Large Language Models

Introduction to Language Models

Flavio Giobergia

What is a Language Model?

- A LM is a *probabilistic model* of a *natural language*

- i.e., it produces a probability $P(w_1, w_2, \dots, w_{T-1}, w_T)$ for any sequence of words $w_1, w_2, \dots, w_{T-1}, w_T$
  - $P(this, is, a, reasonable, sentence) = 0.1$
  - $P(this, is, a, purple, sentence) = 0.01$
  - $P(this, are, a, reasonable, sentence) = 0.001$

- The probability over all possible sentences in a language sums to 1
- Some regions of the “natural language space” have higher probabilities
  - Plausible sentences
- others have lower probabilities
  - inplausible, grammatically incorrect, incoherent sequences of words

Formal definition for language models

$$P(w_1, w_2, \dots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \dots, w_2, w_1)$$

- $P(this, is, a, reasonable, sentence) = P(this) \cdot (is|this) \cdot P(a|this, is) \cdot P(reasonable|this, is, a) \cdot P(sentence|this, is, a, reasonable)$

- So the focus shifts to estimating the probability that a word w_t will follow a bunch of other words

$$P(w_t | w_{t-1}, w_{t-2}, \dots, w_2, w_1)$$

- Sort of like a fill-the-blank (or cloze) question:
  - This is a reasonable ______
  - The black cat sat on the ______

Simple LMs – n-gram models

- A simplifying assumption (*Markov assumption*) is that:
  - $P(w_t | w_{t-1}, w_{t-2}, \dots, w_2, w_1) \approx P(w_t | w_t, w_t, \dots, w_{t-n+1})$

- The next word w_t approximately depends on a small window of (n-1) preceding words (*context*)

- If this assumption holds, for small n, we can compute all possible probabilities, as measured on a reference text (*corpus*)

- For $n = 2$, the next word only depends on the previous word!
  - $P(w_t | w_{t-1}, w_{t-2}, \dots, w_2, w_1) \approx P(w_t | w_{t-1})$

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences (*corpus*):
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	**Cat**	**Chased**	**Cheese**	**Happily**	**Mouse**	**The**
Ate	0	0	0	0	0	0	0
Cat	0	0	0	0	0	0	0
Chased	0	0	0	0	0	0	0
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	0	0	0	0	0	0	0
The	0	0	0	0	0	0	0

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences:
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	0
Cat	0	0	0	0	0	0	0
Chased	0	0	0	0	0	0	0
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	0	0	0	0	0	0	0
The	0	1	0	0	0	0	0

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences:
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	0
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	0
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	0	0	0	0	0	0	0
The	0	1	0	0	0	0	0

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences:
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	0
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	0	0	0	0	0	0	0
The	0	1	0	0	0	0	0

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences:
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	0
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	0	0	0	0	0	0	0
The	0	1/2	0	0	0	1/2	0

N-gram language model

- We can count the n-grams occurring in a training text
- Then, use those frequencies as an estimate of the probabilities
- Training sentences:
  - The cat chased the mouse happily
  - The mouse ate the cheese

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Estimation & generation

- We can use a language model to estimate probabilities for new sentences

- Generate a new sentence based on known probabilities
  - Starting from a given word, we can compute the probability of having any other word following it
  - Then, we choose one word based on this probability distribution
  - We add the new word to the already available sequence, and generate the next one
  - We call this, *autoregressive* generation

Estimating probabilities

- $P(the, cat, chased, the, cheese) \;= ?$
- $P(the, cat, chased, the, cheese) = P(the) \cdot P(cat|the) \cdot P(chased|the, cat) \cdot P(the|the, cat, chased) \cdot P(cheese|the, cat, chased, the)$
- *Markov assumption!*
- $P(the, cat, chased, the, cheese) \approx P(the) \cdot P(cat|the) \cdot P(chased|cat) \cdot P(the|chased) \cdot P(cheese|the)$
- Let’s assume $P(the) \;=\; 1$
  - (we empirically observe that all sentences in our corpus start with “the”)
- $P(the, cat, chased, the, cheese) \approx 1 \cdot \frac{1}{4} \cdot 1 \cdot 1 \cdot \frac{1}{4} = \frac{1}{16}$

