NPTEL ONLINE CERTIFICATION COURSES

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 04 : N-gram Language Models: Part 2

PROF. PAWAN GOYAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

CONCEPTS COVERED

- Smoothing in n-gram LMs
- Evaluation: Perplexity
- Sampling from the distribution for generation
- Larger n-grams

Some Important Points with N-gram LMs

- For unigram counts, P(w) is always non-zero
  - if our dictionary is derived from the document collection
- This won't be true of $P(w_k|w_{k-1})$. Let's take an example below.

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

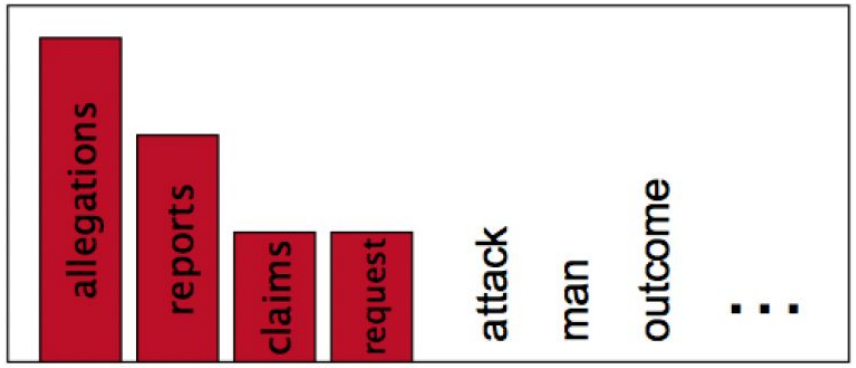
- ... denied the offer
- ... denied the loan

P(offer | denied the) = 0, the test sentence will be assigned a probability of 0!

Smoothing

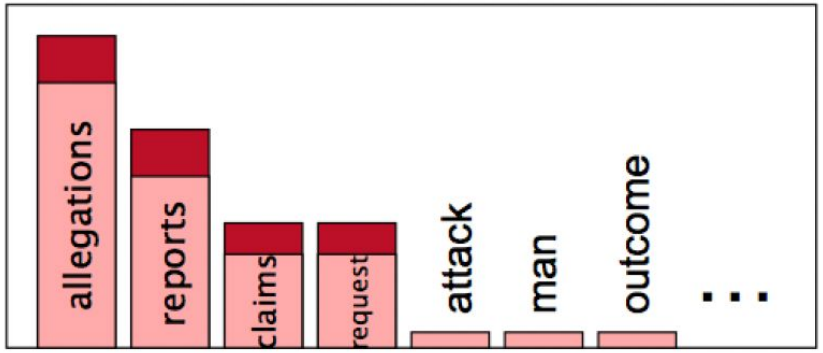
With sparse statistics

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-1 Smoothing

- Pretend as if we saw each word (N-gram) one more time that we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V}$

Add-1 Smoothing: Example

Given a corpus C, the bigram probability of “paper | question” is 0.3 and the count of occurrences of the word “question” is 600. What will be the frequency of the pair (question, paper) in the corpus C? → *We got this as 180*

Now, suppose the vocabulary size is 1210. What will be the probability of “paper | question” after add-1 smoothing?

$$P_{add\_1}(\text{“paper | question”}) = (freq(question, paper)+1)/(freq(question)+V)$$

$$= (180+1)/(600+1210)$$

$$= 0.1$$

More General Formulations

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+k}{c(w_{i-1})+kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+m(\frac{1}{V})}{c(w_{i-1})+m}$$

Interpolate with unigram probabilities instead

$$P_{UnigramPrior}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+mP(w_i)}{c(w_{i-1})+m}$$

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you know less about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram
- Interpolation works better

Source: https://web.stanford.edu/~jurafsky/slp3.

How to evaluate N-gram models

- "Extrinsic (in-vivo) Evaluation"

To compare models A and B

1. Put each model in a real task
   - Machine Translation, speech recognition, etc.
2. Run the task, get a score for A and for B
   - How many words translated correctly
   - How many words transcribed correctly
3. Compare accuracy for A and B

Source: https://web.stanford.edu/~jurafsky/slp3.

