Overview of LLMs and Transformers

Eric Wallace

Berkeley NLP

Berkeley AI Research
Lecture Overview

- State of the field
- Why the task of language modeling?
- Transformer architecture and tokenization
- Large-scale pre-training
- Instruction-tuning and RLHF
Today’s NLP Recipe
Today’s NLP Recipe

Curate massive pre-training data
Today’s NLP Recipe

Curate massive pre-training data

Create fine-tuning data
Today’s NLP Recipe

- Curate massive pre-training data
- Create fine-tuning data
- Train massive model
Today’s NLP Recipe

- Curate massive pre-training data
- Create fine-tuning data
- Train massive model
- Deploy model widely
Today’s NLP Recipe

- Curate massive pre-training data
- Create fine-tuning data
- Train massive model
- Deploy model widely
- Update using user interactions
Language Modeling

\[ p(x_1, x_2, \ldots, x_L) \]
Language Modeling

\[ p(x_1, x_2, \ldots, x_L) \]

\[ p(\text{the, mouse, ate, the, cheese}) = 0.001 \]

\[ p(\text{mouse, the, cheese, ate, the}) = 10^{-20} \]
Language Modeling

\[ p(x_1, x_2, \ldots, x_L) = \prod_{i=1}^{L} p(x_i \mid x_{1:i-1}) \]
Neural Language Models

\[
\prod_{i=1}^{L} p(x_i \mid x_{1:i-1})
\]
Neural Language Models

Prompt: The mouse ate the

\[
\prod_{i=1}^{L} p(x_i \mid x_{1:i-1})
\]
The mouse ate the

Neural network

Prompt
The mouse ate the

$$\prod_{i=1}^{L} p(x_i | x_{1:i-1})$$
The mouse ate the cheese

Neural Language Models

Prompt

The mouse ate the

Neural network

<table>
<thead>
<tr>
<th>Token</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheese</td>
<td>0.20</td>
</tr>
<tr>
<td>cookie</td>
<td>0.12</td>
</tr>
<tr>
<td>nibble</td>
<td>0.08</td>
</tr>
<tr>
<td>crumb</td>
<td>0.07</td>
</tr>
<tr>
<td>man</td>
<td>0.05</td>
</tr>
<tr>
<td>tail</td>
<td>0.04</td>
</tr>
</tbody>
</table>

\[
\prod_{i=1}^{L} p(x_i \mid x_{1:i-1})
\]
Why Language Modeling?

- Zero- and few-shot learning with language models
Why Language Modeling?

- Zero- and few-shot learning with language models

**Prompt**

Question: What is the sentiment of the sentence “Superb acting”?

Answer:
Why Language Modeling?

- Zero- and few-shot learning with language models

**Prompt**

Question: What is the sentiment of the sentence “Superb acting”?

**Answer:**

<table>
<thead>
<tr>
<th>Token</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0.82</td>
</tr>
<tr>
<td>negative</td>
<td>0.10</td>
</tr>
<tr>
<td>yes</td>
<td>0.006</td>
</tr>
<tr>
<td>hello</td>
<td>0.005</td>
</tr>
<tr>
<td>acting</td>
<td>0.003</td>
</tr>
<tr>
<td>amazing</td>
<td>0.001</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Why Language Modeling?

- “Oracle” LM can perform any task
Why Language Modeling?

- “Oracle” LM can perform any task
- Language modeling leads to rich representations
  - George Washington was born in the year _______
  - If it is raining, you may need an _____
  - Using the power rule, the derivative of $3x^5$ is _____
Why Language Modeling?

- “Oracle” LM can perform any task
- Language modeling leads to rich representations
  - George Washington was born in the year ______
  - If it is raining, you may need an _____
  - Using the power rule, the derivative of $3x^5$ is ____
Why Language Modeling?

