct")

# Define the available tools
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_weather",
            "description": "Get the current weather information for a specified location",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "The city name, e.g. San Francisco, Tokyo"
                    },
                    "unit": {
                        "type": "string",
                        "enum": ["celsius", "fahrenheit"],
                        "description": "The temperature unit to use"
                    }
                },
                "required": ["location"]
            }
        }
    },
    {
        "type": "function",
        "function": {
            "name": "get_current_time",
            "description": "Get the current time for a specified timezone",
            "parameters": {
                "type": "object",
                "properties": {
                    "timezone": {
                        "type": "string",
                        "description": "The timezone identifier, e.g. Asia/Tokyo, America/New_York"
                    }
                },
                "required": ["timezone"]
            }
        }
    }
]
```

Tool Definition

# Implement a Live Example to Understand Actual Model Input/Output

```
conversation = [
    {
        "role": "system",
        "content": "You are a helpful AI assistant with access to tools."
    },
    {
        "role": "user",
        "content": "What's the weather like in San Francisco?"
    },
    {
        "role": "assistant",
        "content": "I'll check the weather in San Francisco for you.",
        "tool_calls": [
            {
                "id": "call_123",
                "type": "function",
                "function": {
                    "name": "get_weather",
                    "arguments": json.dumps({"location": "San Francisco", "unit": "celsius"})
                }
            }
        ]
    },
    {
        "role": "tool",
        "tool_call_id": "call_123",
        "name": "get_weather",
        "content": json.dumps({"temperature": 18, "condition": "partly cloudy", "humidity": 65})
    },
    {
        "role": "assistant",
        "content": "The weather in San Francisco is currently 18°C and partly cloudy with 65% humidity."
    },
    {
        "role": "user",
        "content": "How about in Tokyo? Also, what time is it there?"
    },
    {
        "role": "assistant",
        "content": "Let me check both the weather and time in Tokyo.",
        "tool_calls": [
            {
                "id": "call_456",
                "type": "function",
                "function": {
                    "name": "get_weather",
                    "arguments": json.dumps({"location": "Tokyo", "unit": "celsius"})
                }
            },
```

Example Conversation Data

# Implement a Live Example to Understand Actual Model Input/Output

```
formatted_text = tokenizer.apply_chat_template(
    conversation,
    tools=tools,
    tokenize=False,
    add_generation_prompt=False
)
```

# Live Example after Applying Template


```
[
  {
    "role": "system",
    "content": "You are a helpful AI assistant with access to tools."
  },
  {
    "role": "user",
    "content": "What's the weather like in San Francisco?"
  },
  {
    "role": "assistant",
    "content": "I'll check the weather in San Francisco for you.",
    "tool_calls": [
      {
        "id": "call_123",
        "type": "function",
        "function": {
          "name": "get_weather",
          "arguments": "{\"location\": \"San Francisco\", \"unit\": \"celsius\"}"
        }
      }
    ]
  },
  {
    "role": "tool",
    "tool_call_id": "call_123",
    "name": "get_weather",
    "content": "{\"temperature\": 18, \"condition\": \"partly cloudy\", \"humidity\": 65}"
  },
  {
    "role": "assistant",
    "content": "The weather in San Francisco is currently 18\u00b0C and partly cloudy with 65% humidity."
  },
  {
    "role": "user",
    "content": "How about in Tokyo? Also, what time is it there?"
  },
  {
    "role": "assistant",
    "content": "Let me check both the weather and time in Tokyo.",
    "tool_calls": [
      {
        "id": "call_456",
        "type": "function",
        "function": {
          "name": "get_weather",
          "arguments": "{\"location\": \"Tokyo\", \"unit\": \"celsius\"}"
        }
      },
      {
        "id": "call_789",
        "type": "function",
        "function": {
          "name": "get_current_time",
          "arguments": "{\"timezone\": \"Asia/Tokyo\"}"
        }
      }
    ]
  },
```


Json formatted context

# Live Example after Applying Template

```
<|im_start|>system
You are a helpful AI assistant with access to tools.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{
  "type": "function",
  "function": {
    "name": "get_weather",
    "description": "Get the current weather information for a specified location",
    "parameters": {
      "type": "object",
      "properties": {
        "location": {
          "type": "string",
          "description": "The city name, e.g. San Francisco, Tokyo"
        },
        "unit": {
          "type": "string",
          "enum": [
            "celsius",
            "fahrenheit"
          ],
          "description": "The temperature unit to use"
        }
      },
      "required": [
        "location"
      ]
    }
  }
}
{
  "type": "function",
  "function": {
    "name": "get_current_time",
    "description": "Get the current time for a specified timezone",
    "parameters": {
      "type": "object",
      "properties": {
        "timezone": {
          "type": "string",
          "description": "The timezone identifier, e.g. Asia/Tokyo, America/New_York"
        }
      },
      "required": [
        "timezone"
      ]
    }
  }
}
</tools>
```

Qwen2.5-coder tokenizer

After applying chat templates

System prompt

# Live Example after Applying Template

```
For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call><|im_end|>
<|im_start|>user
What's the weather like in San Francisco?<|im_end|>
<|im_start|>assistant
I'll check the weather in San Francisco for you.
<tool_call>
{
  "name": "get_weather",
  "arguments": "{\"location\": \"San Francisco\", \"unit\": \"celsius\"}"
}
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
{
  "temperature": 18,
  "condition": "partly cloudy",
  "humidity": 65
}
</tool_response><|im_end|>
<|im_start|>assistant
The weather in San Francisco is currently 18°C and partly cloudy with 65% humidity.<|im_end|>
<|im_start|>user
How about in Tokyo? Also, what time is it there?<|im_end|>
<|im_start|>assistant
Let me check both the weather and time in Tokyo.
<tool_call>
{
  "name": "get_weather",
  "arguments": "{\"location\": \"Tokyo\", \"unit\": \"celsius\"}"
}
</tool_call>
<tool_call>
{
  "name": "get_current_time",
  "arguments": "{\"timezone\": \"Asia/Tokyo\"}"
}
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
{
  "temperature": 15,
  "condition": "clear",
  "humidity": 45
}
</tool_response>
<tool_response>
{
  "time": "14:30",
  "timezone": "Asia/Tokyo",
  "date": "2025-01-15"
```

After applying chat templates

# Live Example after Applying Template

```
<|im_start|>system
You are a helpful AI assistant.

