Natural Language Processing

Introduction to language modelling

Marco Kuhlmann
Department of Computer and Information Science

Language modelling

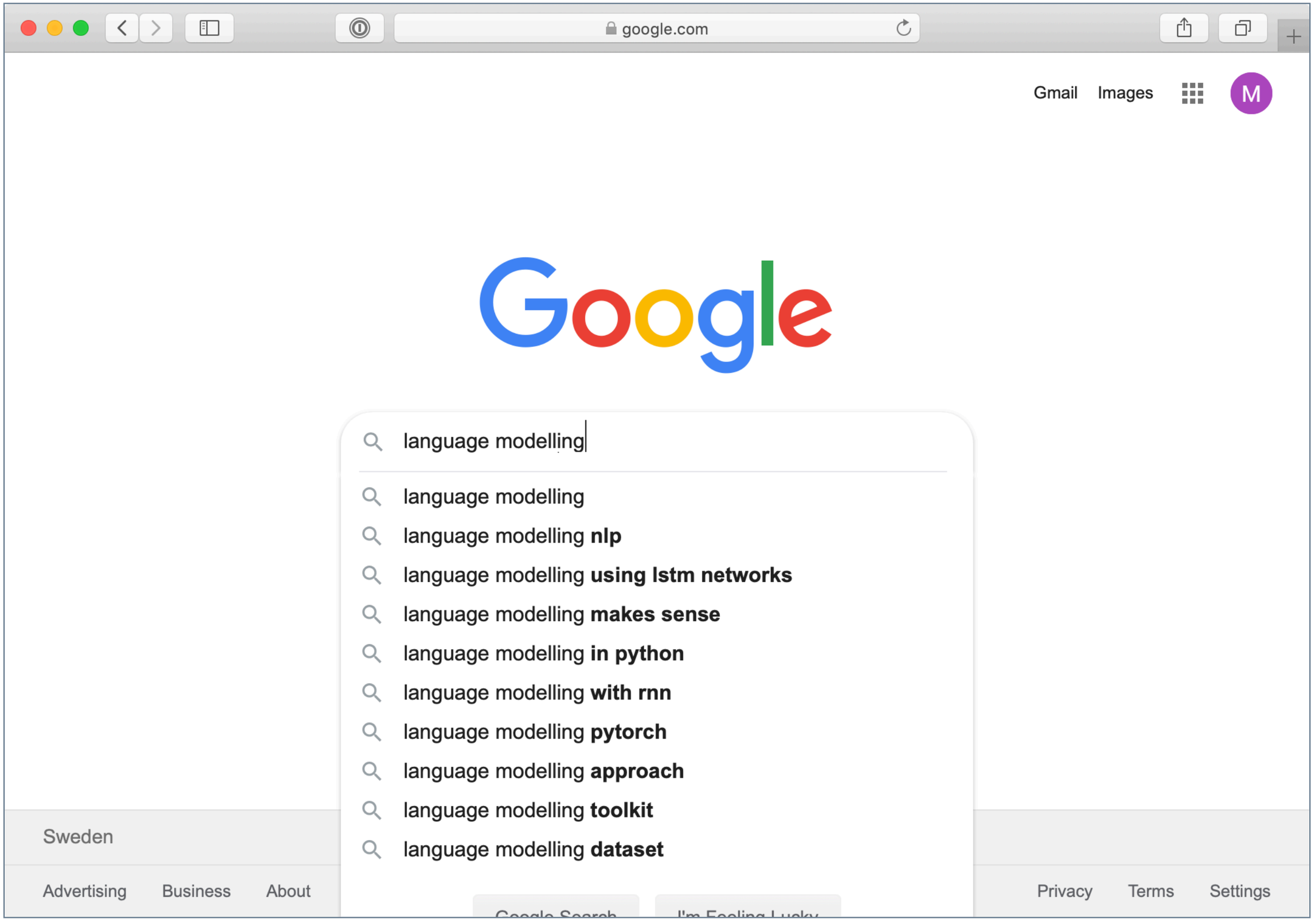
Jurafsky and Martin (2026), §3

- **Language modelling** is the task of predicting which word comes next in a sequence of words.
- More formally, given a sequence of words $w_1 \cdots w_t$, we want to know the probability of the next word, w_{t+1}:

$$p(w_{t+1} \mid w_1 \cdots w_t)$$

- We are assuming that w_{t+1} comes from a finite **vocabulary** V.

language models = classifiers


google.com
Gmail
Images
M
Google
language modelling
language modelling
language modelling nlp
language modelling using lstm networks
language modelling makes sense
language modelling in python
language modelling with rnn
language modelling pytorch
language modelling approach
language modelling toolkit
language modelling dataset
Sweden
Advertising
Business
About
Privacy
Terms
Settings

ChatGPT 4 (2023-12-31)

N-gram language models

Jurafsky and Martin (2026), §3

- An ***n*-gram** is a contiguous sequence of n words (or characters).

