School of Cities Workshop

https://github.com/upadhyan/SoC-AI-Workshop

About Me

Nakul Upadhya

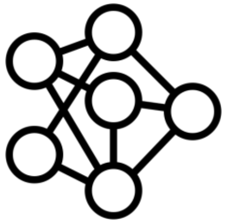
Nakul.Upadhya@mail.utoronto.ca

The Optimization and Machine Learning (OptiMaL) Lab

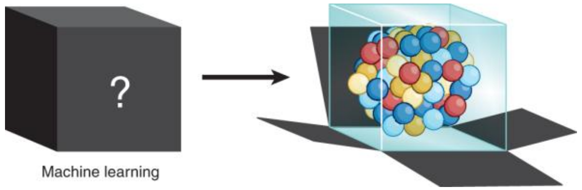
https://optimal.mie.utoronto.ca/

Research Areas

Deep Learning

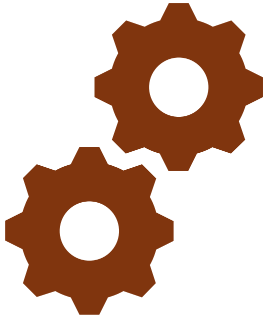


XAI



Applications

Manufacturing



Healthcare

Agenda

- AI/ML/LLM Primer
- NLP Lab
- Break
- Explainability in AI
- Explainability Lab
- Geospatial Analysis Lab

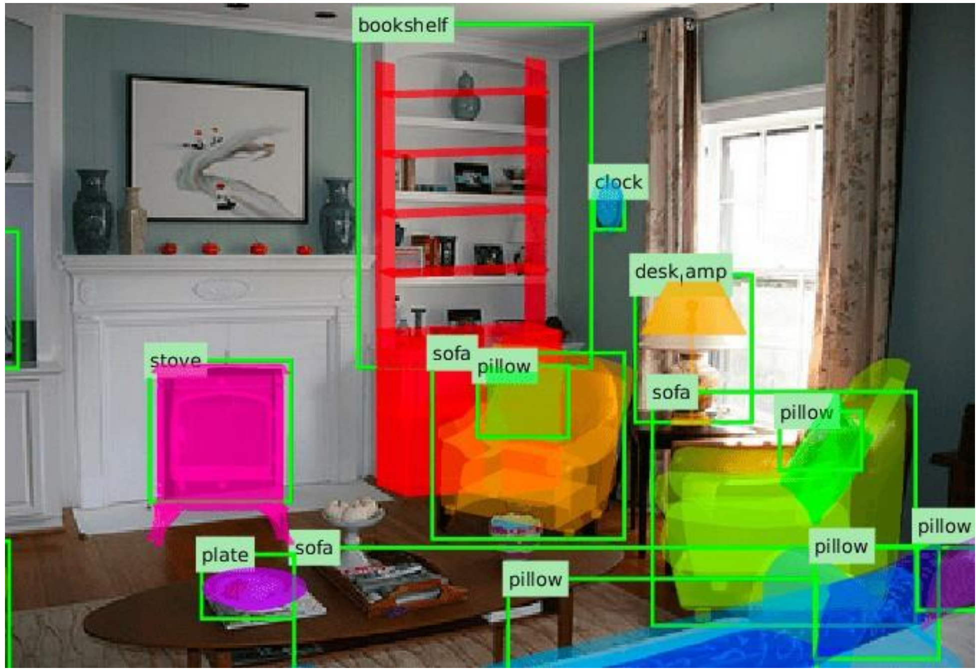
AI, ML and LLM Primer

Slides by Alex Olson and Nakul Upadhya

Presented by Nakul Upadhya

Artificial Intelligence

- Getting computers to behave intelligently:
    - Perform non-trivial tasks as well as humans do
    - Perform tasks that even humans struggle with
- Many sub-goals:
    - Perception
    - Reasoning
    - Control
    - Planning



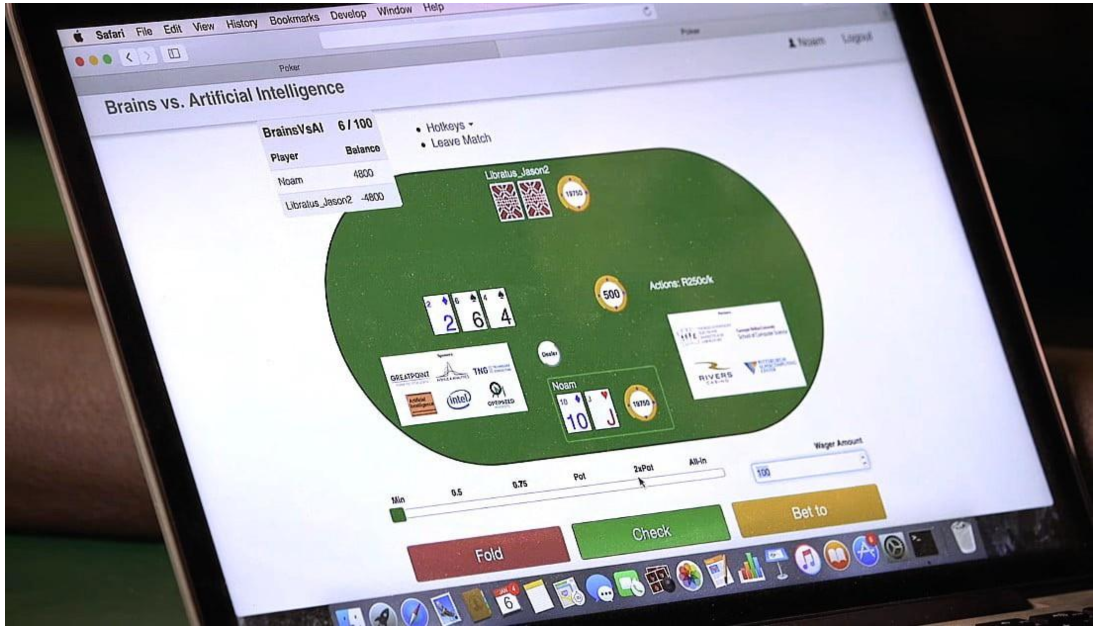
My poker face: AI wins multiplayer game for first time