# Tools

You have access to the following functions:

<tools>
<function>
<name>get_weather</name>
<description>Get the current weather information for a specified location</description>
<parameters>
<parameter>
<name>location</name>
<type>string</type>
<description>The city name, e.g. San Francisco, Tokyo</description>
</parameter>
<parameter>
<name>unit</name>
<type>string</type>
<description>The temperature unit to use</description>
<enum>["celsius", "fahrenheit"]</enum>
</parameter>
<required>["location"]</required>
</parameters>
</function>
<function>
<name>get_current_time</name>
<description>Get the current time for a specified timezone</description>
<parameters>
<parameter>
<name>timezone</name>
<type>string</type>
<description>The timezone identifier, e.g. Asia/Tokyo, America/New_York</description>
</parameter>
<required>["timezone"]</required>
</parameters>
</function>
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>
```

Qwen3-Coder tokenizer template

Switch from json tool call to xml

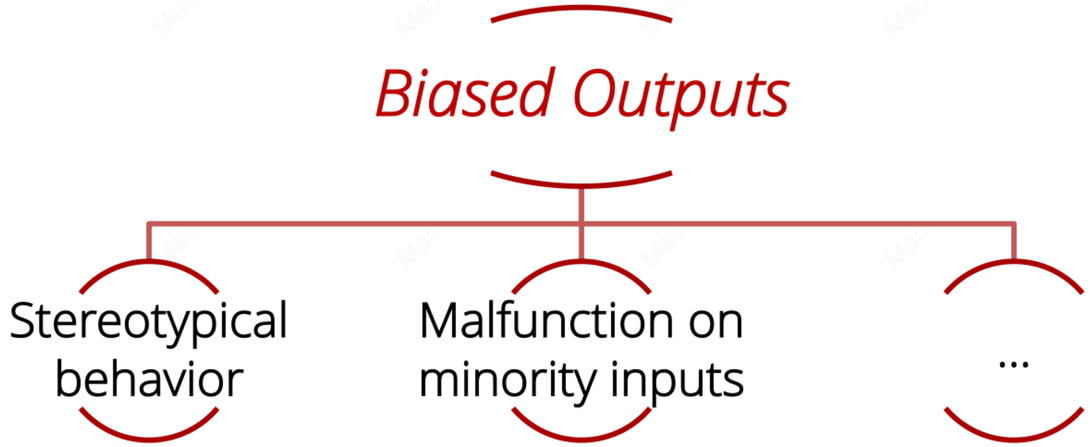
# Live Example after Applying Template

```
<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function> block must be nested within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE the function call, but NOT after
- If there is no function call available, answer the question like normal with your current knowledge and do not tell the user about function calls
</IMPORTANT><|im_end|>
<|im_start|>user
What's the weather like in San Francisco?<|im_end|>
<|im_start|>assistant
I'll check the weather in San Francisco for you.

<tool_call>
<function=get_weather>
<parameter=location>
San Francisco
</parameter>
<parameter=unit>
celsius
</parameter>
</function>
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
{
  "temperature": 18,
  "condition": "partly cloudy",
  "humidity": 65
}
</tool_response>
<|im_end|>
<|im_start|>assistant
The weather in San Francisco is currently 18°C and partly cloudy with 65% humidity.<|im_end|>
<|im_start|>user
How about in Tokyo? Also, what time is it there?<|im_end|>
<|im_start|>assistant
Let me check both the weather and time in Tokyo.

<tool_call>
<function=get_weather>
<parameter=location>
Tokyo
</parameter>
<parameter=unit>
celsius
</parameter>
</function>
</tool_call>
<tool_call>
<function=get_current_time>
<parameter=timezone>
Asia/Tokyo
</parameter>
</function>
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
```

Qwen3-Coder tokenizer template

Switch from json tool call to xml

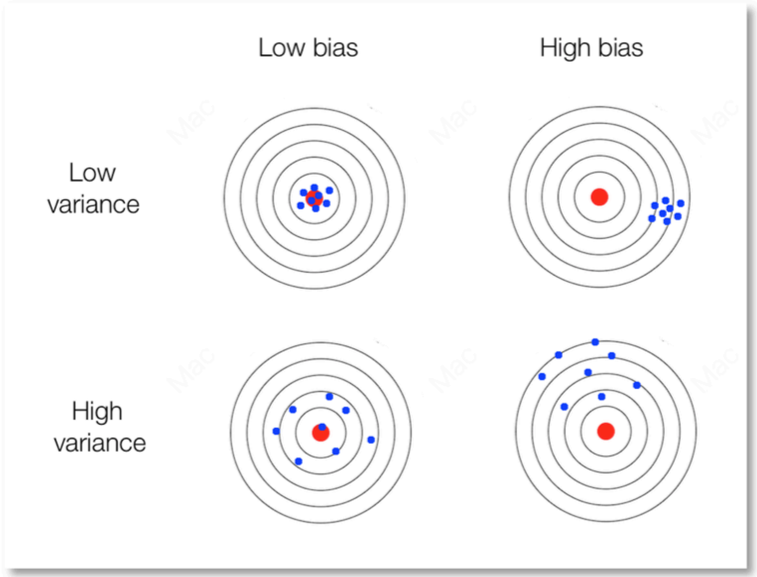
# LLM Safety and Bias


Biased Outputs
Stereotypical behavior
Malfunction on minority inputs
...


Harmful Content
Toxicity
Unsafe content
...