● “Oracle” LM can perform any task

● Language modeling leads to rich representations
  ○ George Washington was born in the year ______
  ○ If it is raining, you may need an _____
  ○ Using the power rule, the derivative of 3x^5 is _____

● There is effectively “unlimited” data for language modeling
  ○ enables powerful function approximators (large transformers)
Neural LMs from Scratch
Tokenization

character-based models

faster

faste r

fast est

quicke st

vocabulary size

faster

fast er

fast est

quick est

word-based models

faster

fast er

fast est

quicke st
Data for Pre-training
Data for Pre-training

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sampling prop.</th>
<th>Epochs</th>
<th>Disk size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonCrawl</td>
<td>67.0%</td>
<td>1.10</td>
<td>3.3 TB</td>
</tr>
<tr>
<td>C4</td>
<td>15.0%</td>
<td>1.06</td>
<td>783 GB</td>
</tr>
<tr>
<td>Github</td>
<td>4.5%</td>
<td>0.64</td>
<td>328 GB</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4.5%</td>
<td>2.45</td>
<td>83 GB</td>
</tr>
<tr>
<td>Books</td>
<td>4.5%</td>
<td>2.23</td>
<td>85 GB</td>
</tr>
<tr>
<td>ArXiv</td>
<td>2.5%</td>
<td>1.06</td>
<td>92 GB</td>
</tr>
<tr>
<td>StackExchange</td>
<td>2.0%</td>
<td>1.03</td>
<td>78 GB</td>
</tr>
</tbody>
</table>

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.
Data for Pre-training
Data for Pre-training

Language Model
Data for Pre-training

Language Model

$p(x)$ $p(x)$ $p(x)$ $p(x)$ $p(x)$
Transformer Architecture
Transformer Architecture

ENCODER

Feed Forward Neural Network

Self-Attention
Transformer Architecture

Input

Embedding

Queries

Keys

Values

Thinking

Machines

$X_1$

$X_2$

$q_1$

$q_2$

$k_1$

$k_2$

$v_1$

$v_2$

$W^Q$

$W^K$

$W^V$
Transformer Architecture

Input
Embedding
Queries
Keys
Values
Score
Divide by $8 \cdot \sqrt{d_v}$
Softmax
Softmax
X
Value
Sum

Thinking

Machines

$x_1$  $x_2$
$q_1$  $q_2$
$k_1$  $k_2$
$v_1$  $v_2$

$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$

Score

14
12

Softmax

0.88
0.12

Value

$z_1$
$z_2$
Transformer Architecture
Transformer Architecture

**ATTENTION HEAD #0**

- $Q_0$
- $K_0$
- $V_0$

- $W_0^Q$
- $W_0^K$
- $W_0^V$

**ATTENTION HEAD #1**

- $Q_1$
- $K_1$
- $V_1$

- $W_1^Q$
- $W_1^K$
- $W_1^V$
Position Embeddings
Transformer Architecture
Feed-Forward

layer input

linear projection

non-linearity (often relu)

projected intermediate

linear projection

layer output
enjoys  playing  tennis  .  \textbf{<eos>}

She  enjoys  playing  tennis  .
Failures of Pre-trained LMs

PROMPT

*Explain the moon landing to a 6 year old in a few sentences.*
Failures of Pre-trained LMs

<table>
<thead>
<tr>
<th>PROMPT</th>
<th>Explain the moon landing to a 6 year old in a few sentences.</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETION</td>
<td>GPT-3</td>
</tr>
<tr>
<td></td>
<td>Explain the theory of gravity to a 6 year old.</td>
</tr>
<tr>
<td></td>
<td>Explain the theory of relativity to a 6 year old in a few sentences.</td>
</tr>
<tr>
<td></td>
<td>Explain the big bang theory to a 6 year old.</td>
</tr>
<tr>
<td></td>
<td>Explain evolution to a 6 year old.</td>
</tr>
</tbody>
</table>
## Failures of Pre-trained LMs

<table>
<thead>
<tr>
<th>PROMPT</th>
<th>Explain the moon landing to a 6 year old in a few sentences.</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETION</td>
<td>GPT-3</td>
</tr>
<tr>
<td>Explain the theory of gravity to a 6 year old.</td>
<td></td>
</tr>
<tr>
<td>Explain the theory of relativity to a 6 year old in a few sentences.</td>
<td></td>
</tr>
<tr>
<td>Explain the big bang theory to a 6 year old.</td>
<td></td>
</tr>
<tr>
<td>Explain evolution to a 6 year old.</td>
<td></td>
</tr>
</tbody>
</table>

