Language models for statisticians: from $n$-grams to transformers to chatbots

Bob Carpenter
Center for Computational Mathematics
Flatiron Institute

July 2023
What is a language model?

- **Language** uses a **finite** number of symbols called **tokens**
  - we assume a finite **token set** Tok of size $K$

- Tokens may be letters, words, sounds, syllables, etc.
  - GPT uses **sequences of letters** (average 1.5 tokens per English word)

- Treat language as a **stochastic process**
  - $Y = Y_1, Y_2, \ldots$ for random variables $Y_n \in \text{Tok}$

- Models typically **autoregressive**, predicting next word from previous
**N-gram language models**

(Shannon 1948)

- Assume language process is **order-\(N\) Markov**
  - tokens conditionally independent given previous \(N - 1\) tokens

\[
p(y_k \mid y_{k-1}, \ldots, y_1) = p(y_k \mid y_{k-1}, \ldots, y_{k-N+1}).
\]

- Even **GPT is Markovian**
  - GPT-3: \(N = 4096\)  
  - GPT-4: \(N = 8192\)  
  - Claude: \(N = 100,000\)
  - **bottleneck** is \(O(N^2)\) attention algorithm (Claude more clever?)
  - cf. a real computer is technically a finite-state machine
Shannon’s $N$-gram models


- Shannon used English *letters* ($K = 1, 2, 3$) and *words* ($K = 1, 2$)

- **What is English?** How do we collect a *sample*?

- Shannon used *books of frequencies*
  - *letter trigrams* (1939 book); *word bigrams* (1923 book)

- Fit and inference usually *regularized MLE* for efficiency
  - ensures *non-zero probability* for any sequence
Shannon’s fit

- MLE probabilities from compiled tables of letters (1923), words (1939)
  - or, open books at random, find current context, generate following word

- Shannon generated random examples
  - **Order 1, letters**: OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.
  - **Order 3, letters**: IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA
  - **Order 1, words**: REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO
  - **Order 2, words**: THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE
Measuring accuracy with entropy

• Accuracy of $N$-gram language model $p_Y$ measured with **entropy (rate)**

• Given a random sequence $Y \in \text{Tok}^K$, its **entropy** in **bits** (base 2) is

$$H[Y] = \mathbb{E} [ \log_2 p_Y(Y)] = \sum_{y \in \text{Tok}^K} p_Y(y) \cdot \log_2 p_Y(y).$$

• The **entropy rate** is average entropy per token, $\lim_{K \to \infty} H[Y]/K$,

• The entropy rate for $N$-grams is given by **conditional entropy**,

$$H[Y_K \mid Y_{K-1}, \ldots, Y_{K-N-1}] = \mathbb{E} [ \log_2 p(Y_K \mid Y_{K-1}, \ldots, Y_{K-N-1}) ]$$
Signal processing: entropy and compression

- Shannon (1948) introduced information theory to model signal compression and decompression for communication.

- Assume a language model with pmf $p_Y$.

- **Compress** $y \in \text{Tok}^*$, to $\lceil \log_2 p_Y(y) \rceil$ bits
  - in practice with arithmetic coding (Witten, Neal, Cleary 1987)
OpenAI’s GPT-3: Published

- **Training set sizes**

<table>
<thead>
<tr>
<th>Source</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Crawl</td>
<td>410 billion</td>
</tr>
<tr>
<td>Books2</td>
<td>55 billion</td>
</tr>
<tr>
<td>WebText2</td>
<td>19 billion</td>
</tr>
<tr>
<td>Books1</td>
<td>12 billion</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>3 billion</td>
</tr>
<tr>
<td></td>
<td>≈ 500 billion</td>
</tr>
</tbody>
</table>

- **Number of parameters**: ≈175 billion

- **Context history size**: 4K tokens

- Let’s turn to **how it works** …
Top-level architecture

- Transformer, but decoder only
- \( \text{tok}_n \): \( n \)-th input token
- \( x^k_n \): value of token \( n \) at layer \( k \)
- \( \text{prob}_t \): probability next token is \( t \)
- circles enclose model parameters
Attention architecture

- attention then feedforward neural net

- **ResNet architecture**: tees to add input
  - ≈ hierarchical model of differences
  - non-centered parameterization