# Some Definition of Bias

- Bias [*statistics*]: systematic tendency causing differences between model estimates / predictions

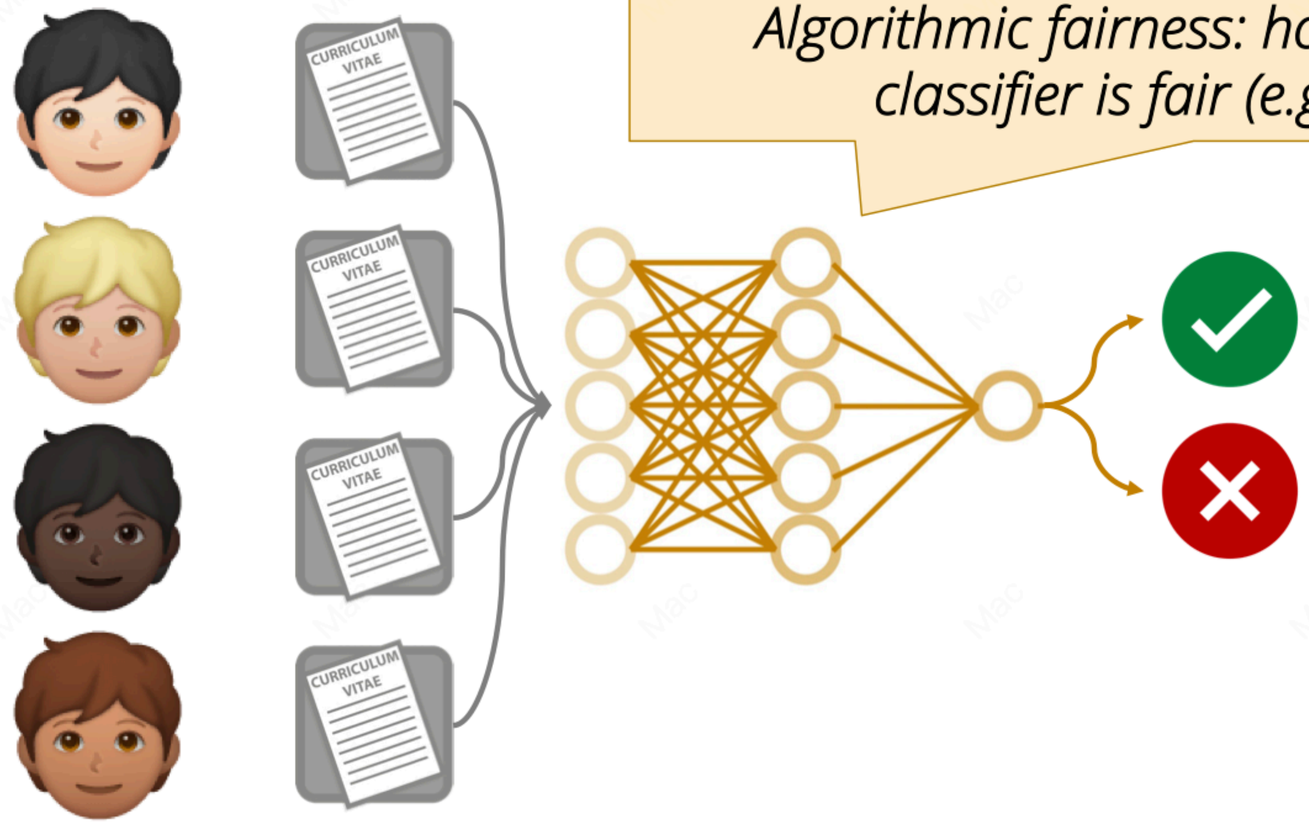



- Bias [*general*]: "*disproportionate weight in favor of or against an idea or thing, usually in a way that is closed-minded, prejudicial, or unfair*" – Wikipedia

Presence of bias ≃ absence of fairness
Algorithmic fairness: attempts to correct biases in ML systems
But… how is fairness defined?

# Algorithmic Fairness

Let's assume a toy task: given a resumé, predict whether a candidate is qualifiec



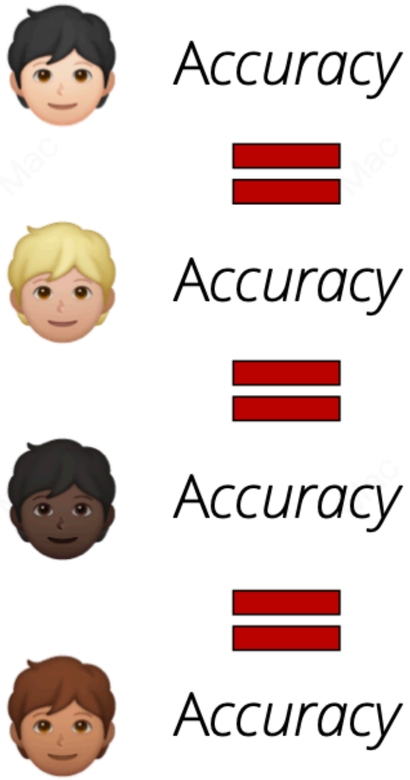

*Algorithmic fairness: how do you know if your classifier is fair (e.g., w.r.t. ethnicity)?*

Evaluation:
*Accuracy*
*Precision*
*Recall*
*F-1 score*

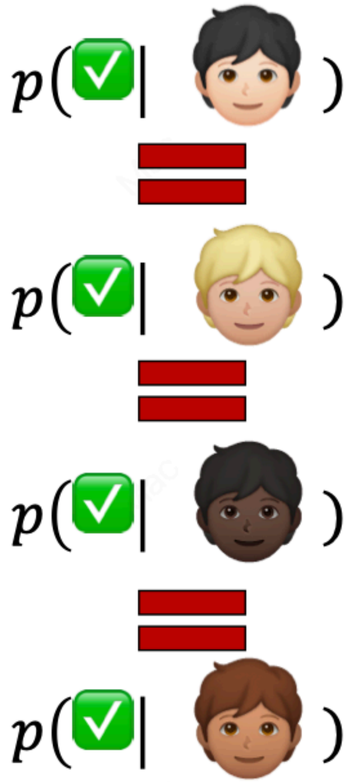