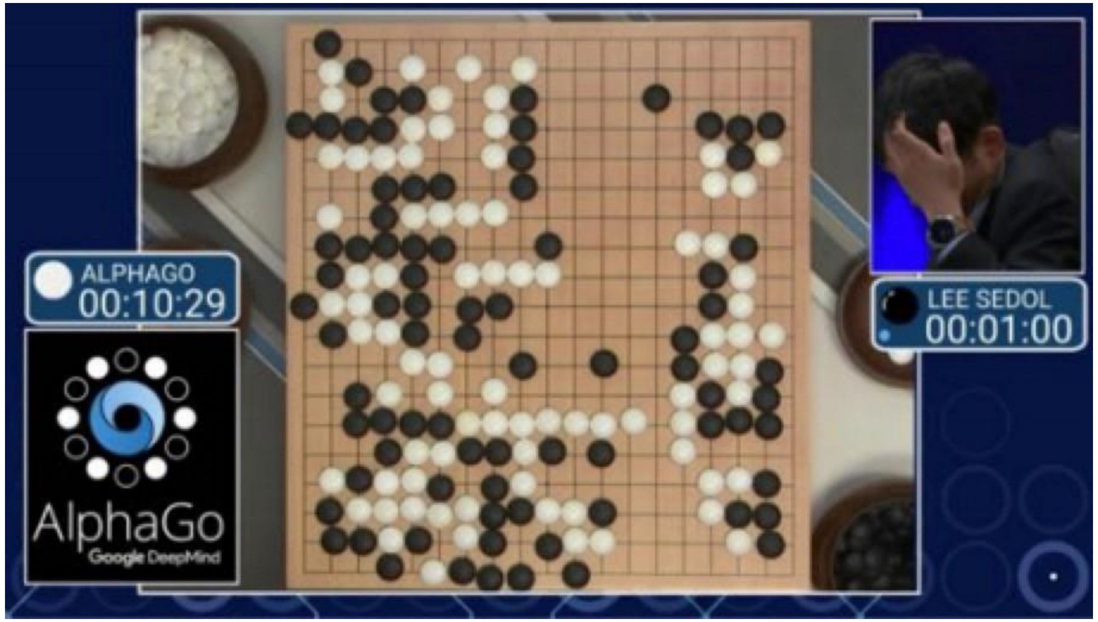
Pluribus wins 12-day session of Texas hold'em against some of the world's best human players

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Centre for Analytics and Artificial Intelligence Engineering

Speech Recognition: Perception + Reasoning

Game Playing: Reasoning + Planning





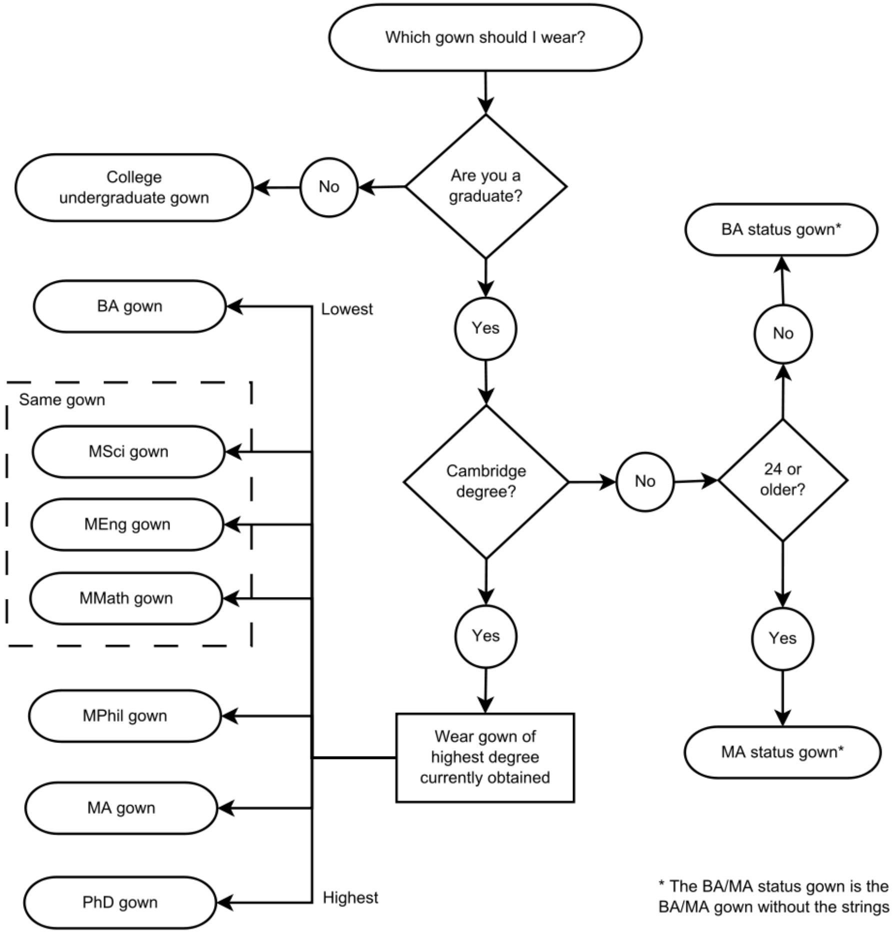
Autonomous Driving: Perception + Reasoning Control + Planning

Knowledge-Based AI

Write programs that simulate how people solve the problem

Fundamental limitations:

- Will never get better than a person
- Requires deep domain knowledge
- We don't know how we do some things (e.g., riding a bicycle)



Data-Based AI = Machine Learning

Write programs that learn the task from examples

✅ No need to know how we do it as humans
✅ Performance should improve with more examples

❌ May need many examples!
❌ May not understand how the program works!

Machine Learning

- Study of algorithms that
  - Improve their <u>performance</u> P
  - At some <u>task</u> T
  - With <u>experience</u> E
- Well defined learning task: <P,T,E>

The Machine Learning Process

- Study of algorithms that
  - Improve their performance P
  - At some task T
  - With experience E
- Well defined learning task: <P,T,E>

- Experience
  - Examples of the form

    (input, correct output)
- Task
  - Mapping from input to output
- Performance
  - ”Loss function” that measures error w.r.t. desired outcome

Choices in ML Problem Formulation

- Experience
  - Examples of the form (input, correct output)
- Task
  - Mapping from input to output
  - Performed by a “model”
- Performance
  - ”Loss function” that measures error w.r.t. desired outcome

Loan Applications

- What historical examples do I have? What is a correct output?
- Predict probability of default? Loan decision? Credit score?
- Do I care more about minimizing False Positives? False negatives?

What is a “model”?

- A **useful approximation** of the world.
- Maps between inputs $\mathbf{x}$ and outputs $f(\mathbf{x} \mid a, b, c, \ldots)$.
- Examples:
  - Input: Age, Income, Debt.
  - Output: Credit Score
- Typically, there are **many reasonable models** for the same data
- **Training** a model = finding appropriate values for (a,b,c,…)
  - An **optimization** problem
  - “appropriate” = **minimizes the Loss (cost)** function

Example: Linear Regression

Dependent Variable
(Response Variable)

Independent Variables
(Predictors)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \varepsilon$$

Y intercept

Slope
Coefficient

Error Term

- Learn the weights that minimize the error.

