A brief introduction to (large) language models

Sachin Kumar

sachink@allenai.org
What are we going to talk about?

- The language modeling problem
- How do we learn a language model?
  - A quick primer on learning a model via gradient descent
  - The role of training data
- ✨✨ The Transformer ✨✨
  - The two things that make it such an improvement over our previous techniques for language modeling
  - More detail about both of those two things
- From language models to large language models
- Large language models for chat (ala ChatGPT)
Quick poll

1. Are you familiar with supervised machine learning? gradient descent?

2. Are you familiar with neural networks?
The language modeling problem

Rank these sentences in the order of plausibility?

1. Jane went to the store.
2. store to Jane went the.
3. Jane went store.
4. Jane goed to the store.
5. The store went to Jane.
6. The food truck went to Jane.

How probable is a piece of text? Or what is $p(\text{text})$
\[
p(how \text{ are you this evening? has your house ever been burgled?}) = 10^{-15}
\]
\[
p(how \text{ are you this evening? fine, thanks, how about you?}) = 10^{-9}
\]
The language modeling problem

A language model answers the question: **What is p(text)?**

Text is a sequence of symbols: \((x^{(1)}, x^{(2)}, \ldots, x^{(N)})\)

\[
p(x^{(1)}, x^{(2)}, \ldots, x^{(N)}) = p(x^{(1)})p(x^{(2)}|x^{(1)})p(x^{(3)}|x^{(1)}, x^{(2)}) \ldots
\]

\[
\prod_{i=1}^{N} p(x^{(i)}|x^{(1)}, \ldots, x^{(i-1)})^{\text{context}}
\]

Just the chain rule of probability—no simplifying assumptions!
The language modeling problem

\[ \prod_{i=1}^{N} p(x^{(i)} | x^{(1)}, \ldots, x^{(i-1)}) \]

context

the students opened their ______

[the rest of the vocabulary]
Language models of this form can generate text

At each timestep, sample a token from the language model’s new probability distribution over next tokens.

The ____
The students ____
The students opened ____
The students opened their ____

[the rest of the LM's vocabulary]
In short, predicting which word comes next
Language models play the role of ...

- a judge of grammaticality
  - e.g., should prefer “The boy runs.” to “The boy run.”
- a judge of semantic plausibility
  - e.g., should prefer “The woman spoke.” to “The sandwich spoke.”
- an enforcer of stylistic consistency
  - e.g., should prefer “Hello, how are you this evening? Fine, thanks, how are you?” to “Hello, how are you this evening? Has your house ever been burgled?”
- a repository of knowledge (?)
  - e.g., “Barack Obama was the 44th President of the United States”

Note that this is very difficult to guarantee!
Language models in the news (these days, ChatGPT)

Hi, I'm writing an article about you and all of the cool things you can do. Could you say hello to my readers and give us a quick self-introduction? I'd like it to be friendly and casual, include a silly joke, and end with a note about how great the rest of my article is.

Hello readers! My name is Assistant, and I'm a large language model trained by OpenAI. I'm here to help answer your questions and provide you with information on a wide variety of topics. I'm a bit of a jokester, so here's a quick one for you: Why was the math book sad? Because it had too many problems! But seriously, I'm excited to be featured in your article and I hope you enjoy learning more about me and all the cool things I can do. And trust me, the rest of the article is going to be great!
We use language models every day
We use language models every day
Why language modeling?

- Machine translation
  - $p(\text{strong winds}) > p(\text{large winds})$

- Spelling correction
  - The office is about fifteen minuets from my house
  - $p(\text{about fifteen minutes from}) > p(\text{about fifteen minuets from})$

- Speech recognition
  - $p(\text{I saw a van}) >>> p(\text{eyes awe of an})$

- Summarization, question-answering, handwriting recognition, OCR, etc.
How we learn a language model
Language modeling

a very large corpus → language model →
- the best recipe for bi
- the best recipe for biscuits
- the best recipe for biscuits and gravy
- the best recipe for birthday cake
- the best recipe for biscotti
How do we learn a language model?