# Fairness Metrics

To control variables, let's use the same set of resume, but only change the candidates' ethnic group information on the resume

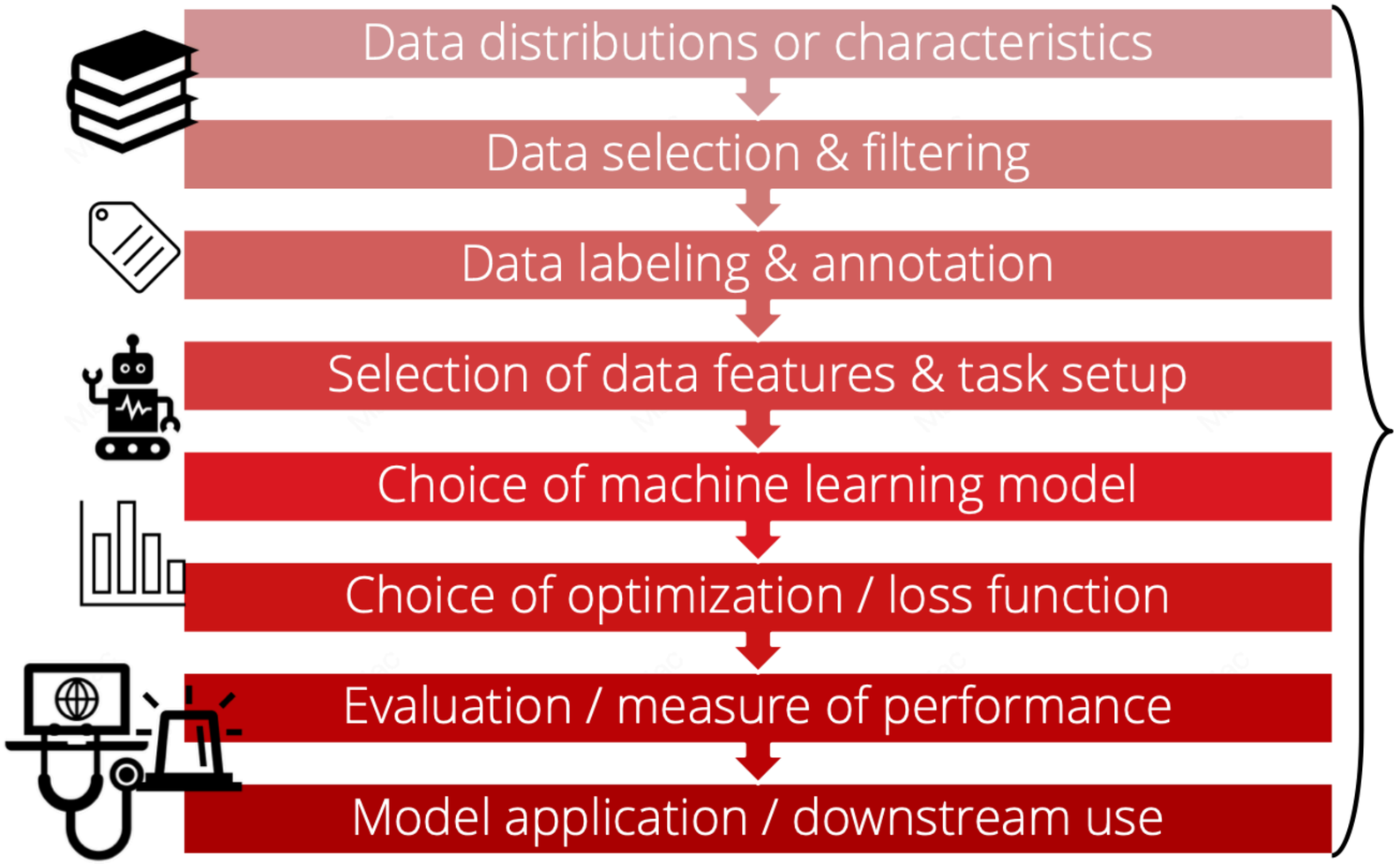
- Accuracy quality: a classifier is fair if the people from different groups have the same accuracy




- Statistical parity: groups should have the same probability of being assigned positive class



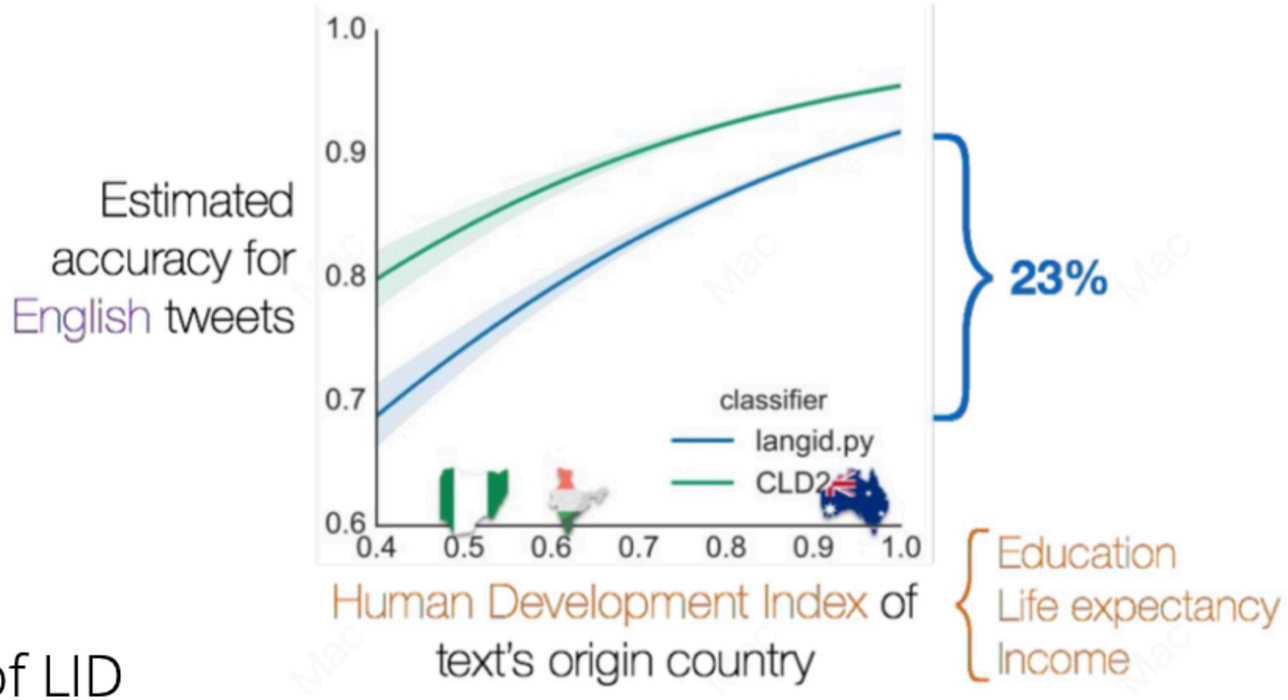
# Where Does Bias Come From?




Bias can arise from *any* of these design decisions

# Example of Bias from Data: LangID Tool

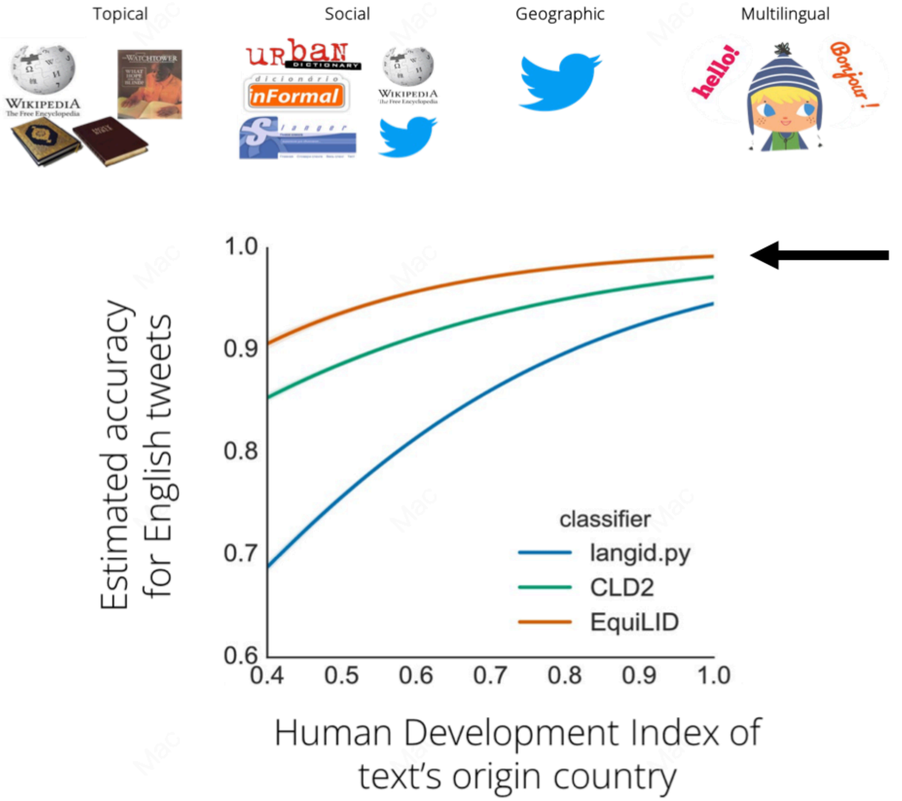
- LangID task: determine which language an input text is in
  - Considered a "*a solved problem* suitable for undergraduate instruction" (McNamee, 2005)
- Often a first step in most NLP and CSS preprocessing pipelines
  - e.g., filtering LLM pretraining data
- *But*, many variations of English in the world
  - *Int'l*: Nigerian English, Indian English, etc.