- standard two-layer **neural nets**
  - shared params for each value

- standardized for numerics
Pseudocode: GPT in 40 lines

SIZES

----------------------------------------------------
T: number of distinct tokens
N: size of context (history)

V: size of token embedding vectors
A: number of attention layers
K: size of keys and queries
L: width of neural network
DECODE(tok:: int<low=1,up=T>[N], alpha:: matrix(T, V),
    betas:: { query::matrix(K, V),
                key::matrix(K, V),
                value:: matrix(V, V ) }[A]
    gammas:: nn(V, L)[A],
    delta:: {1: vector[T],
              2: matrix(T, N * V)}): simplex[T]

for n in 1:N:
    xs[0, n] = LEX(tok[n], alpha) + POS(n)
for a in 1:A:
    xs[a] = ATTEND(xs[a - 1], betas[a], gammas[a])
    for n in 1:N:
        xs[a, n] = FEED_FORWARD(xs[a, n], gammas[a])
    y = STANDARDIZE(delta.1 + delta.2 * xs[A].flatten())
return SOFTMAX(y)
LEX(t: \text{int}<\text{low}=1,\text{up}=T>,
   \text{alpha}: \text{vector}(V)[T]): \text{vector}(V)
----------------------------------------------------------
return \text{alpha}[t]

POS(n: \text{int}<\text{low}=1,\text{up}=N>): \text{vector}(V)
----------------------------------------------------------
for i in 1:V / 2:
   r = n / N^*(2 \times i / V) \quad // \text{pos} / \text{max_pos}^[0..2]
   u[2 \times i] = \sin(r)
   u[2 \times i + 1] = \cos(r)
return u
ATTEND(x: vector(V)[N],
    beta: { query: matrix(K, V), key: matrix(K, V),
            value: matrix(V, V)},
    gamma: nn(V, L)): vector(V)[N]
-----------------------------------------------------------
for n in 1:N:
    q[n] = beta.query * x[n]
    k[n] = beta.key * x[n]
    v[n] = beta.value * x[n]
for n in 1:N:
    lp[1:n-1] = [q[n]’ * k[1], ..., q[n]’ * k[n-1]] / sqrt(V)
    lp[n:N] = -inf
    p = SOFTMAX(lp)
    u[n] = SUM(n in 1:N) p[n] * v[n]
    y[n] = STANDARDIZE(u[n] + x[n])
return y
FEED_FORWARD(x: real[R],
    alpha: { 1: real[S], 2: real[S, R],
            3: real[R], 4: real[R, S]): real[R]
----------------------------------------------------------
  u = alpha.1 + alpha.2 * x
  v = GELU(u)
  y = alpha.3 + alpha.4 * v
return STANDARDIZE(x + y)

GELU(v: real[R]): real[R]
  return [v_i * Phi(v_i) for v_i in v]

STANDARDIZE(v: real[R]): real[R]
  return (v - mean(v)) / std_dev(v)

SOFTMAX(real[R] v): simplex(R)
  return exp(v) / sum(exp(v))
Multi-head attention

- What we have presented is **single-head attention**

- In practice, GPT uses **multi-head attention**
  - $J$ parallel attention “heads”
  - keys, values, queries for each head **projected from previous layer value**
  - **value projected** for each head down to size to $V/J$
  - **concatenate** output of each head to produce size $V$ value

- GPT-4 rumored to use parallel GPTs in an ensemble
GPT-3 sizes

• 175 billion parameters
• 96 layers
• 12,288 total value width
• 96 parallel attention heads
  – 128 value width per head
From LLM to Chatbot

• **LLM goal**: predict *next token on web* page

• **Chatbot goal** is to train a model that is
  – **helpful**: help users solve task
  – **honest**: shouldn’t fabricate or mislead user
  – **harmless**: shouldn’t cause physical, psychological, social, or environmental harm

• Strategy is to **align** an LLM to be a Chatbot with **fine tuning**
  – LLM acts as an **informative prior**
  – In ML terms, LLM provides **inductive bias**
Reinforcement learning with human feedback (RLHF)

1. Supervised fine tuning
   - human raters **provide desired output** for sampled prompts
   - **fine-tune** LLM with **supervised learning**