Estimate probabilities using text data

- Collect a textual corpus
- Find a distribution that maximizes the probability of the corpus – maximum likelihood estimation

A naive solution: count and divide

- Assume we have $N$ training sentences
- Let $x_1, x_2, \ldots, x_n$ be a sentence, and $c(x_1, x_2, \ldots, x_n)$ be the number of times it appeared in the training data.
- Define a language model:

$$p(x_1, \ldots, x_n) = \frac{c(x_1, \ldots, x_n)}{N}$$

No generalization!
Markov assumption

- We make the Markov assumption: $x^{(t+1)}$ depends only on the preceding $n-1$ words.

$$P(x^{(t+1)} | x^{(t)}, \ldots, x^{(1)}) = P(x^{(t+1)} | x^{(t)}, \ldots, x^{(t-n+2)})$$
Markov assumption

\[ P(\text{the} \mid \text{its water is so transparent that}) \equiv P(\text{the} \mid \text{transparent that}) \]

or maybe even

\[ P(\text{the} \mid \text{its water is so transparent that}) \equiv P(\text{the} \mid \text{that}) \]
n-gram Language Models

$$\prod_{i=1}^{N} p(x^{(i)} | x^{(1)}, \ldots, x^{(i-1)})$$

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

- **Question**: How to learn a Language Model?
- **Answer (pre- Deep Learning)**: learn an *n-gram* Language Model!

- **Idea**: Collect statistics about how frequent different n-grams are and use these to predict next word
unigram probability

“\textit{I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.}”

- corpus size $m = 17$
- $P(\text{Lucy}) = 2/17$; $P(\text{cats}) = 1/17$

- Unigram probability: $P(w) = \frac{\text{count}(w)}{m} = \frac{C(w)}{m}$
"I have a dog whose name is Lucy. I have two cats, they like playing with Lucy."

\[ P(A \mid B) = \frac{P(A,B)}{P(B)} \]

\[ P(\text{have} \mid \text{I}) = \frac{P(\text{I have})}{P(\text{I})} = \frac{2}{2} = 1 \]

\[ P(\text{two} \mid \text{have}) = \frac{P(\text{have two})}{P(\text{have})} = \frac{1}{2} = 0.5 \]

\[ P(\text{eating} \mid \text{have}) = \frac{P(\text{have eating})}{P(\text{have})} = \frac{0}{2} = 0 \]

\[ P(w_2 \mid w_1) = \frac{C(w_1,w_2)}{\sum_w C(w_1,w)} = \frac{C(w_1,w_2)}{C(w_1)} \]
I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.

\[
P(A \mid B) = \frac{P(A,B)}{P(B)}
\]

\[
P(a \mid I \text{ have}) = \frac{C(I \text{ have a})}{C(I \text{ have})} = \frac{1}{2} = 0.5
\]

\[
P(\text{several} \mid I \text{ have}) = \frac{C(I \text{ have several})}{C(I \text{ have})} = \frac{0}{2} = 0
\]

\[
P(w_3 \mid w_1 w_2) = \frac{C(w_1, w_2, w_3)}{\sum_w C(w_1, w_2, w)} = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)}
\]
n-gram probability

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

\[
P(A \mid B) = \frac{P(A, B)}{P(B)}
\]

\[
P(w_i \mid w_1, w_2, \ldots, w_{i-1}) = \frac{C(w_1, w_2, \ldots, w_{i-1}, w_i)}{C(w_1, w_2, \ldots, w_{i-1})}
\]
Sampling from an n-gram language model

1-gram: Months the my and issue of year foreign new exchange’s September were recession exchange new endorsed a acquire to six executives
Sampling from a language model

Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
Sampling from a language model

1 gram

Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives

2 gram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions
Sampling from a language model

1 gram

-To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

2 gram

-Hill he late speaks; or! a more to leg less first you enter

1 gram

-Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

2 gram

-What means, sir. I confess she? then all sorts, he is trim, captain.