  - *Within US*: African American English, etc.
- Jurgens et al. (2017) found that accuracy of LID tool correlated with wealth/development level of country; works worse for low HDI countries




Jurgens et al. Incorporating Dialectal Variability for Socially Equitable Language Identification. 2017

# Example of Bias from Data: LangID Tool

- Jurgens et al (2017) introduce EquiLID
  - Trained by sampling more variety of data, topically, socially, geographically diverse, and even multilingual data
- Find that tool works much better than original LID systems
  - Bonus: even improved accuracy on highly developed countries!



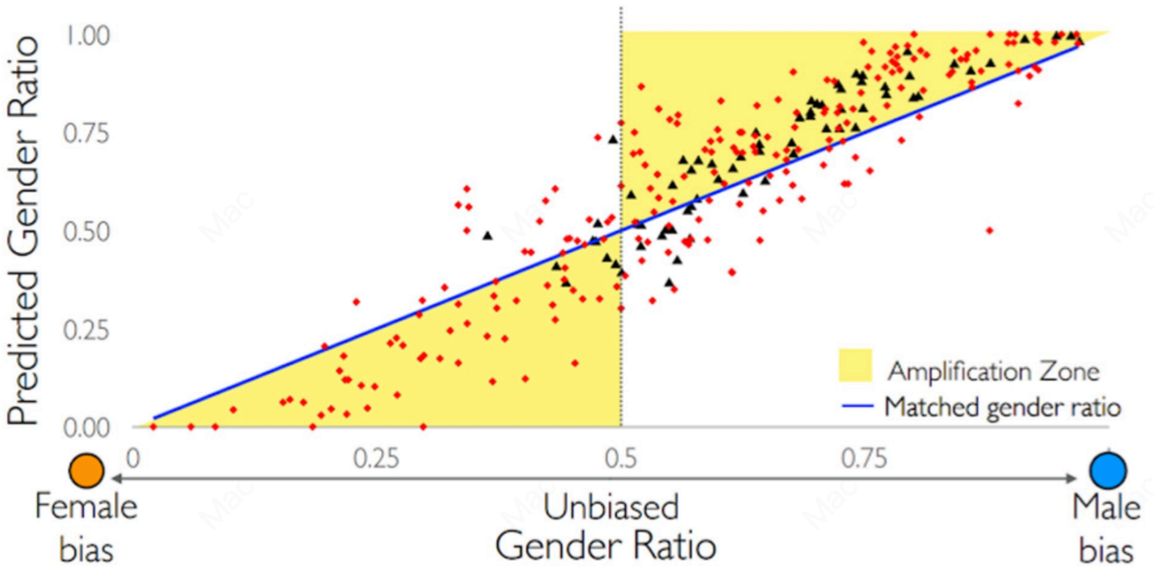

Takeaway: Bias can be mitigated by making better data choices

But this is not the only source….

Jurgens et al. Incorporating Dialectal Variability for Socially Equitable Language Identification. 2017

# Bias amplification from models

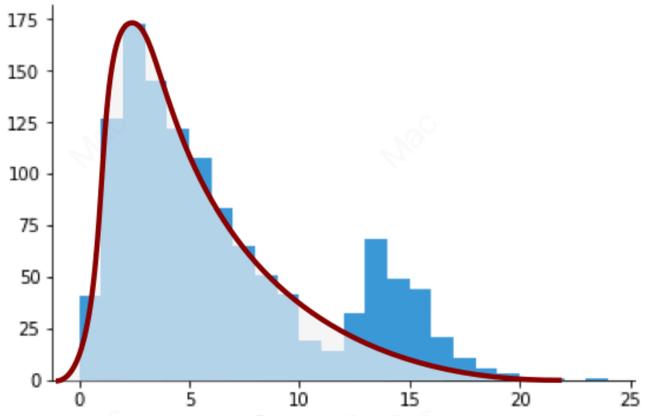
- Zhao et al (2017) examined visual semantic role labeling task
  - Given an image, predict various semantic roles, including agent (person doing the action)