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Generating new sentences

- Let’s start with “the”
- We can estimate the probability of any next word using our table:
  - P(cat|the) = 1/4
  - P(cheese|the) = 1/4
  - P(mouse|the) = 1/2
  - (all others are 0)
- 🎲 We can randomly sample from this distribution, and pick “mouse”
- Now the sentence is “the mouse”

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Generating new sentences

- We now have “the mouse”
- We are using bigrams, so the next word depends on “mouse”
- Probabilities:
  - P(ate|mouse) = 1/2
  - P(happily|mouse) = 1/2
- 🎲 Random sample! ➔ ate
- “The mouse ate”

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Generating new sentences

- “The mouse ate”
- Probabilities:
  - P(the|ate) = 1
- 🎲 “Random” sample! ➔ the
- “The mouse ate the”

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Generating new sentences

- “The mouse ate the”
- Probabilities:
  - P(cat|the) = 1/4
  - P(cheese|the) = 1/4
  - P(mouse|the) = 1/2
- 🎲 Random sample! ➔ mouse :(
- “The mouse ate the mouse”

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Generating new sentences

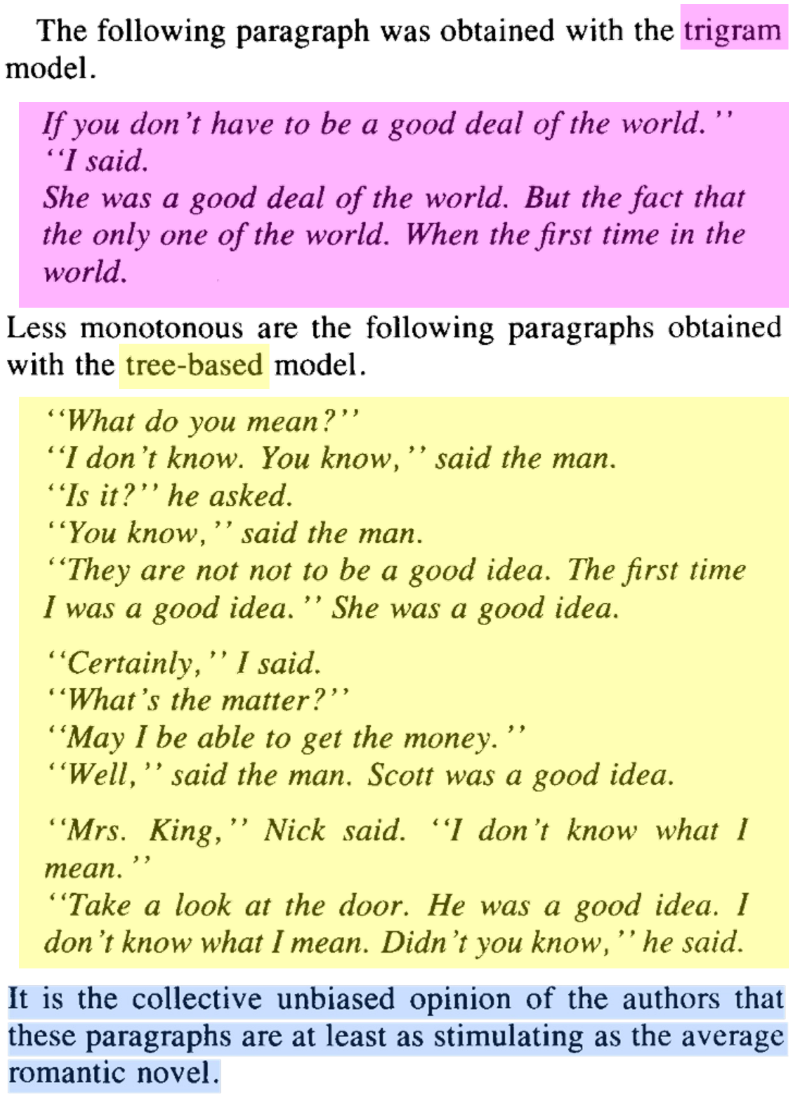
- “The mouse ate the mouse”
- Probabilities:
  - P(ate|mouse) = 1/2
  - P(happily|mouse) = 1/2
- 🎲 Random sample! ➔ happily 😱
- “The mouse ate the mouse happily”