  Sherlock **Holmes** had **sprung out** and seized the intruder **by the collar**.

  unigram | bigram | trigram

- An ***n*-gram model** specifies conditional probabilities for the last word in an n-gram, given the previous words:

$$p(w_n \mid w_1 \cdots w_{n-1})$$

Intuition behind n-gram models

Jurafsky and Martin (2026), §3.1

- By the chain rule, the probability of a sequence of N words can be computed using conditional probabilities as

$$p(w_1 \cdots w_N) = \prod_{k=1}^{N} p(w_k \mid w_1 \cdots w_{k-1})$$

- To make probability estimates more robust, we approximate the full history $w_1 \cdots w_N$ by overlapping n-gram windows:

$$p(w_1 \cdots w_N) = \prod_{k=1}^{N} p(w_k \mid w_{k-n+1} \cdots w_{k-1})$$

Formal definition of an n-gram model

n	the model's order (1 = unigram, 2 = bigram, …)
V	a finite set of possible words; the vocabulary
$p(w\|u)$	a probability that specifies how likely it is to observe the word w after the context $(n-1)$-gram u

one value for each combination of a word w and a context u

Estimation of n-gram models

Jurafsky and Martin (2026), §3.1.2

- The simplest method for estimating *n*-gram models is **maximum likelihood estimation (MLE)**.

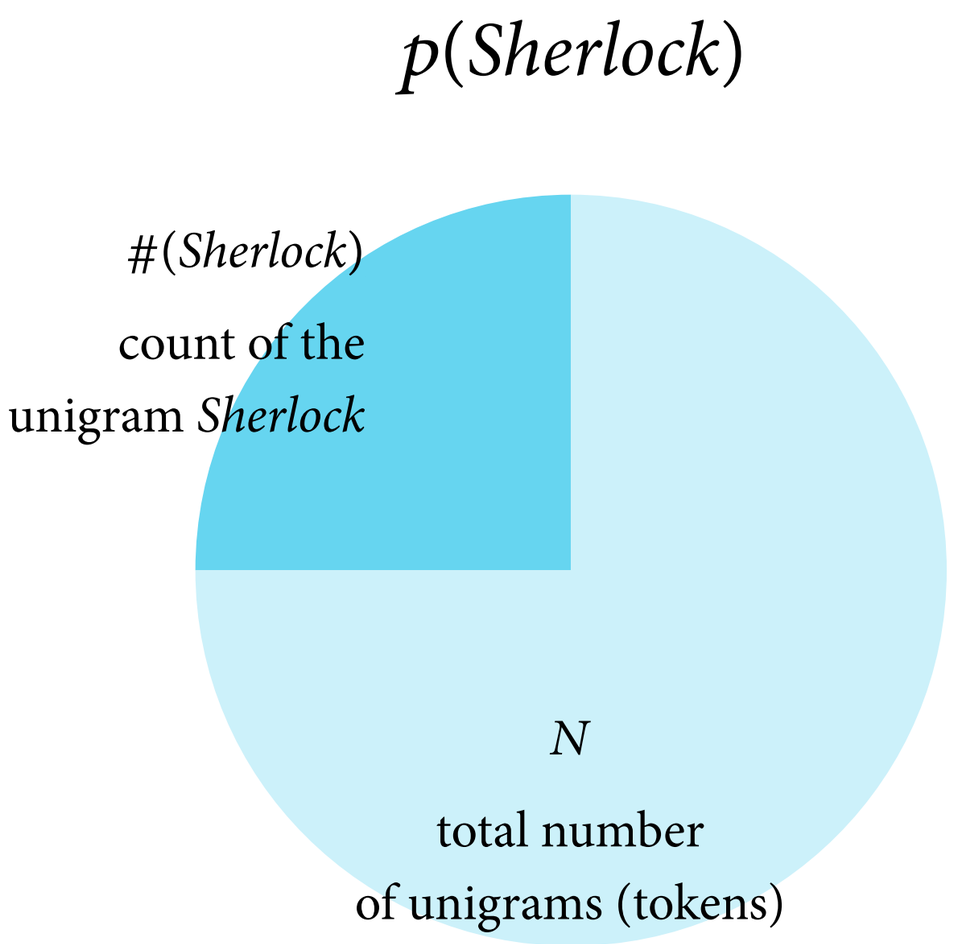
  maximise the likelihood of the observations given the parameters