Intrinsic (in-vitro) evaluation

- Extrinsic evaluation not always possible
  - Expensive, time-consuming
  - Doesn't always generalize to other applications
- Intrinsic evaluation: **perplexity**
  - Directly measures language model performance at predicting words.
  - Doesn't necessarily correspond with real application performance
  - But gives us a single general metric for language models
  - Useful for large language models (LLMs) as well as n-grams

Source: https://web.stanford.edu/~jurafsky/slp3.

Training sets and test sets

We train parameters of our model on a **training set**.

We test the model's performance on data we haven't seen.

- A **test set** is an unseen dataset; different from training set.
  - Intuition: we want to measure generalization to unseen data
- An **evaluation metric** (like **perplexity**) tells us how well our model does on the test set.

Source: https://web.stanford.edu/~jurafsky/slp3.

Choosing training and test sets

- If we're building an LM for a specific task
  - The test set should reflect the task language we want to use the model for
- If we're building a general-purpose model
  - We'll need lots of different kinds of training data
  - We don't want the training set or the test set to be just from one domain or author or language.

Source: https://web.stanford.edu/~jurafsky/slp3.

Training on the test set

We can't allow test sentences into the training set

- Or else the LM will assign that sentence an artificially high probability when we see it in the test set
- And hence assign the whole test set a falsely high probability.
- Making the LM look better than it really is

This is called "Training on the test set"

Bad science!

Source: https://web.stanford.edu/~jurafsky/slp3.

Dev sets

- If we test on the test set many times we might implicitly tune to its characteristics
  - Noticing which changes make the model better.
- So we run on the test set only once, or a few times
- That means we need a third dataset:
  - A **development test set** or, **devset**.
  - We test our LM on the devset until the very end
  - And then test our LM on the **test set** once

Source: https://web.stanford.edu/~jurafsky/slp3.

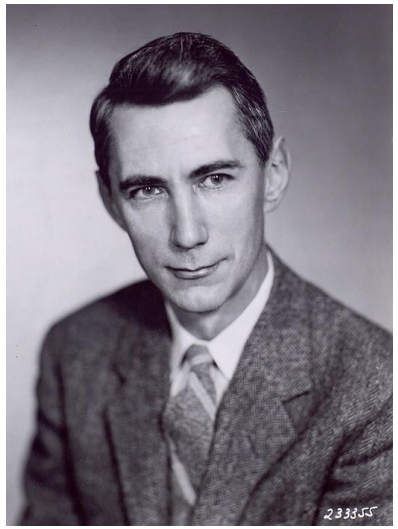
Intuition of perplexity as evaluation metric: How good is our language model?

Intuition: A good LM prefers "real" sentences

- Assign higher probability to “real” or “frequently observed” sentences
- Assigns lower probability to “word salad” or “rarely observed” sentences?

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 2: Predicting upcoming words



Claude Shannon

The Shannon Game: **How well can we predict the next word**?

- Once upon a _____
- That is a picture of a _____
- For breakfast I ate my usual _____

time	0.9
dream	0.03
midnight	0.02
…	
and	1e-100

Unigrams are terrible at this game (Why?)

A good LM is one that assigns a higher probability to the next word that actually occurs

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 3: The best language model is one that best predicts the entire unseen test set

- We said: a good LM is one that assigns a higher probability to the next word that actually occurs.
- Let's generalize to all the words!
  - The best LM assigns high probability to the entire test set.
- When comparing two LMs, A and B
  - We compute P_A(test set) and P_B(test set)
  - The better LM will give a higher probability to (=be less surprised by) the test set than the other LM.

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 4: Use perplexity instead of raw probability

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - Better: a metric that is **per-word**, normalized by length
- **Perplexity** is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 5: the **inverse**

Perplexity is the **inverse** probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

(The inverse comes from the original definition of perplexity from cross-entropy rate in information theory)

Probability range is [0,1], perplexity range is [1,∞]

Minimizing perplexity is the same as maximizing probability

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 6: N-grams

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

Bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Source: https://web.stanford.edu/~jurafsky/slp3.

Intuition of perplexity 7: Weighted average branching factor

Perplexity is also the **weighted average branching factor** of a language.