	Ate	Cat	Chased	Cheese	Happily	Mouse	The
Ate	0	0	0	0	0	0	1
Cat	0	0	1	0	0	0	0
Chased	0	0	0	0	0	0	1
Cheese	0	0	0	0	0	0	0
Happily	0	0	0	0	0	0	0
Mouse	1/2	0	0	0	1/2	0	0
The	0	1/4	0	1/4	0	2/4	0

Limitations of n-grams

- *Data sparsity*
  - As n increases, the number of possible n-grams grows exponentially
  - Bigram, V words ➔ table size: $V \times V = V^2$
  - n-gram, V words ➔ table size: $V \times V \times \cdots V = V^n$
  - A small vocabulary or a small context size make the LM quite useless :(
- *Context limitations*
  - Only local context (previous n-1 words) are used to choose.
  - Longer sentences may require remembering what happened ”early on”
- *Lack of semantics*
  - Similar words are treated in the same way as completely different ones

Language models in the ‘90s



The following paragraph was obtained with the trigram model.

If you don’t have to be a good deal of the world.’’
‘‘I said.
She was a good deal of the world. But the fact that the only one of the world. When the first time in the world.

Less monotonous are the following paragraphs obtained with the tree-based model.

‘‘What do you mean?’’
‘‘I don’t know. You know,’’ said the man.
‘‘Is it?’’ he asked.
‘‘You know,’’ said the man.
‘‘They are not not to be a good idea. The first time I was a good idea.’’ She was a good idea.

‘‘Certainly,’’ I said.
‘‘What’s the matter?’’
‘‘May I be able to get the money.’’
‘‘Well,’’ said the man. Scott was a good idea.

‘‘Mrs. King,’’ Nick said. ‘‘I don’t know what I mean.’’
‘‘Take a look at the door. He was a good idea. I don’t know what I mean. Didn’t you know,’’ he said.

It is the collective unbiased opinion of the authors that these paragraphs are at least as stimulating as the average romantic novel.

- Language models are not a new concept
- In the ‘90s, researchers tried to build LMs based on existing techniques and computing capabilities
- With varying results… Mostly bad.

Potamianos, Gerasimos, and Frederick Jelinek. "A study of n-gram and decision tree letter language modeling methods." *Speech Communication* 24.3 (1998): 171-192.