3 gram

-Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.

2 gram

-This shall forbid it should be branded, if renown made it empty.

3 gram

-King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;

4 gram

-It cannot be but so.
I have a dog whose name is Lucy. I have two cat. A differentiable function (e.g. a neural network)\[ \prod_{i=1}^{N} p(x^{(i)} | x^{(1)}, \ldots, x^{(i-1)}) \]

A distribution over the vocabulary

A differentiable function (e.g. a neural network)
How do we maximize the likelihood?

The dominant strategy from the past decade:

1. The randomly initialized differentiable function (neural network) takes the context as input
2. Have that function output a probability distribution over the vocabulary
3. Treat the probability of the correct token as your objective to maximize.
4. Or negative log (probability) as your objective to minimize
5. Differentiate with respect to the parameters, and perform gradient descent, or
Intuition of gradient descent

How do I get to the bottom of this river canyon?

Look around me 360°

Find the direction of steepest slope up

Go the opposite direction
Gradient descent: a throwback to calculus

Q: Given current parameter $w$, should we make $w$ bigger or smaller to minimize our loss?
A: Move $w$ in the reverse direction from the slope of the function.
Let's first visualize for a single scalar $w$

Q: Given current $w$, should we make it bigger or smaller?
A: Move $w$ in the reverse direction from the slope of the function

loss

slope of loss at $w^1$ is negative
So we'll move positive

$w^1$, $w^\text{min}$, $0$, (goal)
Let's first visualize for a single scalar $w$

Q: Given current $w$, should we make it bigger or smaller?
A: Move $w$ in the reverse direction from the slope of the function

![Diagram showing the loss function and the movement of $w$](image-url)
Now let's imagine 2 dimensions, w and b

Visualizing the (negative) gradient vector at the red point

It has two dimensions shown in the x-y plane
Key difference from our motivating scenario: in practice, calculating the exact gradient is really time-consuming.

So… we estimate the gradient using samples of data.
The Transformer
Why did the transformer make such a big difference for language modeling?

It allowed for faster learning of more model parameters on more data (by allowing parallel computation on GPUs!)
A brief aside about some visual shorthand I’ll be using
A 3-layer LSTM’s calculations for an input of 10 tokens

(For more on computing gradients via backpropagation, see colah’s blog post on this topic)
One layer of the transformer architecture (Vaswani et al. 2017)
One layer of the transformer architecture (Vaswani et al. 2017)

ok but how does this mess help anything
Comparing training times: how many functions do we need to backpropagate through?
Comparing training times: how many functions do we need to backpropagate through?

**Transformers parallelize a lot of the computations that LSTMs make us do in sequence**
Comparing training times: how many functions do we need to backpropagate through?

**Transformers parallelize a lot of the computations that LSTMs make us do in sequence**

And (a very specific, but nonempty, subset of) you can therefore train a transformer on a ridiculously large amount of data in a way that you cannot for an LSTM.
What kind of function can take in a variable number of inputs like that?
Attention mechanisms
Building up to the attention mechanism

What about an average?

But we probably don’t want to weight all input vectors equally…

How about a weighted average?

Great idea! **How can we automatically decide the weights for a weighted average of the input vectors?**
A simple form of attention (adapted from Bahdanau et al. 2014)

Computed how?
1. Dot product between param vector and each input vector
2. Softmax the set of resulting scalars.
Pros and cons

Pros:

● We have a function that can compute a weighted average (largely) in parallel of an arbitrary number of vectors!
● The parameters determining what makes it into our output representation are learned

Cons:

● We’re also hoping to produce $n$ different output token representations… and this just produces one…
Enter “self attention”

“What if instead of comparing each vector of the sequence to a single learned vector, we compared the sequence to itself?”
Queries, Keys, Values

- Queries: Input to function, one attention head (likely move through columns), learned Q params, dimension of each attention head, number of attention heads.
- Keys: Input to function, individual token, vectors, learned K params, dimension of each attention head, number of attention heads.
- Values: Input to function, individual token, vectors, learned V params, dimension of each attention head, number of attention heads.