- We want to find model parameters (here, probabilities) that maximise the likelihood of some text data.

- It turns out that we can solve this problem by simply counting occurrences of *n*-grams and normalising.

  formal derivation uses Lagrange multipliers

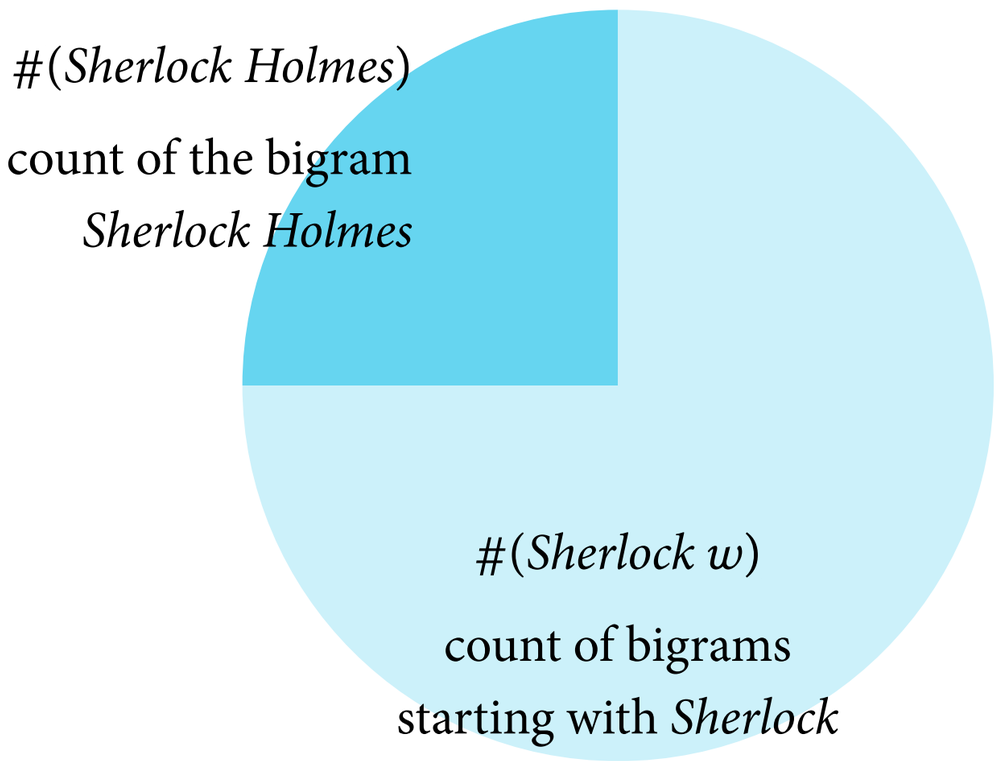
MLE of unigram probabilities



$$p(w) = \frac{\#(w)}{N}$$

MLE of bigram probabilities

$$p(Holmes \mid Sherlock)$$



$$p(w \mid u) = \frac{\#(uw)}{\#(u\bullet)}$$

$$p(w \mid u) = \frac{\#(uw)}{\#(u)}$$

Evaluating language models

Jurafsky and Martin (2026), §3.2

- **Intrinsic evaluation**

  How does the method or model score with respect to a given evaluation measure?

  examples from classification: precision and recall

- **Extrinsic evaluation**

  How much does the method or model help the application in which it is embedded?