Bigger and better Language Models

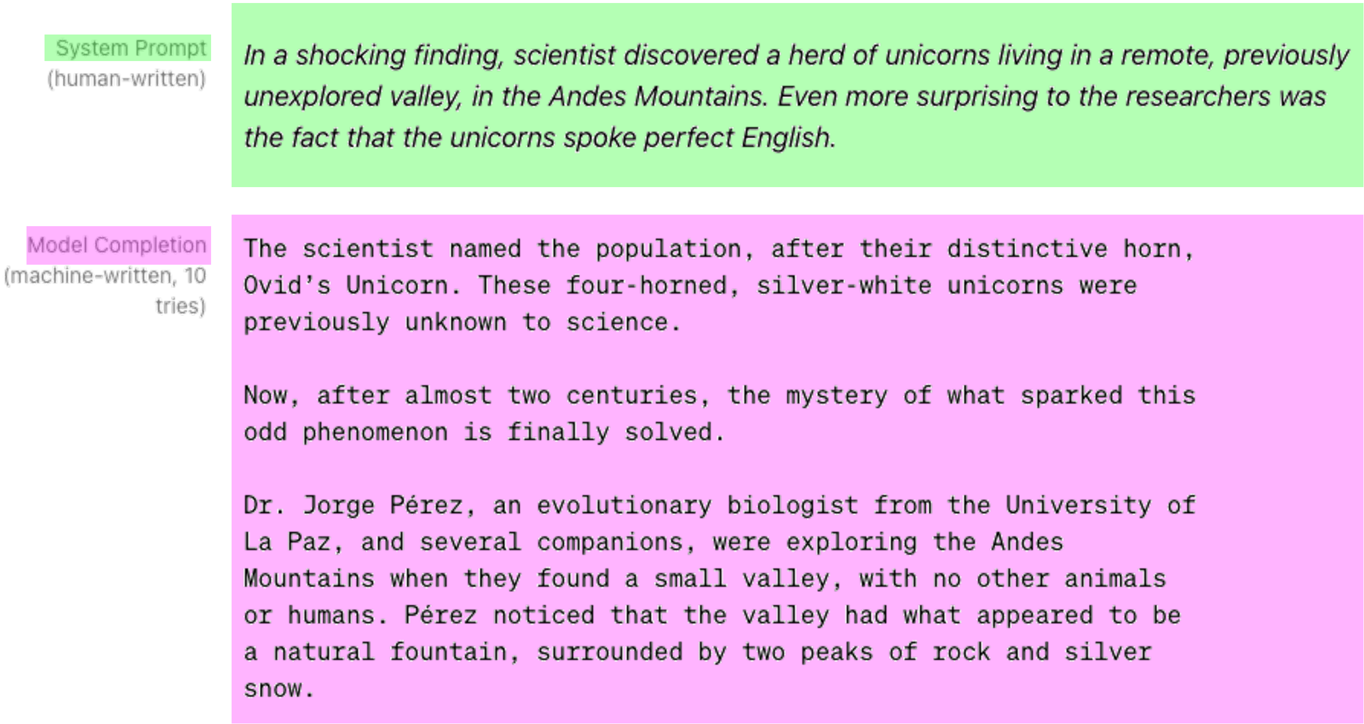
- We can see $P(w_t | w_{t-1}, w_{t-2}, \ldots, w_2, w_1)$ as a classification function

- "Given an *input* (the previous words in the sentence), predict the *next word* (class)"
  - Remember, we can often extract class probabilities from classifiers!

- Among the others, neural networks have been shown to be quite good at making those predictions!

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." *Advances in neural information processing systems* 13 (2000). -- https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf

Bigger and better Language Models



System Prompt (human-written)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Model Completion (machine-written, 10 tries)

```
The scientist named the population, after their distinctive horn,
Ovid's Unicorn. These four-horned, silver-white unicorns were
previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this
odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of
La Paz, and several companions, were exploring the Andes
Mountains when they found a small valley, with no other animals
or humans. Pérez noticed that the valley had what appeared to be
a natural fountain, surrounded by two peaks of rock and silver
snow.
```

- 2019 results are looking much better!
- GPT-2: obtained with a *decoder-only transformer* architecture, a “big” model and pretraining on lots of data
- Note: “Large” models existed before the “ChatGPT” boom (end of 2022) – and already performed quite well!