Branching factor: number of possible next words that can follow any word

Example: Deterministic language L = {red,blue, green}

Branching factor = 3 (any word can be followed by red, blue, green)

Now assume LM A where each word follows any other word with equal probability ⅓

Given a test set T = "red red red red blue"

$Perplexity_A(T) = P_A(\text{red red red red blue})^{-1/5} = 3$

- But now suppose red was very likely in training set, such that for LM B:
  - P(red) = .8 p(green) = .1 p(blue) = .1
- We would expect the probability to be higher, and hence the perplexity to be smaller:

$Perplexity_B(T) = P_B(\text{red red red red blue})^{-1/5}$

$= (.8 * .8 * .8 * .8 * .1)^{-1/5} \quad = .04096^{-1/5} \quad = .527^{-1} \quad = 1.89$

Holding test set constant:
Lower perplexity = better language model

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

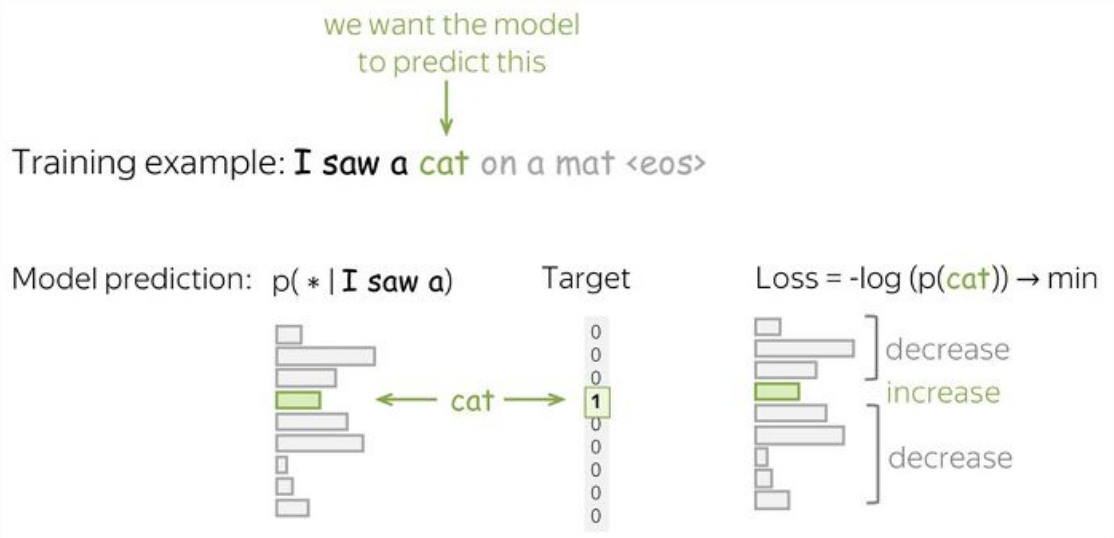
Heavily abbreviated history of LMs

1948: Claude Shannon models English

1948-2017: 🤯

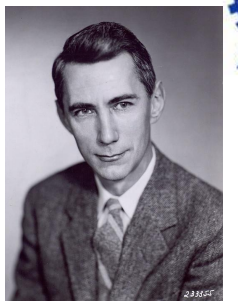
$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t})).$$

At each step, we maximize the probability a model assigns to the correct token. Look at the illustration for a single timestep.



Source: https://web.stanford.edu/class/cs224n/slides/cs224n-spr2024-lecture15-life-after-dpo-lambert.pdf

The Shannon (1948) Visualization Method Sample words from an LM



Claude Shannon

- Unigram:

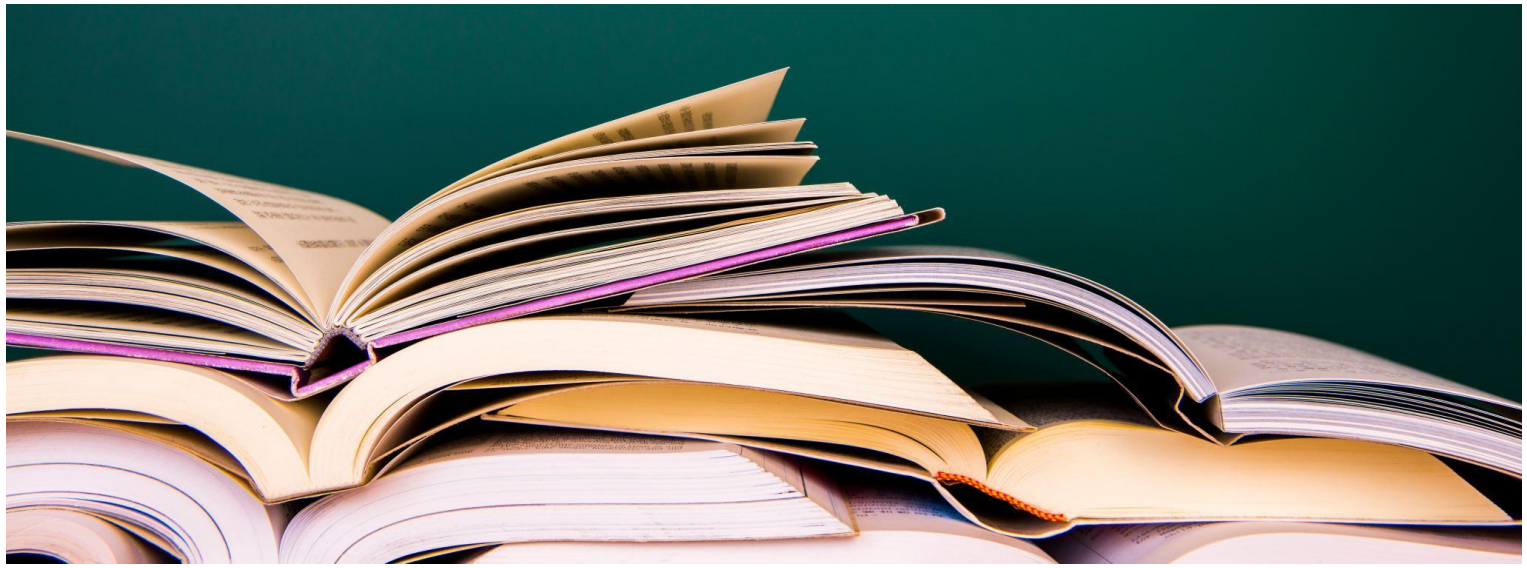
REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

- Bigram:

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Source: https://web.stanford.edu/~jurafsky/slp3.

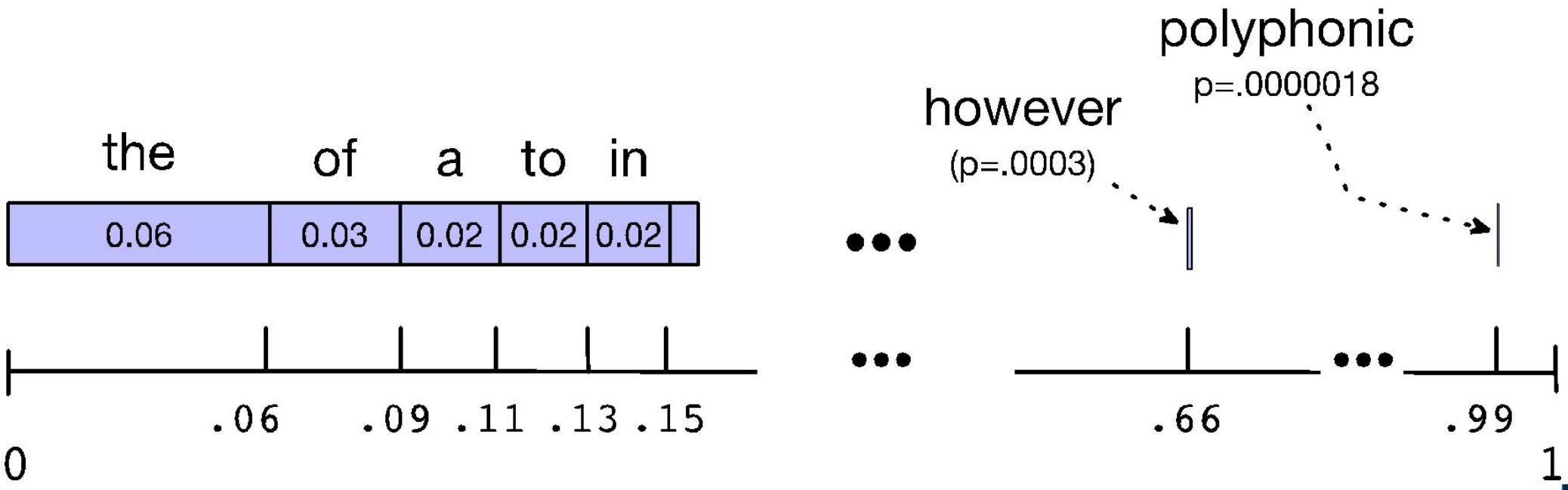
How Shannon sampled those words in 1948



"Open a book at random and select a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc."