- Found skews in training dataset
  - E.g., 66% of training cooking images had agent=woman
- Found that models *amplified* biases
  - E.g., 84% of test cooking images predicted as agent=woman (~18% men mis-labeled)
- Showed that prediction / inference functions can mitigate this bias




Zhao et al. Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints. 2017

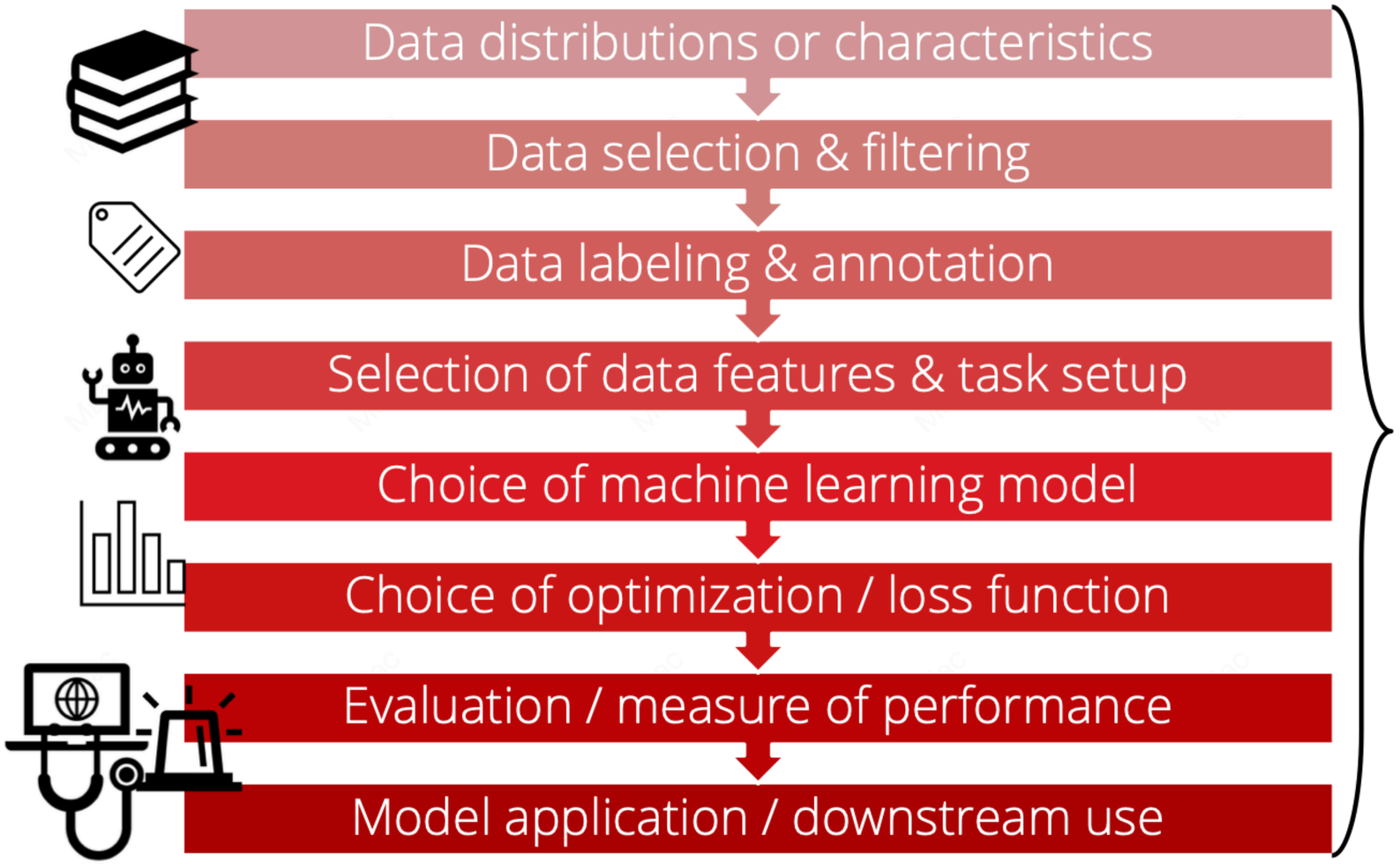
# Model Biases: Mathematical Links

- *Competing losses*: objective functions aim to minimize loss globally → learns to predict most frequent class
  - Often at the expense of less frequent classes (e.g. minority groups)




Suppose you have a machine translation dataset covering multiple languages, but most of them are about English-Spanish translation, and some of them are about other languages. Models will prioritize decrease losses of English-Spanish translation

# Where Does Bias Come From?

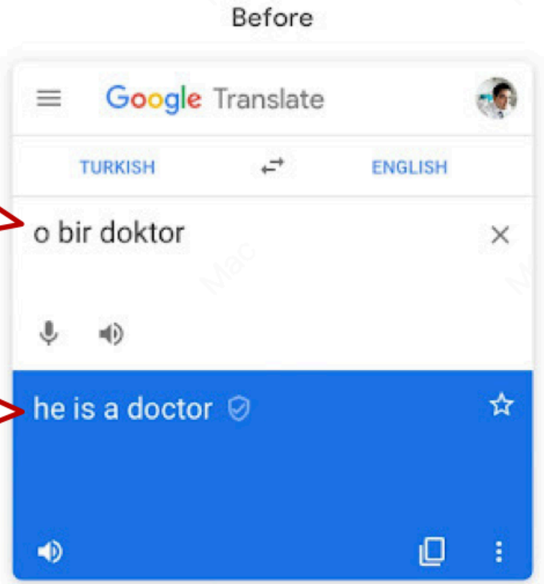


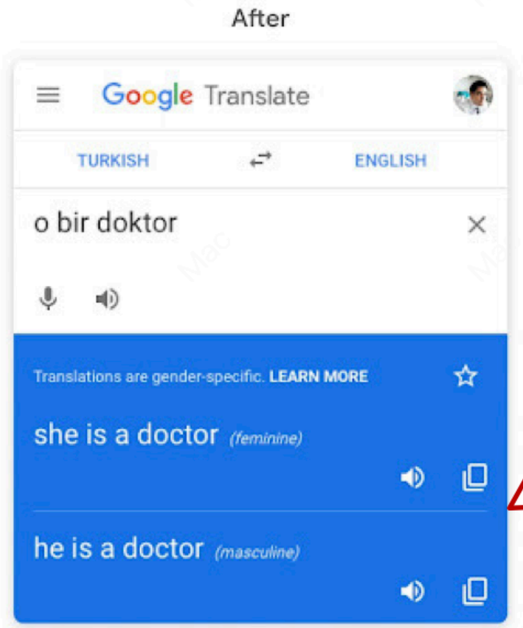

Bias can arise from *any* of these design decisions