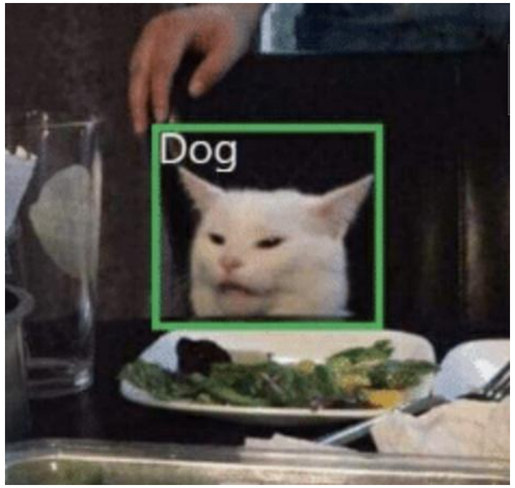
Classification Loss (Error) Function

- How unhappy are you with the answer that the model gave?

- $\mathbf{L_{0\text{-}1}(y, f(x))} = 1 \quad$ if: $\mathbf{y \neq f(x)}$
  $\phantom{\mathbf{L_{0\text{-}1}(y, f(x))} =} 0 \quad$ otherwise

- **0-1 loss** function: intuitive but hard to optimize = train

- In practice, we use:
  - **approximations** of the 0-1 loss – getting warmer or getting colder
  - Loss functions based on probability



Why should this work at all?

The main theoretical basis of ML:

With a **sufficient amount of "similar" data**

+

an **expressive model class**:

Minimizing the loss function on the training data yields a highly accurate model on **unseen test data**, with high probability

1. Data: $S = \{(x_i, y_i)\}_{i = 1,...,n}$
   - x_i: data example with d attributes
   - y_i: label of example (what you care about)
2. Model: a function $f_{(a,b,c,...)}$
   - Maps from X to Y
   - (a,b,c,…) are the parameters
3. Loss function: L(y, f(x))
   - Penalizes the model's mistakes

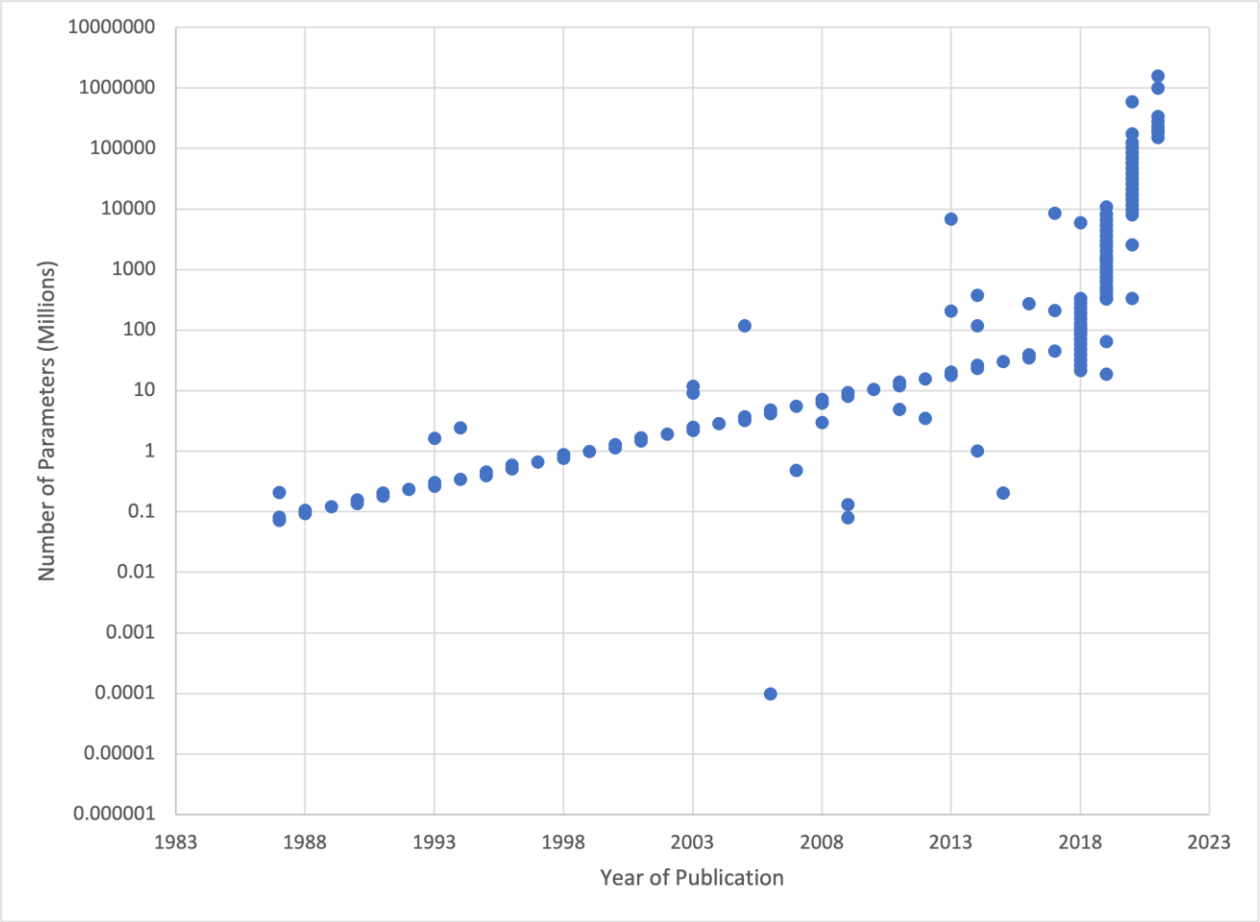
Language Modeling

- Language Modeling is a classification task!
- Goal: Predict the next word.
  - More specifically, predict the probability that the next word is y.
- Data:
  - Attributes: An incomplete Sentence or prompt.
  - Label: the next word/token.
  - Instances: Lots and lots of examples.

Centre for Analytics and Artificial Intelligence Engineering

<u>Large</u> Language Models

- Latest models are capable of learning from much more data due to:
  - Hardware Improvements.
  - Availability of data.
  - Willingness to spend $$.



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Centre for Analytics and Artificial Intelligence Engineering
Centre for Analytics and Artificial Intelligence Engineering

How do LLMs work?

- We’ll begin with a *user prompt*: the message that the user sends to the system.

- `User: What’s the capital of France?`

How do LLMs work?

- We'll begin with a *user prompt*: the message that the user sends to the system.
- Our system doesn't just receive this prompt. It also receives a *system prompt*, which primes the model to behave how we'd like.

- `System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.`
- `User: What's the capital of France?`

How do LLMs work?

- We'll begin with a *user prompt*: the message that the user sends to the system.
- Our system doesn't just receive this prompt. It also receives a *system prompt*, which primes the model to behave how we'd like.
- Tools like ChatGPT may also supply "memories" about the user, or user-defined instructions.