2. Reward model training
   - human raters **rank multiple outputs** for sample prompts
   - train a **reward model**

3. Reinforcement learning
   - **policy ranks outputs** for sample prompts
   - fine-tune LLM with **proximal policy optimization** (PPO)
Some caveats (OpenAI 2022)

- “This procedure aligns the behavior of GPT-3 to the stated preferences of a specific group of people (mostly our labelers and researchers), rather than to any broader notion of “human values”.
  - cf. Cultural consensus theory provides mixture model of “values”

- “During RLHF fine-turning, we observe performance regressions compared to GPT-3 on certain public NLP datasets.
  - i.e., performance degrades relative to unaligned model
  - partially mitigated by hierarchical modeling alternating reinforcement and supervision
• Accuracy is \textbf{bounded by parameter size} (right)

• Accuracy is \textbf{bounded by data size} (left)

Larger models require \textbf{fewer samples} to reach the same performance.

![Graph showing loss vs. tokens and model size](image)

- **Loss vs. tokens, model size** (OpenAI)

- Accuracies are bounded by parameter size and data size.

- Larger models require fewer samples to reach the same performance.

- The optimal model size grows smoothly with the computational budget.

- Increasing data and parallelism is increasingly sample efficient.

- Researchers typically train smaller models for longer than would be expected as we scale up.

- Computational requirements for training increase parallelism through larger batch sizes.

- A relatively small increase in data is needed to avoid reuse.

- For models with a limited number of parameters, trained to convergence on sufficiently large datasets, an optimally-sized model, and a sufficiently small batch size (making optimal use of the compute budget) should be used to make predictions.
Scaling models (DeepMind)

- Accuracy **determined by flops**
  - for given flops, there is an **optimal choice** of training tokens and model size
  - fits held out predictions **very well**

- (l) loss by model size, (c) optimal parameters, (r) optimal train tokens
OpenAI’s GPT-4: Unpublished

- **Training set** unpublished (estimated $\approx 5$ trillion)
- **Parameter set** unpublished (estimated $\approx 2$ trillion)
- **Context history size**: 8K or 32K tokens
- **Cluster cost** training: $\approx$US$500$ million (incl. 10K+ US$15K$ GPUs)
- **Marginal cost** training: $\approx$US$10$s of millions (hardware, power, staff)

- **Open AI** is now **ClosedAI**: “Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.”
The cat’s out of the bag

- Transformer LLM architecture published by Google (2017)
- Alignment to ChatBots published by OpenAI (2022)
  - Meta (nee Facebook): LLaMA
    * Open source for research (since leaked)
    * Stanford CS: Alpaca fine-tuned
    * Runs 2 tokens/second on iMac with 4-bit floating point
  - Google: Bard
  - Google and OpenAI: Copilot (code/programming API integration)
  - Anthropic: Claude (100K token context) (branded as Poe for writing)
  - Many smaller, less widely used alternatives
LLM References

1. *The transformer paper*:
   Vaswani et al. (Google). 2017. *(82K citations)*
   **Attention is all you need.** *NeurIPS.*

2. *LLMs are highly generalizable*:
   Brown et al. (OpenAI). 2020. *(12K citations)*
   **Language models are few-shot learners.** *NeurIPS.*

3. *Going from GPT to ChatGPT*:
   Ouyang et al. (OpenAI). 2022. *(1.5K citations)*
   **Training language models to follow instructions.** *NeurIPS.*
4. *Original OpenAI paper on scaling*: 

Kaplan et al. 2020.  
**Scaling laws for neural language models.** *arXiv*  
(0.6K citations)

5. *Chinchilla paper on scaling laws for transformers*: 

Hoffmann et al. (DeepMind) 2022.  
**Training compute-optimal large language models.** *arXiv*  
(0.1K citations)

6. *What can GPT-4 do?* 

Bubeck et al. (Microsoft). 2023.  
**Sparks of artificial general intelligence.** *arXiv.*  
(0.4K citations)
7. Another pseudocode I found after I did mine:

Phuong & Hutter (DeepMind). 2022. (0.02K citations)
Formal algorithms for transformers. arXiv.

8. Reproducible PyTorch case study with Colab notebook fitting Shakespeare:

Andrej Karpathy (now at OpenAI). 2023. (2.8M views)
Let’s build GPT: from scratch, in code, spelled out. YouTube!