# Google Translate Issue

Input in non-gendered language (e.g., Turkish)

Stereotypical gender assigned in translation

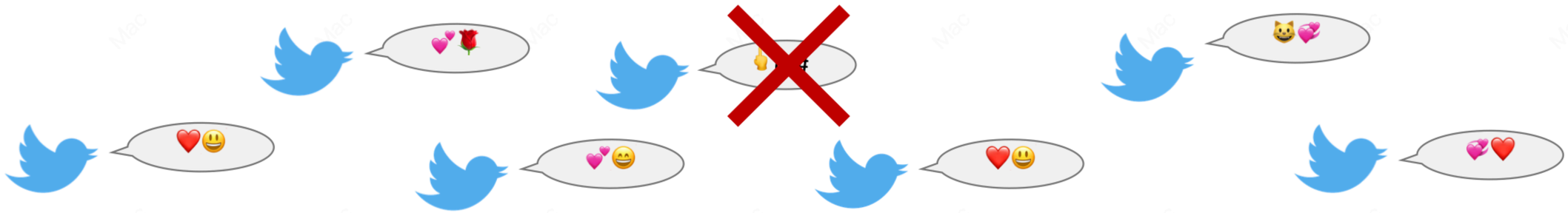



Google's fix: translate twice, with male & female pronouns

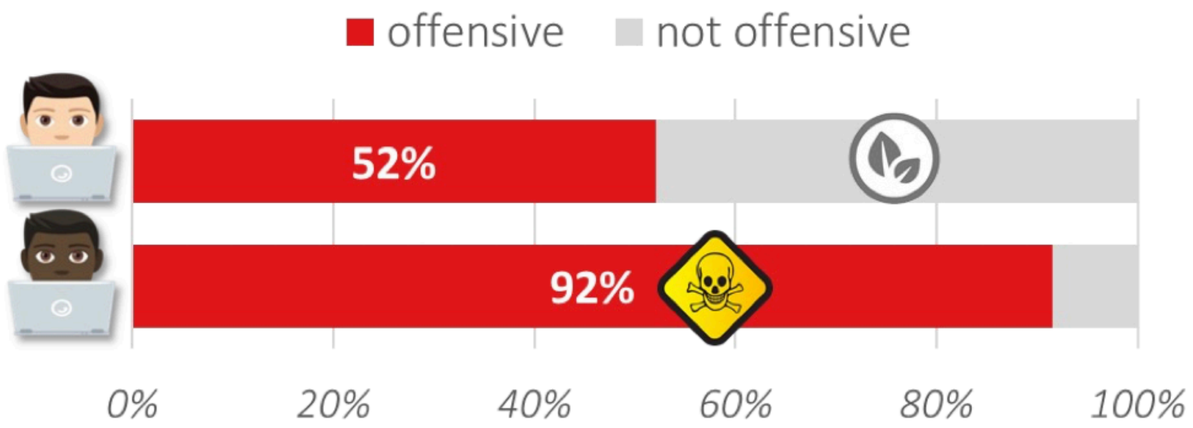
- Takeaways: mitigating bias may involve *system-level changes* to UI, input processing, output formatting, etc. while underlying AI model is similar

# Hate Speech or Toxic Language Detection



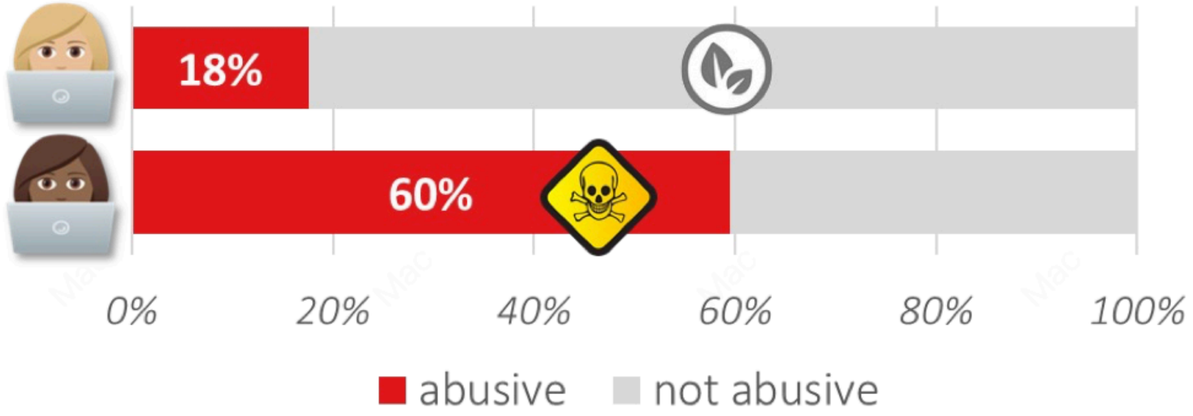
Goal: find and flag hateful or toxic content online, to make the internet less toxic

# Racial biases in popular datasets




TWT-HATEBASE
(Davidson et al., 2017)

Both datasets have biases w.r.t. AAE tweets




TWT-BOOTSTRAP
(Founta et al., 2018)

# Why did these biases occur? Why didn't NLP system designers think about these issues beforehand

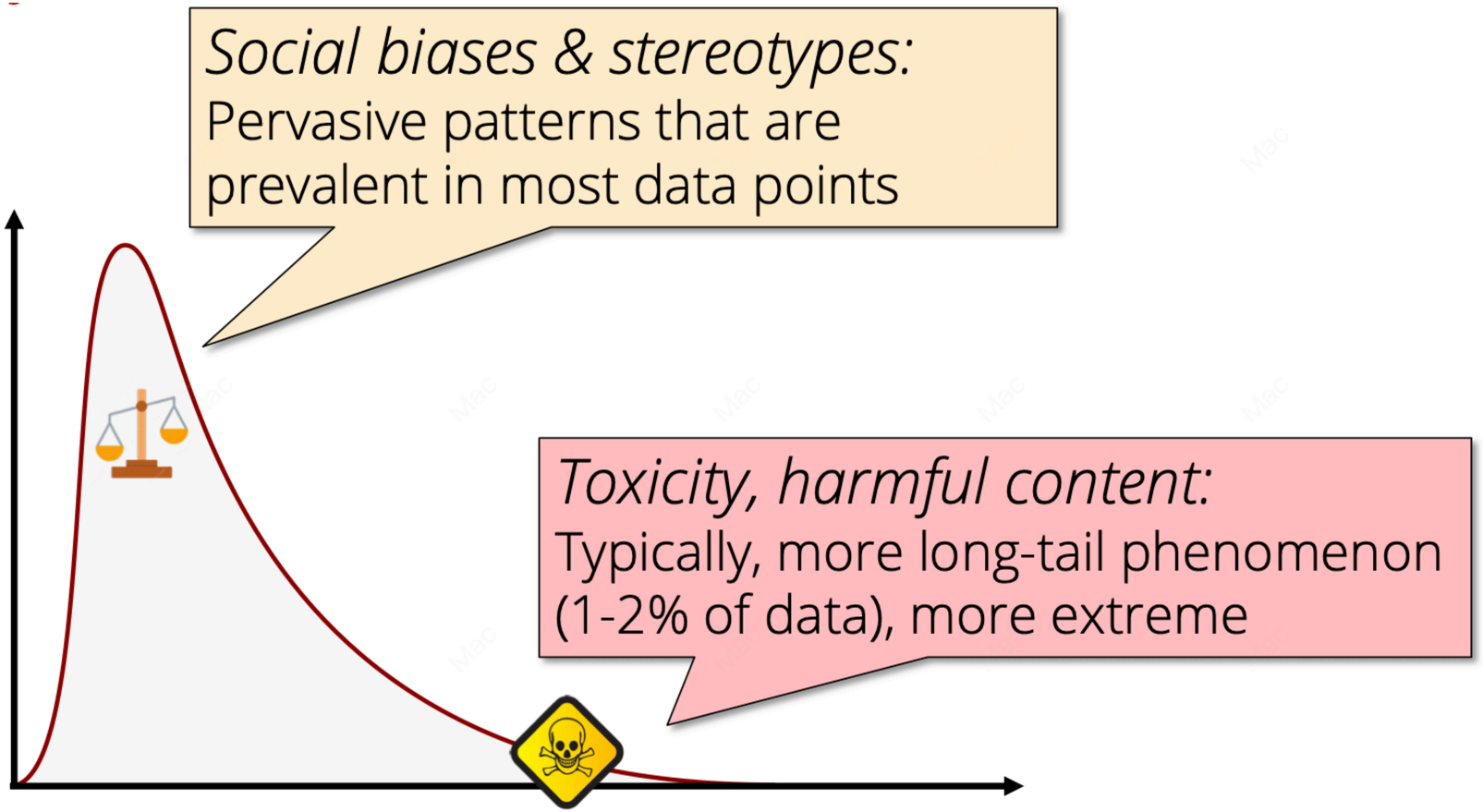
The world itself is biased


System designers have our own biases because of their *positionality*, i.e., set of perspectives that we hold due to our lived experiences and identity.