- `System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.`
- `Memories:`
  - `2024-04-08 User asked for recommendations on modern philosophy. Recommended "The History of Philosophy" by A.C. Grayling`
  - `2024-03-15 User reported trouble installing Python libraries on a Mac. Explained how to use Pip and Homebrew to install Python packages.`
- `User Profile:`
  - `Name: Alex`
  - `Profession: Senior Research Associate`
  - `Interaction Style: professional and concise`
- `User: What's the capital of France?`

All of this is the "X"

How do LLMs work?

- System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.
- Memories:
  - 2024-04-08 User asked for recommendations on modern philosophy. Recommended “The History of Philosophy” by A.C. Grayling
  - 2024-03-15 User reported trouble installing Python libraries on a Mac. Explained how to use Pip and Homebrew to install Python packages.
- User Profile:
  - Name: Alex
  - Profession: Senior Research Associate
  - Interaction Style: professional and concise
- User: What’s the capital of France?

- Output: Paris

And this is the “y”

How do LLMs work?

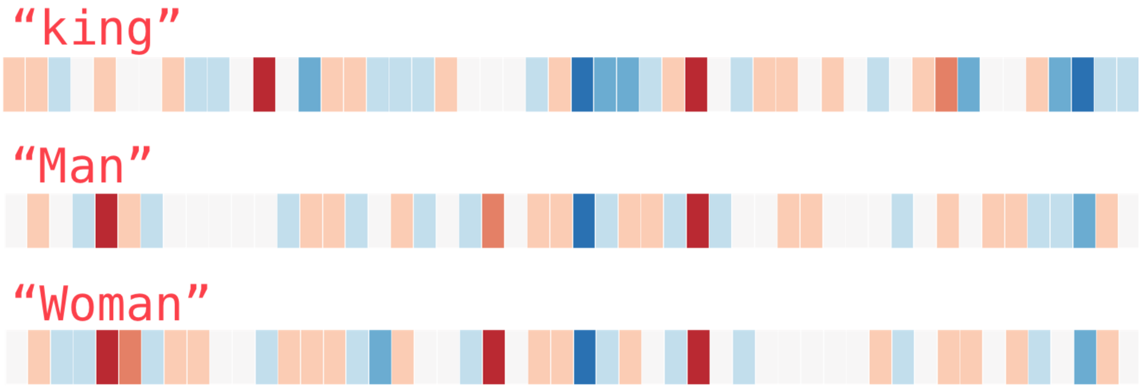
- System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.
- Memories:
  - 2024-04-08 User asked for recommendations on modern philosophy. Recommended “The History of Philosophy” by A.C. Grayling
  - 2024-03-15 User reported trouble installing Python libraries on a Mac. Explained how to use Pip and Homebrew to install Python packages.
- User Profile:
  - Name: Alex
  - Profession: Senior Research Associate
  - Interaction Style: professional and concise
- User: What’s the capital of France?

- Output: Paris

How do we make this prediction?

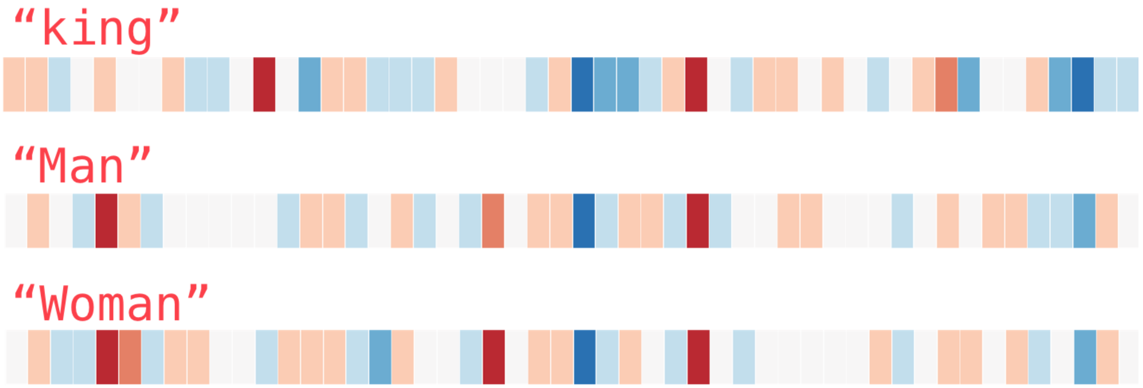
Word Embeddings:

- Problem: Computers operate using numbers: how does a model understand text?
- Solution: Encode tokens as **vectors of numbers** that the model can work with.



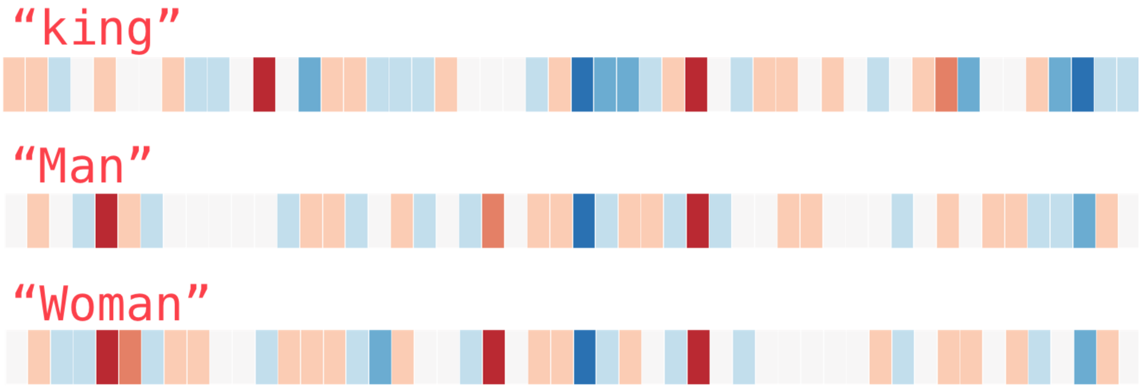
Embeddings:

- Problem: Computers operate using numbers: how does a model understand text?
- Solution: Encode tokens as **vectors of numbers** that the model can work with.
- We call these vectors **Embeddings.**
  - Word2Vec: 300 features
  - GPT-3: 12,888 features