Language models are not *aligned* with user intent [Ouyang et al., 2022].
## Failures of Pre-trained LMs

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Completion</th>
</tr>
</thead>
</table>
| Explain the moon landing to a 6 year old in a few sentences. | Explain the theory of gravity to a 6 year old.  
Explain the theory of relativity to a 6 year old in a few sentences.  
Explain the big bang theory to a 6 year old.  
Explain evolution to a 6 year old. |

Language models are not *aligned* with user intent [Ouyang et al., 2022].  
Finetuning to the rescue!
Failures of Pre-trained LMs

**Prompt**

*Explain the moon landing to a 6 year old in a few sentences.*

**Completion**

**Human**

A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

Language models are not *aligned* with user intent [Ouyang et al., 2022]. Finetuning to the rescue!
Instruction-tuning LLMs

• **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM
Instruction-tuning LLMs

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM

[FLAN-T5; Chung et al., 2022]
Instruction-tuning LLMs

- Collect examples of (instruction, output) pairs across many tasks and finetune an LM

- Evaluate on unseen tasks

[FLAN-T5; Chung et al., 2022]
Instruction tuning for language models (LMs) involves modifying the model's behavior through explicit instructions or directives provided in the form of prompts or conditioning phrases. This approach aims to guide the output generation of the LM towards desired outcomes, making it more controllable and aligned with user intentions. By adjusting the instructions, users can influence the style, content, or context of the generated text, enabling more specific and tailored responses from the model. Instruction tuning enhances the practicality and adaptability of LMs by allowing users to exert finer control over the generated outputs and align the model's responses with their preferences and requirements.

Can you do it again, but now with 3 paragraphs?

Instruction tuning for language models (LMs) involves the strategic manipulation of input prompts or conditioning phrases to steer the model's output towards desired outcomes. In the context of fine-tuning or adapting pre-trained LMs, instruction tuning provides a mechanism for users to exert greater control over the generated text, ensuring that
Limitations of Instruction-tuning

- **Problem 1**: it’s expensive to collect ground-truth data for tasks
  - Provide me five active research areas in April 2023 for LLMs

- **Problem 2**: tasks like open-ended creative generation have no right answer.
  - Write me a story about a dog and her pet grasshopper.

- Can we explicitly attempt to satisfy human preferences?
RL from Human Feedback
Write a Python function that checks if a given string is a palindrome.

Certainly! Here's a Python function that checks if a given string is a palindrome:

```python
def is_palindrome(word):
    # Convert the word to lowercase and remove spaces
    word = word.lower().replace(" ", "")
    # Compare the word with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```python
word1 = "refer"
print(is_palindrome(word1))  # Output: True

word2 = "Hello World"
print(is_palindrome(word2))  # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing `[::-1]`. If the word is equal to its reverse, it is a palindrome, and the function returns `True`. Otherwise, it returns `False`.
Write a Python function that checks if a given string is a palindrome.

Certainly! Here's a Python function that checks if a given string is a palindrome:

```
def is_palindrome(word):
    # Convert the word to lowercase and remove spaces
    word = word.lower().replace(' ', '')
    # Compare the word with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```
word1 = "world"
print(is_palindrome(word1))  # Output: False

word2 = "Hello World"
print(is_palindrome(word2))  # Output: True
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing `[::1]`. If the word is equal to its reverse, it is a palindrome, and the function returns `True`. Otherwise, it returns `False`.

This function takes a string as input, converts it to lowercase, and removes any spaces. It then checks if the reversed string is equal to the original string. If they are equal, the function returns `True`, indicating that the string is a palindrome. Otherwise, it returns `False`.

Here's an example of how you can use the function:

```
word = "madam"
if is_palindrome(word):
    print(f"""word" is a palindrome.")
else:
    print(f"""word" is not a palindrome."")
```

Output:

```
madam is a palindrome.
```

Note that this function considers only alphanumeric characters and ignores case sensitivity and spaces. If you want to consider other characters as well, you can modify the function accordingly.