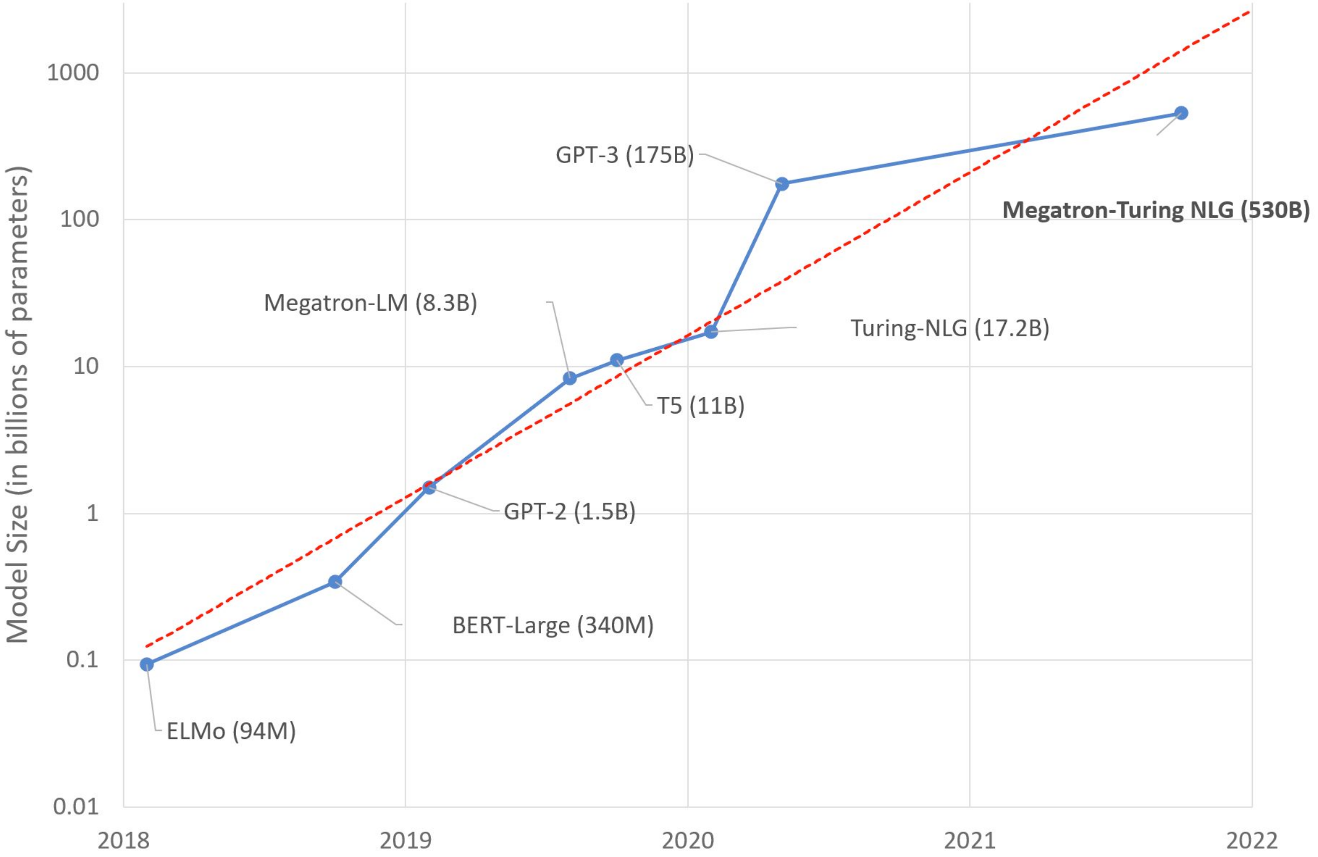
Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." *OpenAI blog* 1, no. 8 (2019): 9.

Putting the “Large” in LLM

- A recent trend in Language Models has been to:

1. ”Fix” the architecture (some variation of decoder-only transformer model)
2. Increase the model size

- Although this recipe will probably stop working at some point, it seems to scale “well”
    - However, some researchers noted that models were getting oversized & undertrained



https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

Takeaways

- Language Models are a probabilistic model of a language
- We can use LMs to
  - Compute the probability of any sentence, or
  - Generate new sentences
- Old-school models work quite poorly
- Using neural networks (➔ transformers) proved to work quite well
- Increasing the size of neural networks resulted in better models
  - Large Language Models!