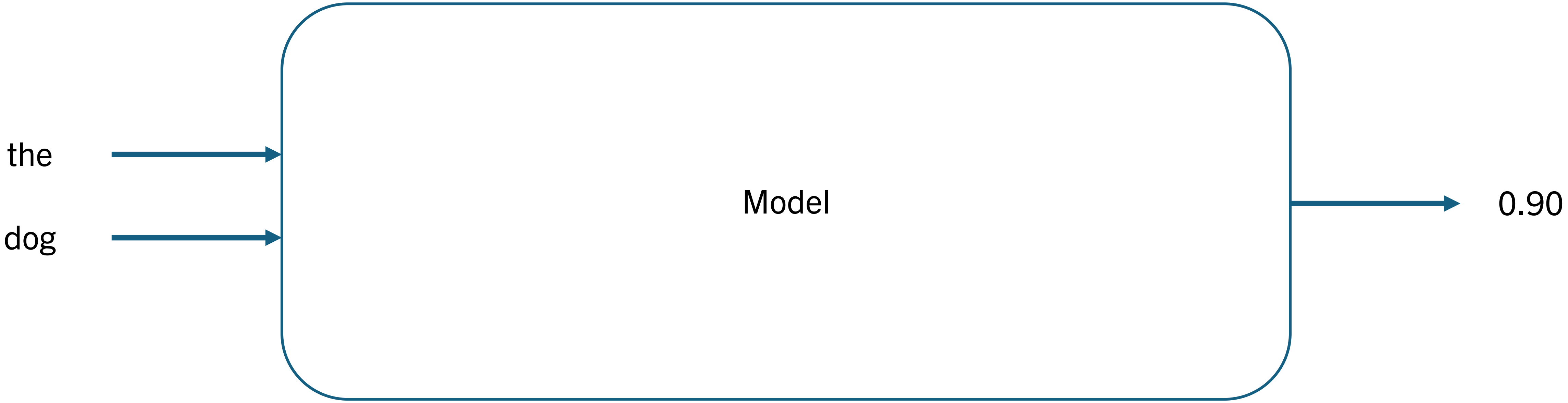


How does an LM “understand” word meaning?

- Problem:
  - What numbers do we fill the embedding with?
  - How do we make these embeddings contain information?
- Solution:
  - We learn the numbers for each vector .
  - Key concept of word embeddings: similar words should have similar vectors.
  - How do we accomplish this?



Building Word Embeddings



Building Word Embeddings

the

dog

Look at word embeddings

$$\underset{the}{\begin{pmatrix} 0.4 \\ 0.3 \\ -0.7 \\ \vdots \\ -0.1 \end{pmatrix}} \quad \underset{dog}{\begin{pmatrix} -0.2 \\ 0.2 \\ -0.1 \\ \vdots \\ 0.2 \end{pmatrix}}$$

Compute similarity

$$\text{similarity}(the, dog)$$

0.90

Building Word Embeddings

the

puppy

Look at word embeddings

$$\underset{the}{\begin{pmatrix} 0.4 \\ 0.3 \\ -0.7 \\ \vdots \\ -0.1 \end{pmatrix}} \quad \underset{puppy}{\begin{pmatrix} -0.3 \\ 0.1 \\ -0.1 \\ \vdots \\ 0.2 \end{pmatrix}}$$

Compute similarity

$$\text{similarity}(the, puppy)$$

0.87

Building Word Embeddings

king

dog

Look at word embeddings

$$\begin{pmatrix} 0.7 \\ 0.9 \\ -0.8 \\ \vdots \\ -0.5 \end{pmatrix} \quad \begin{pmatrix} -0.2 \\ 0.2 \\ -0.1 \\ \vdots \\ 0.2 \end{pmatrix}$$