Source: https://web.stanford.edu/~jurafsky/slp3.

Sampling a word from a distribution



The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram (<s>,w) as per its probability
- Choose a random bigram (w,x) as per its probability
- And so on until we choose </s>

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Note: there are other sampling methods

Used for neural language models

Many of them avoid generating words from the very unlikely tail of the distribution

We'll discuss when we get to neural LM decoding:

- Temperature sampling
- Top-k sampling
- Top-p sampling

Larger ngrams

- 4-grams, 5-grams
- Large datasets of large n-grams have been released
  - N-grams from Corpus of Contemporary American English (COCA) 1 billion words (Davies 2020)
  - Google Web 5-grams (Franz and Brants 2006) 1 trillion words)
  - Efficiency: quantize probabilities to 4-8 bits instead of 8-byte float

Newest model: infini-grams (∞-grams) (Liu et al 2024)

- No precomputing! Instead, store 5 trillion words of web text in **suffix arrays**.
- Can compute n-gram probabilities with any n!

∞-grams

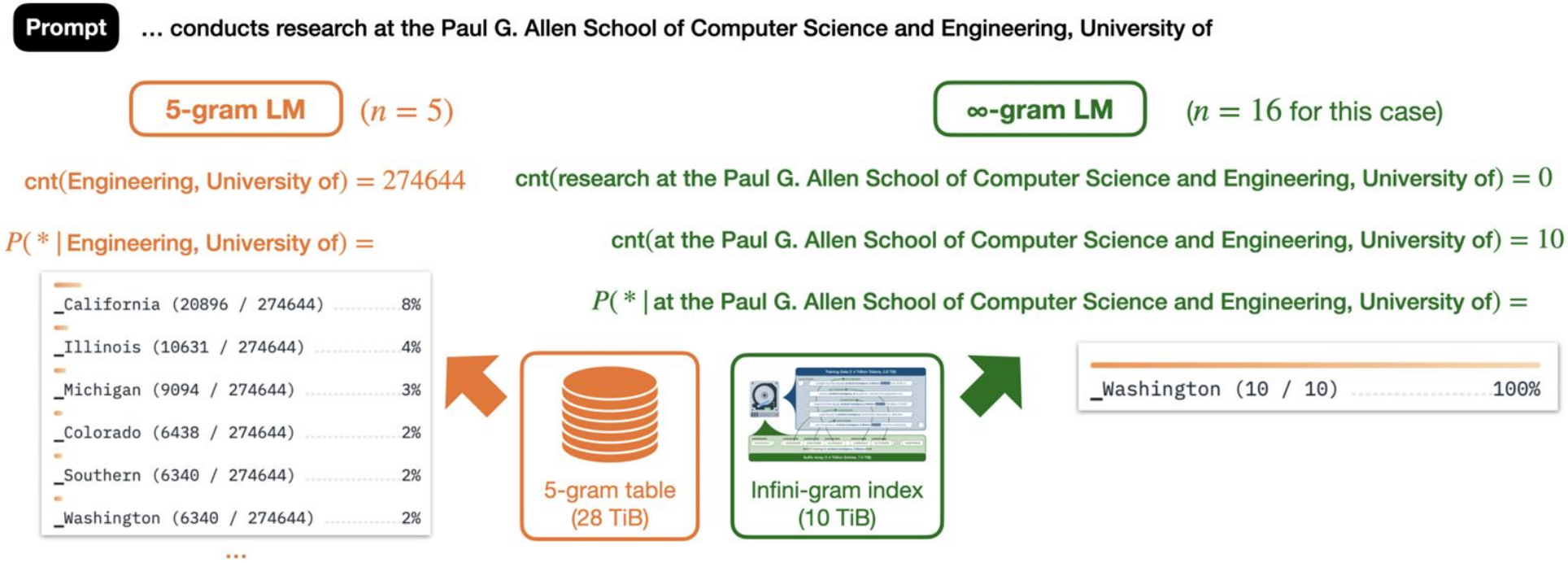
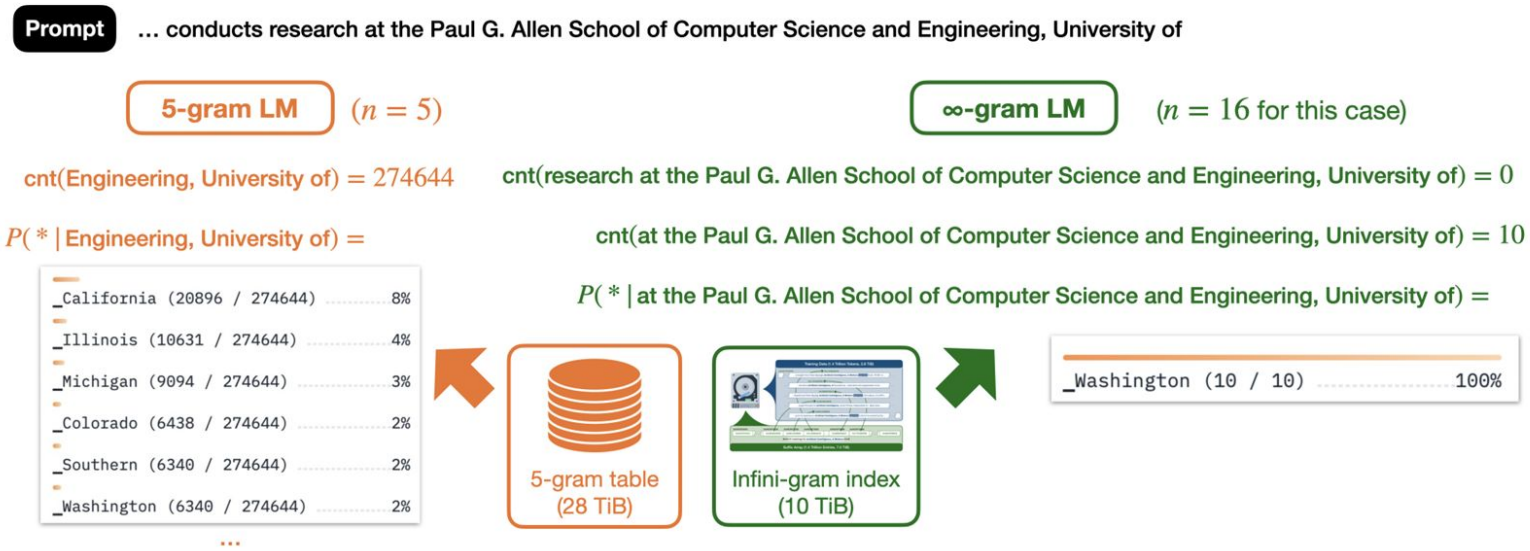


Figure 1: An example where a 5-gram LM gives an incorrect prediction but the ∞-gram gives the correct prediction by using the longest suffix of the prompt that has a non-zero count in the corpus. The counting and distribution estimate in ∞-gram LM are powered by our infini-gram engine.

Source: https://arxiv.org/pdf/2401.17377

∞-grams as a variant of back-off



$$P_\infty(w_i \mid w_{1:i-1}) = \frac{\text{cnt}(w_{i-(n-1):i-1} w_i \mid \mathcal{D})}{\text{cnt}(w_{i-(n-1):i-1} \mid \mathcal{D})}$$

where $w_{1:i-1}$ are all tokens preceding w_i in the document, and

$$n = \max\{n' \in [1, i] \mid \text{cnt}(w_{i-(n'-1):i-1} \mid \mathcal{D}) > 0\}$$

Source: https://arxiv.org/pdf/2401.17377

N-gram LM Toolkits

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

REFERENCES

Daniel Jurafsky and James H. Martin. 2024. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released August 20, 2024. [Chapter 3]


INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
1951
योगः कर्मसु कौशलम्
NPTEL
THANK YOU