In this function, we first remove all non-alphanumeric characters from the string and convert it to lowercase using a list comprehension. Then, we check if the reversed string is equal to the original string using slicing `[::1]` to create a reversed copy of the string. If they are equal, the string is a palindrome, and the function returns `True`; otherwise, it returns `False`.

Here's an example usage of the function:

```
string1 = "A man, a plan, a canal: Panama"
print(is_palindrome(string1))  # True

string2 = "race car"
print(is_palindrome(string2))  # True

string3 = "Hello World"
print(is_palindrome(string3))  # False
```

In the above example, `is_palindrome` is called with three different strings. The function correctly identifies the palindromes ("A man, a plan, a canal: Panama" and "race car") and returns `True`. For the non-palindrome "Hello World," it returns `False". 
Write a Python function that checks if a given string is a palindrome.

Certainly! Here's a Python function that checks if a given string is a palindrome:

```python
def is_palindrome(word):
    # Convert the string to lowercase and remove spaces
    word = word.lower().replace(" ", "")
    # Compare the string with its reverse
    if word == word[::-1]:
        return True
    else:
        return False
```

Here's how you can use the function:

```python
word1 = "radar"
palindrome1 = is_palindrome(word1) # Output: True

word2 = "Hello World"
palindrome2 = is_palindrome(word2) # Output: False
```

The function first converts the input word to lowercase and removes any spaces. Then, it compares the word with its reverse using slicing ("[::-1]"). If the word is equal to its reverse, it is a palindrome, and the function returns 'True'. Otherwise, it returns 'False'.

This function takes a string as input, converts it to lowercase, and removes any spaces. It then checks if the reversed string is equal to the original string. If they are equal, the function returns 'True', indicating that the string is a palindrome. Otherwise, it returns 'False'.

Here's an example of how you can use the function:

```python
word = "radar"
if is_palindrome(word):
    print("word is a palindrome.")
else:
    print("word is not a palindrome.")
```

Output:

```
radar is a palindrome.
```

Note that this function considers only alphanumeric characters and ignores case sensitivity and spaces. If you want to consider other characters as well, you can modify the function accordingly.
RL from Human Feedback

- **Step 1**: train reward model to predict human ratings

- **Step 2**: use RL to directly optimize against reward model
RL from Human Feedback

- **Step 1**: train reward model to predict human ratings
- **Step 2**: use RL to directly optimize against reward model

We reinforce good actions, increasing the chance they happen again.
RL from Human Feedback

- **Step 1:** train reward model to predict human ratings
- **Step 2:** use RL to directly optimize against reward model

We reinforce good actions, increasing the chance they happen again.

---

**GPT Assistant training pipeline**

<table>
<thead>
<tr>
<th>Stage</th>
<th>Pretraining</th>
<th>Supervised Finetuning</th>
<th>Reward Modeling</th>
<th>Reinforcement Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dataset</strong></td>
<td>Raw internet text trillions of words low-quality, large quantity</td>
<td>Demonstrations Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality</td>
<td>Comparisons 100K –1M comparisons written by contractors low quantity, high quality</td>
<td>Prompts ~10K-100K prompts written by contractors low quantity, high quality</td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td>Language modeling predict the next token</td>
<td>Language modeling predict the next token</td>
<td>Binary classification predict rewards consistent w preferences</td>
<td>Reinforcement Learning generate tokens that maximize the reward</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Base model</td>
<td>SFT model</td>
<td>RM model</td>
<td>RL model</td>
</tr>
<tr>
<td><strong>Notes</strong></td>
<td>1000s of GPUs months of training ex: GPT, LLaMA, PaLM can deploy this model</td>
<td>1-100 GPUs days of training ex: Vicuna-13B can deploy this model</td>
<td>1-100 GPUs days of training</td>
<td>1-100 GPUs days of training ex: ChatGPT, Claude can deploy this model</td>
</tr>
</tbody>
</table>

---

**Reinforcement Learning**

- **Prompts:** ~10K-100K prompts written by contractors low quantity, high quality