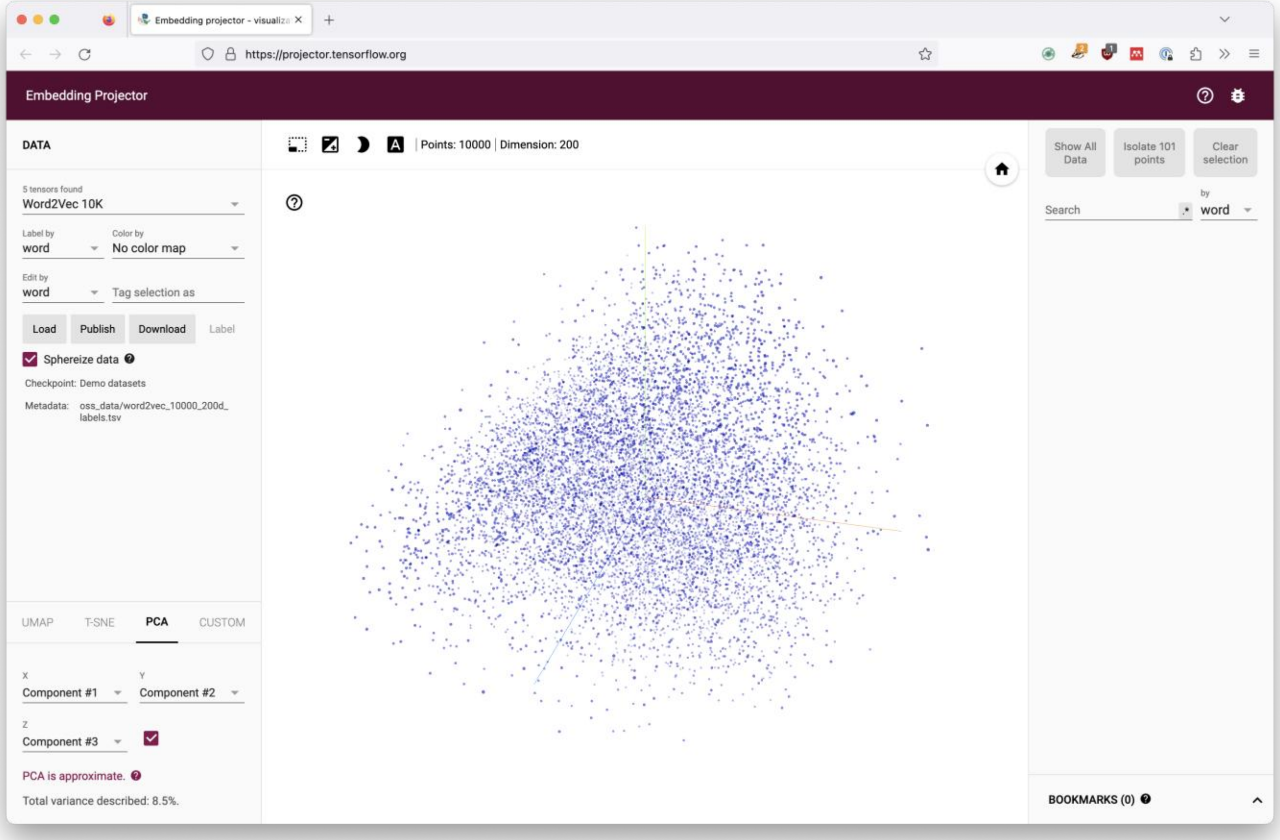
king *dog*

Compute similarity

$$\text{similarity}(king, dog)$$

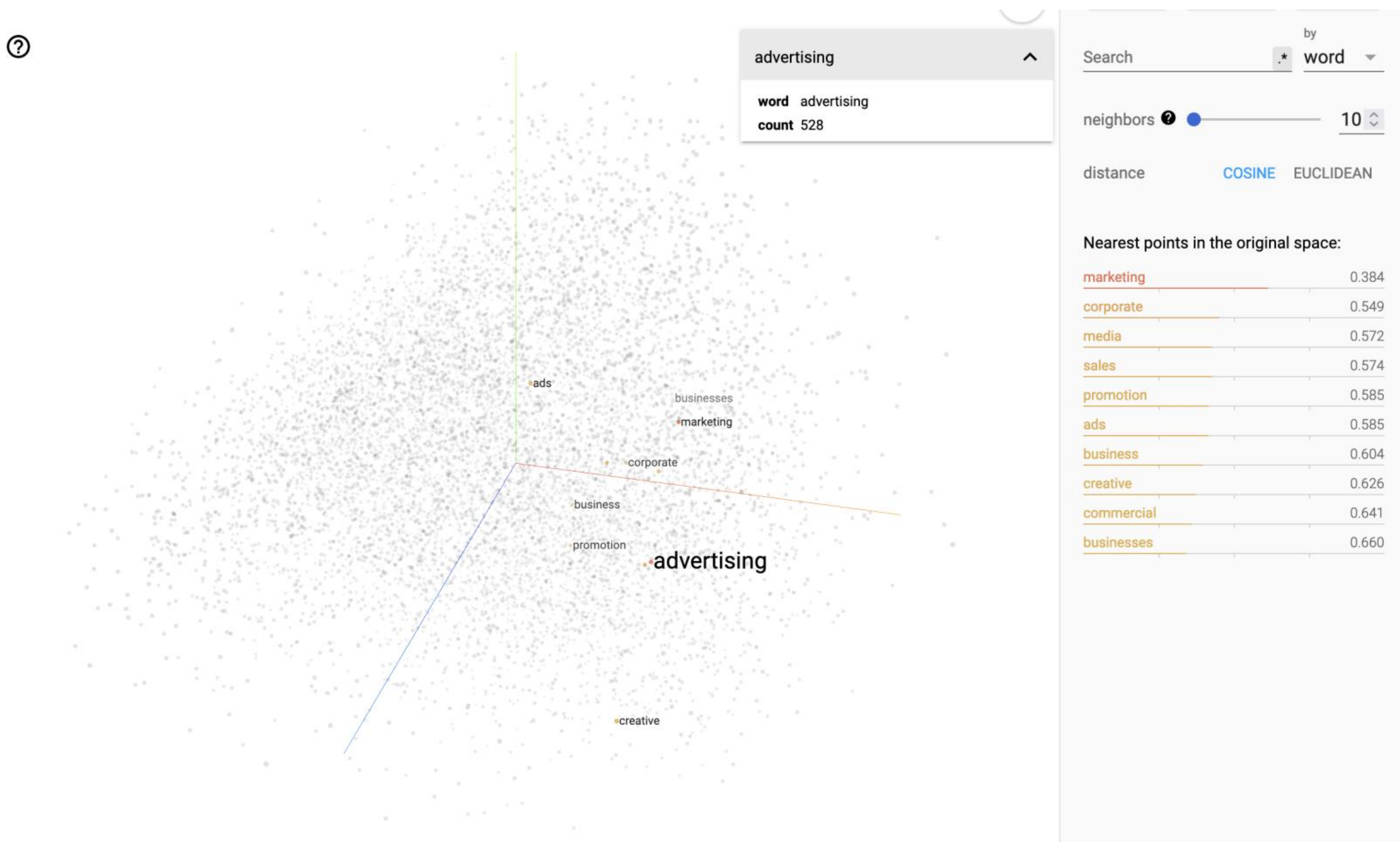
0.13

Results



https://projector.tensorflow.org/

Results


advertising
word advertising
count 528
Search
by
word
neighbors
10
distance
COSINE
EUCLIDEAN
Nearest points in the original space:
marketing 0.384
corporate 0.549
media 0.572
sales 0.574
promotion 0.585
ads 0.585
business 0.604
creative 0.626
commercial 0.641
businesses 0.660
ads
businesses
marketing
corporate
business
promotion
advertising
creative

Tokenization

- LLMs have a fixed vocabulary of words they recognize
- Many words are compound and can be constructed from smaller parts
- Some words may not be in the vocabulary, but we still want to deal with them
- Tokenization involves breaking up words into parts if necessary

Tokens	Characters
7	29

What's the capital of France?

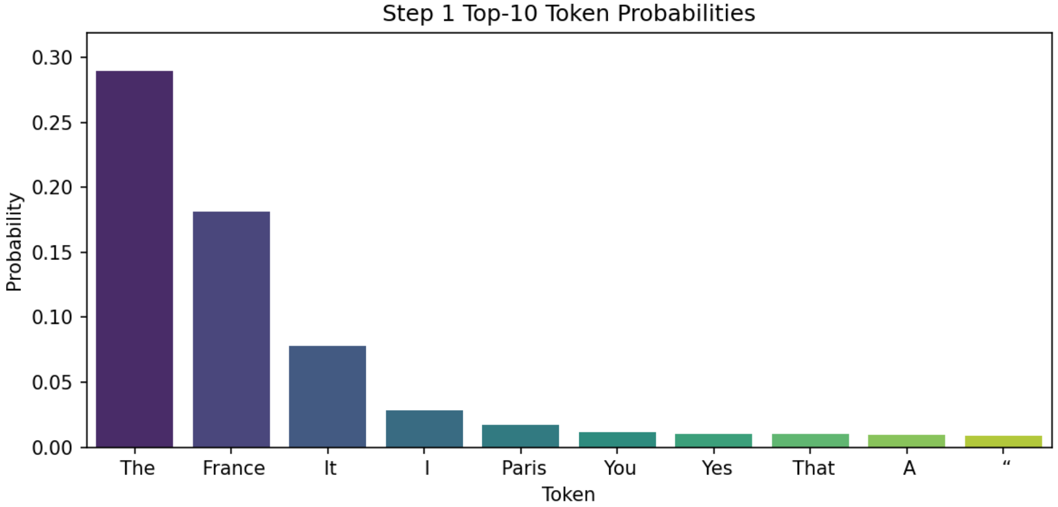
Tokens	Characters
5	28

Antidisestablishmentarianism

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

```
User: What's the capital of France?
```

