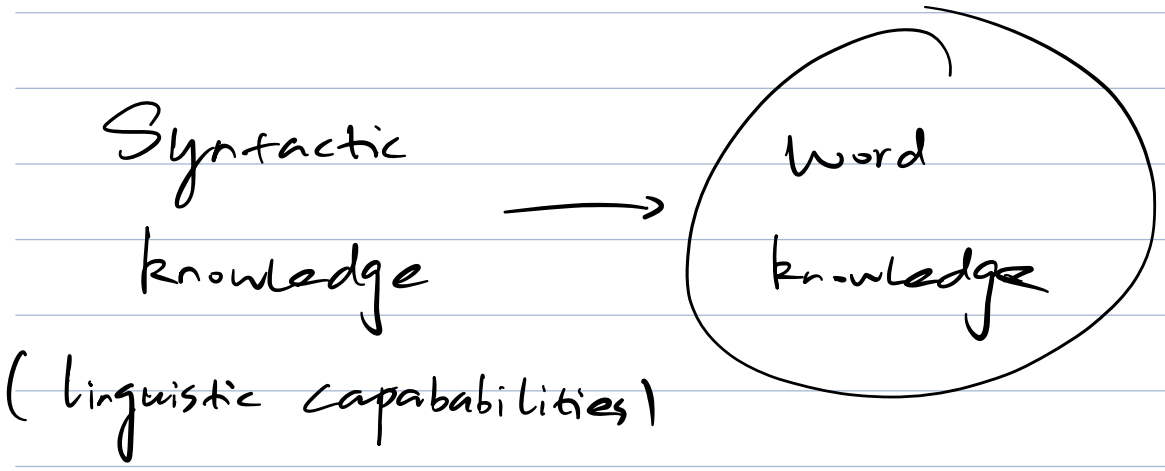
Language Model (LM):

probability distribution p over sequences of tokens $x_{1:L}$

$$p(x_1, \ldots, x_L)$$

sequence of tokens

(intuitively tells us how "good" a sequence of tokens is.)



(linguistic capabilities)

Autoregressive language models

chain rule of probability

$$p(x_{1:L}) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1, x_2) \cdots$$

$$p(x_L \mid x_{1:L-1})$$

$$= \prod_{i=1}^{L} p(x_i \mid x_{1:i-1})$$

Generation: generate an entire sequence

$$x_{1:L} \text{ from } p$$

for $i = 1, \ldots, L$:

$$x_i \sim p(x_i \mid x_{1:i-1})^{\frac{1}{T}}$$

$T \geq 0$ temperature parameter

that controls how much randomness (amount of variability in generation)

- $T = 0$: deterministically choose the most probable token x_i at each position i
- $T = 1$: sample "normally" from the pure language model
- $T = \infty$: sample from a uniform distribution over the entire vocabulary □

Annealed conditional probability distribution

$$p_T(x_i \mid x_{1:i-1}) \propto p(x_i \mid x_{1:i-1})^{\frac{1}{T}}$$

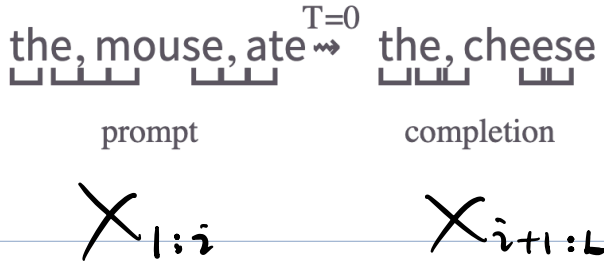
Conditional Generation:

specify some prefix sequence $x_{1:i}$

(prompt)

sample the rest $x_{i+1:L}$

(completion)



Language Models:

Claude Shannon

information theory 1948

Entropy:

$$H(p) = \sum_x p(x) \log \frac{1}{p(x)}$$

measure the expected number of bits any algorithm needs to encode (compress) a sample $x \sim p$ into a bitstring

entropy ↓ → structured ↑

↘ code length ↓ $\log \frac{1}{p(x)}$

Cross Entropy:

$$H(p, q) = \sum_x p(x) \log \frac{1}{q(x)}$$

Measure the expected number of bits (nats) needed to encode a sample $x \sim p$ using the compression scheme given by the model q

(representing x with a code of length $\frac{1}{q(x)}$)

$$H(p, q) \geq H(p)$$

- p: samples from the true data distribution
- q: model

Shannon Game (human language model)

Shannon n-gram models 1948

Noisy channel model

$$p(\text{text} \mid \text{speech}) \propto \underbrace{p(\text{text})}_{\text{language model}} \underbrace{p(\text{speech} \mid \text{text})}_{\text{acoustic model}}$$

N-gram models

token X_i only depends on the last $n-1$ characters $X_{i-(n-1):i-1}$

$$P(X_i \mid X_{1:i-1}) = P(X_i \mid X_{i-(n-1):i-1})$$

Cons:

$n \downarrow \longrightarrow$ capture long-range dependencies

$n \uparrow \longrightarrow$ statistically infeasible to get good estimates of the probabilities

local dependencies

Neural Language Models:

$$P(X_i \mid X_{i-(n-1):i-1})$$

RNNs, LSTMs:

Conditional distribution of a token x_i to depend on the entire context $x_{1:i-1}$

Transformer:

GPT-3 : 2048

fixed context length n

increase in size

in-context learning

✓ Reliability

✓ Social bias

✓ Disinformation

✓ Security

v Legal consideration

Foundation Models : Key Points

(1) Large-scale pre-trained language models
(PLMs, LLMs)

(2) can be instructed by prompts
to perform specific tasks