Language models

↳ an LM that assigns a probability to a given text

$P(\text{"students books opened their"}) < P(\text{"students opened their books"})$

$S = w_1\ w_2\ w_3\ w_4\ \dots\ w_n$

a LM computes

$$P(S) = P(w_1, w_2\ w_3 \dots w_n)$$
$$= P(w_{1 \dots n})$$

What about the "next word prediction"?

$$P(w_n \mid w_{1 \dots n-1})$$
$$= P(w_n \mid w_{<n})$$

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

$P(A,B)$ → joint prob

↳ prefix

$$P(A,B) = P(B|A)\,P(A)$$

$$P(S) = P(w_1, w_2, w_3 \ldots w_n)$$

$$= P(w_1)\,P(w_2|w_1)\,P(w_3|w_1,w_2)\ldots$$

$$= \prod_i^n P(w_i | w_{i-i})$$

---

training data: a set of documents that we will use to estimate these cond. probabilities

My favorite LLM is ____

↳ prefix

$$P(\text{Claude} \mid \text{My favorite LLM is}) \approx \frac{\text{Count}(\text{My fave LLM is Claude})}{\text{Count}(\text{My favorite LLM is})}$$

issues:

- ↳ storage
- ↳ sparsity
- ↳ staleness of data
- ↳ lack of sharing info between related/similar prefixes

n-gram models:

↳ we will approximate these probabilities by dropping context from the prefix

⇒ Markov assumption

$$P(\text{Claude} \mid \text{My fave. LLM is}) \approx P(\text{Claude} \mid \text{is}) \Rightarrow \frac{\text{count}(\text{is Claude})}{\text{count}(\text{is})}$$

bigram / 2-gram

3-gram / trigram

$$\approx p(\text{Claude} \mid \text{LLM is}) \Rightarrow \frac{\text{count}(\text{LLM is Claude})}{\text{count}(\text{LLM is})}$$

how many counts do you have to store for an n-gram model?

↳ vocab size V

↳ V^n

start of sent

end of sent

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$V = \{$ <s>, </s>, I, am, Sam, do, not, like, green, eggs, and, ham $\}$

↳ each unique element of vocab is called a **type**

↳ each instance of a type in a dataset is called a **token**

$$P(I \mid <s>) = \frac{2}{3}$$

$$P(\text{ham} \mid <s>) = 0$$

↳ sparsity

Smoothing: "steals" prob. from observed n-grams and gives it to unseen ones

$$P(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1} w_i) + 1}{\text{count}(w_{i-1}) + V}$$

add 1 smoothing

Given a training dataset, general goal is to find probabilities that make the training data as likely as possible

$$\max P(\text{training data})$$

↳ maximum likelihood estimation

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{<i})$$

↳ w/ long seqs, this gets super small, numerical underflow

↳ product ⇒ sum of log P

$$\log \prod_i P(w_i \mid w_{<i}) = \sum \log P(w_i \mid w_{<i})$$

↳ sum ⇒ avg.

$$\frac{1}{n} \sum \log P(w_i \mid w_{<i})$$

↳ per token

↳ compare docs of diff lengths

↳ Morse code

↳ "E" ⇒ dot

↳ "Q" ⇒ 4 dots / dashes

$-\log_2 P(x) \Rightarrow$ surprisal
bits you need to spend to encode x

↳ high $P \Rightarrow$ short code
low $P \Rightarrow$ long code

neg. log likelihood
NLL: $\frac{-1}{n} \sum \log P(w_i \mid w_{i:n})$

↳ cross-entropy loss

↳ avg. bits/token
nats/token
} want it to be as low as possible

---

let's say NLL = 3.2
what does that mean?

↳ consider a hypothetical model that chooses among k different equally likely

words at each position

$$P(w_i \mid w_{<i}) = \frac{1}{k} \quad \text{for all } i$$

$$-\log\left(\frac{1}{k}\right) = \log k, \qquad \text{avg. NLL} = \log k$$

what k would produce my NLL?

$$\log k = \text{NLL}$$

$$k = \exp(\text{NLL})$$

↳ perplexity

↳ training PPL

↳ test PPL