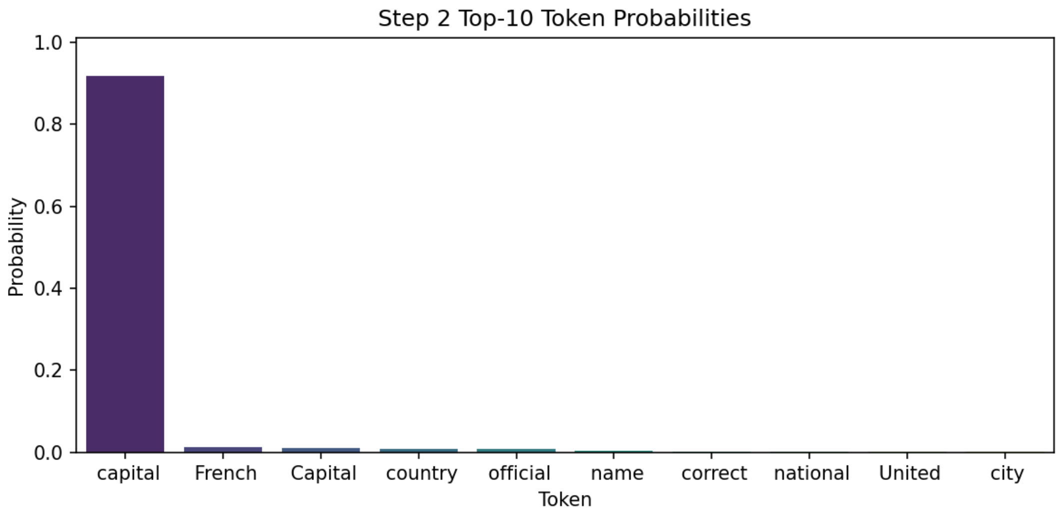



```
System: The
```

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

```
User: What’s the capital of France?
```

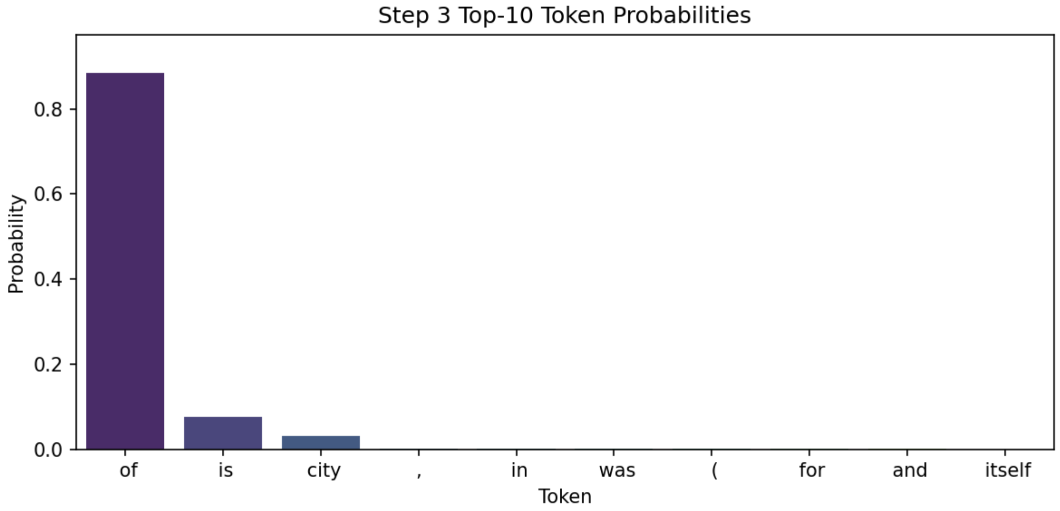



```
System: The capital
```

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

```
User: What's the capital of France?
```

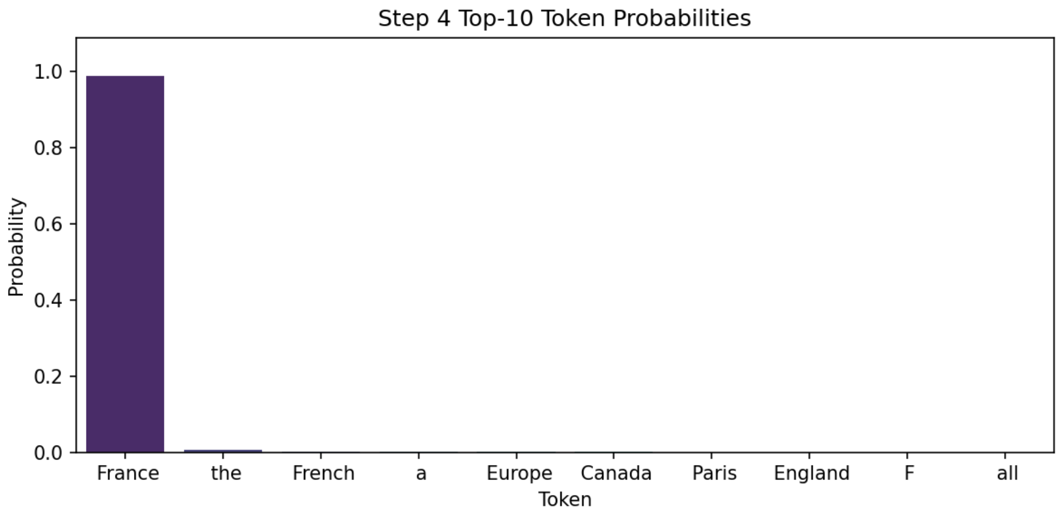



```
System: The capital of
```

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

```
User: What’s the capital of France?
```

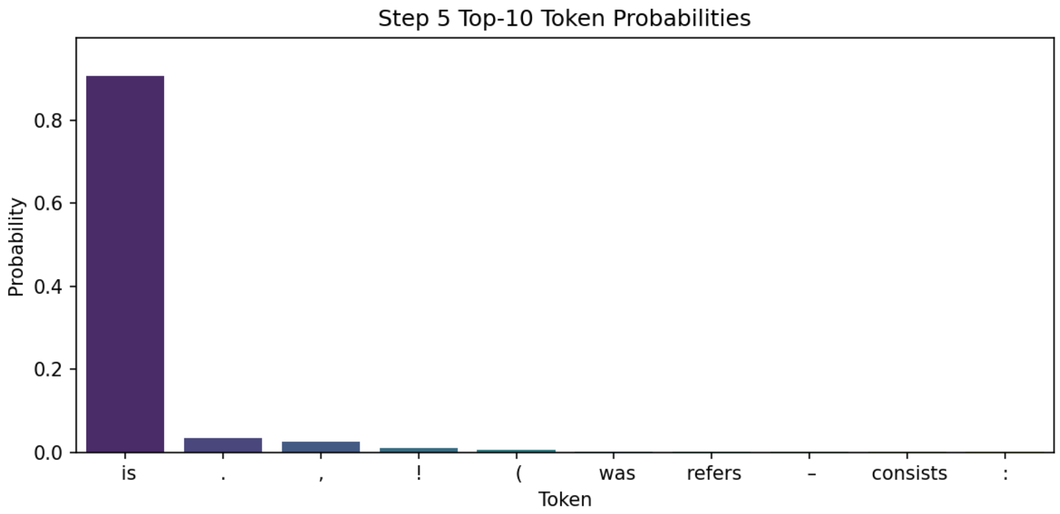



```
System: The capital of France
```