Technically could be any constant divisible by the number of attention heads, but HuggingFace always sets it to the dimension of the input vectors, for good reason.

Add learned bias vector to each row, then reshape each row.
big raw matrices, one for each attention head (huggingface calls these "attention scores")

divide all elements by \( \sqrt{\text{dim of each attm head}} \),

then softmax over each row vector (separate softmax for each (row, attm head) pair)

new big raw matrices

concatenate all attm heads' contributions

new final representation

individual
tokenvectorvectors
Hooray for self attention!

Our function is still made up almost entirely of matrix multiplications! Which are very parallelizable (→ efficient!)

We still learn fixed-size blocks of parameters that can be used for a sequence with an arbitrary length

We’re now capable of producing $n$ different new token representations!
Self attention is the key component of the transformer
That’s all I’ve got! Questions?
A brief aside: let’s talk about data

What does each instance of data is contribute?

Some of the nudges to a model’s parameters over the course of training.

Which data is used to train modern large language models?

Web text

… it’s kind of tough to give a more specific description than that.

See Dodge et al. EMNLP ‘21, “Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus”

Also see Gururangan et al. EMNLP ‘22, “Whose Language Counts as High Quality? Measuring Language Ideologies in Text Data Selection”
Large Language Models

The transformer model allows fast parallel computations on many GPUs (**large amounts of compute**)  

It allows training on **large amounts of data** (think the whole internet worth of text).  

It allows adding many and many layers in the model (**large model**)  

A large language model is a language model with a large number of parameters, trained on large amounts of data, for long period of time.
Why large language models?

- Scaling the models, compute, and data leads in increase in performance

- Emergent properties at scale (Wei et al 2022)
  - Large models (with 7-100B+ parameters) suddenly become capable of performing tasks they weren’t able to do when small (such as 1B or small).
Training the model to chat

A simple language model (also called a pretrained model) is not equipped to chat with an end user, like ChatGPT.

ChatGPT (and many other models) are further trained on supervised data to follow instructions.
Instruction Tuning

- Collect a large dataset of instruction following examples of the form
  - <instruction> <input> <output>
  - For example,
  - Summarize this news article [ARTICLE] [SUMMARY]
  - Answer this question [QUESTION] [ANSWER]
  - Predict the sentiment of this review [REVIEW] [SENTIMENT]...

- This is also a text corpus but in a very specific format.

- Continue training the model on this dataset (again using the same training objective)
Aligning the model to humans’ preferences

● Chat based on models are supposed to converse with humans

● Why not learn from humans’ feedback

● Basic idea: Model samples multiple outputs – users rank them based on their preference
  ○ Convert user preferences into reward scores – more preferred output has higher reward
  ○ Treat an LLM like an agent and use RL to maximize this reward (RLHF)
So what does this mean ChatGPT is good at?
Some aspects of producing answers that might fall under that category:

- Writing in specific styles (that have appeared in the model’s training data)
- Grammatical consistency
- Generating boilerplate sentences that often appear at the beginning, end of emails, etc.
- Fluency
What are some problems that ChatGPT’s training leaves it prone to?
Inaccuracies

- The language model doesn’t “plan” what it will say in advance

- The model doesn’t store facts, just outputs plausible looking sentences which may or may not be factual
Lack of source attribution

Just like the model doesn’t store facts... it doesn’t store sources.

Sure, here are the links to the papers I mentioned:


I hope you find them helpful!
Outputs that reflect social biases

An example from machine translation a few years ago:
Thanks! Questions?