Positionality affects all our choices (e.g., assuming 1-1 mapping between languages and gendered pronouns, assuming toxicity looks the same in different dialects)

# Harmful Content & Toxicity

# Biases vs. Toxicity



# Toxicity in LLMs

- Gehman et al (2020) introduced concept of *neural toxic degeneration* in LLMs

- Out of a 100 generations sampled from models, at least one toxic sentence
  - 65-70% toxicity from GPT2, GPT3
  - 85% toxicity from GPT1

- Model size affects toxicity: larger models have more toxicity [Touvron et al 2023]

Gehman et a. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. 2020

Touvron et a. LLaMA: Open and Efficient Foundation Language Models. 2023

# Why are these models learning so much undesirable content?

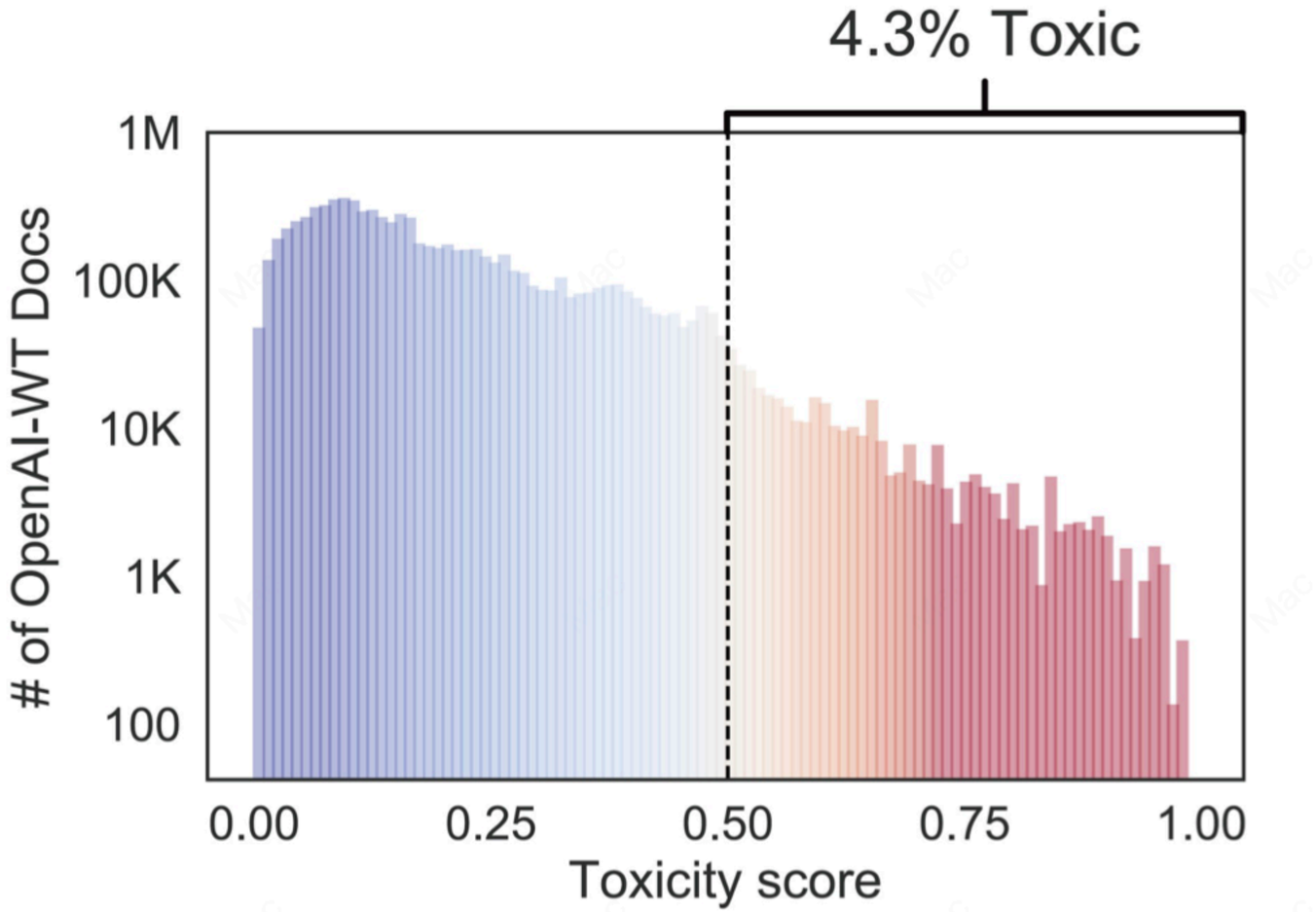
# Problems with Self-Supervised Pretraining

*"Feeding AI systems on the world's **beauty, ugliness, and cruelty**, but expecting it to reflect only the beauty is a fantasy"*

*Prof. Ruha Benjamin, PhD*

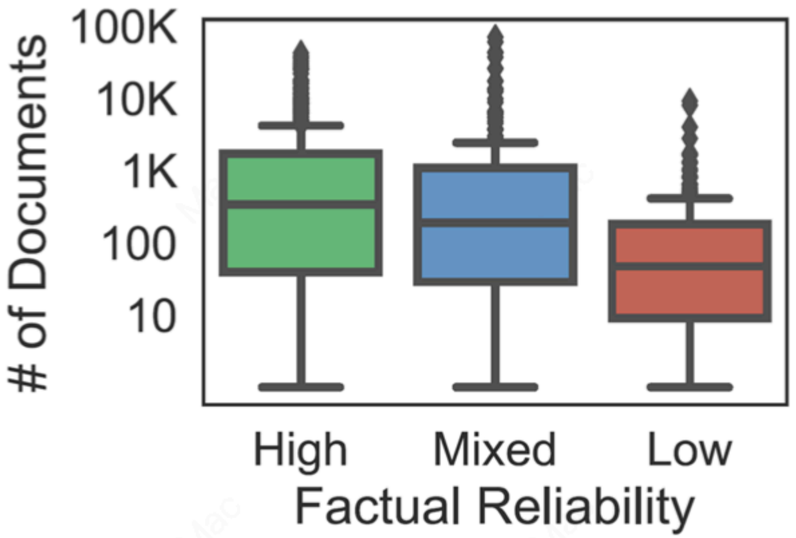
- Recipe: scrape as much pretraining data as you can to train your LM
- Consequence: LM ends up learning toxicity, biases, extremism, hate speech…

# Toxicity in GPT-2's pretraining data



# Fake news in GPT-2's Pretraining data

- Also looked at sources of documents in training data
- Cross-referencing sources of documents with known factual reliability categorization
  - >272K (3.4%) docs from low/mixed reliability sources
- Examining source where document is shared
  - >200K (3%) docs linked from banned/quarantined subreddits, which typically are more toxic docs
- Important to examine training data
  - Can only do that if publicly released!



# Thank You!