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

`User: What’s the capital of France?`

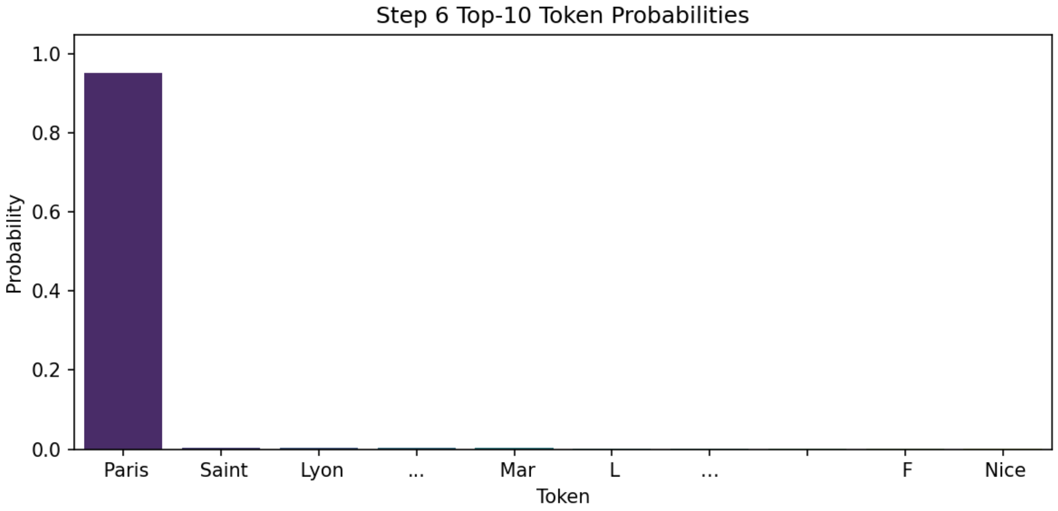


`System: The capital of France is`

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

`User: What’s the capital of France?`

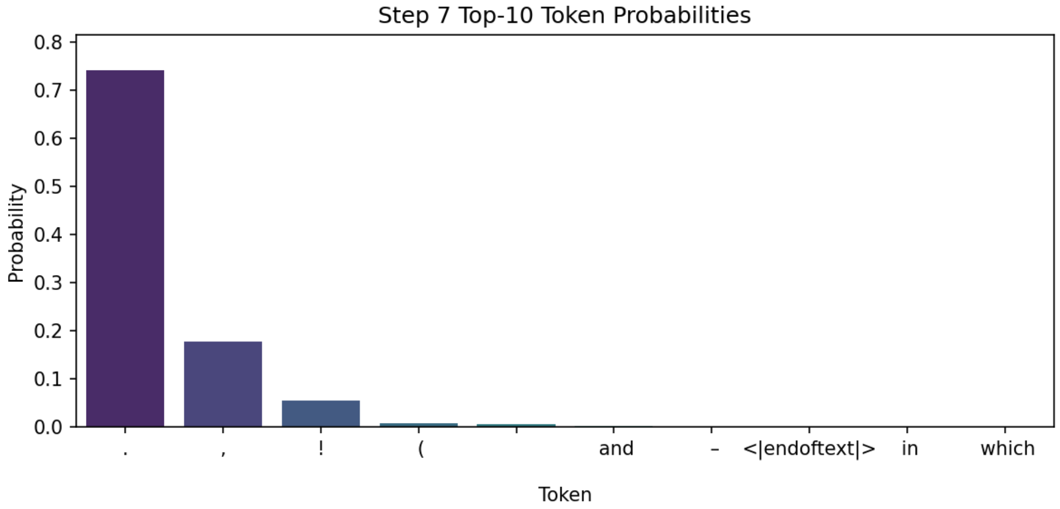


`System: The capital of France is Paris`

Text Generation

- LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
- The token to show to the user is semi-randomly selected, with weighting by estimated likelihood

```
User: What's the capital of France?
```




```
System: The capital of France is Paris.
```

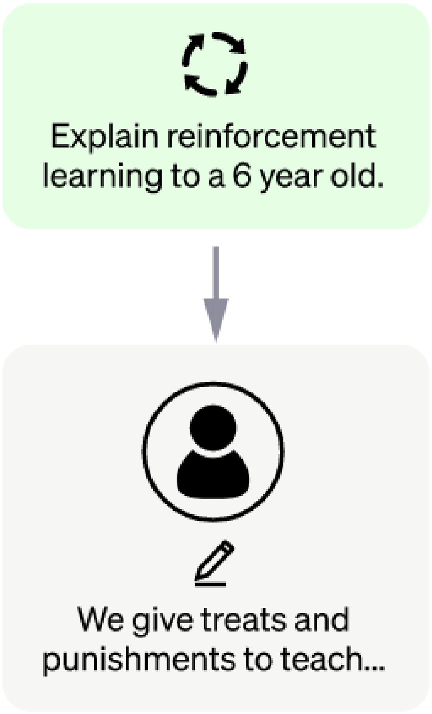
How do LLMs know what to generate?

- LLMs are trained on vast quantities of reference examples, called training data.
- These examples come from a wide variety of sources and allow the model to learn the fundamentals of language.
- During training, the model is provided with a nearly complete sentence and is asked to “fill in the blank”.

The training innovation of ChatGPT

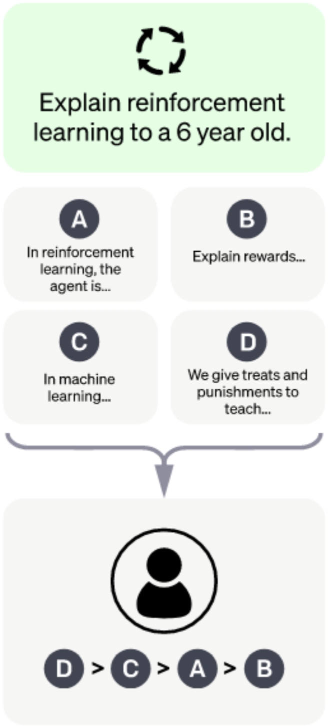
No more humans involved!

Human annotators write answers to questions



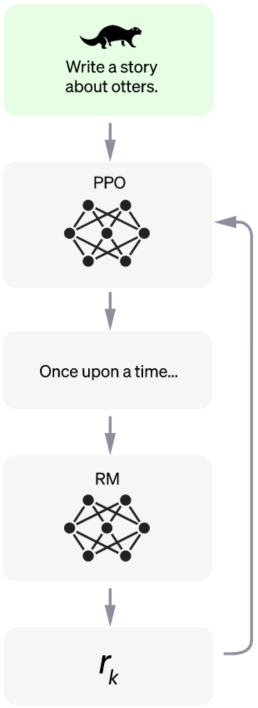
The generalist GPT-3 model is taught from these Q&A pairs

Human annotators write more answers, and someone else ranks them



A separate model learns to rate the quality of an answer

GPT writes answers to sampled questions



The reward model rates each answer, allowing GPT to keep learning

Figures: https://openai.com/blog/chatgpt/