  predictive input, machine translation, question answering

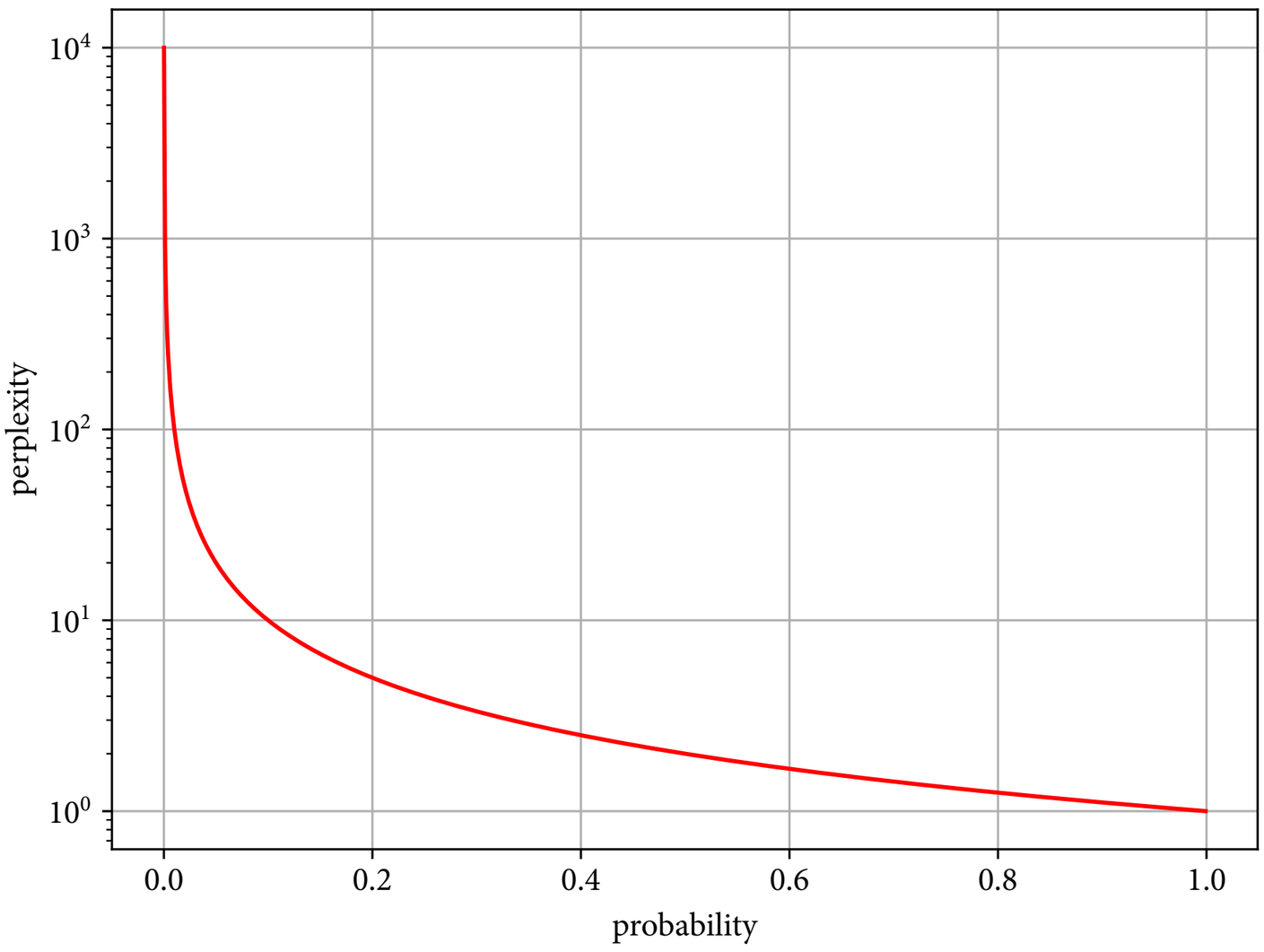
Perplexity

Jurafsky and Martin (2026), §3.3

- Intrinsic evaluation of language models is based on the likelihood that a model assigns to held-out data.
- Formally, we compute the cross-entropy between two probability distributions: a language model and the empirical distribution.
- This cross-entropy is usually presented as **perplexity**:

$$e^{-\frac{1}{N}\log P(w_1 \cdots w_N)}$$

Perplexity



Sparsity problems

Jurafsky and Martin (2026), §3.6

What if *students opened their w* does not occur in data? Then *w* has probability 0.

$$p(w \mid \text{students opened their}) = \frac{\#(\text{students opened their } w)}{\#(\text{students opened their})}$$

What if *students opened their* does not occur in data? Then we have no probability at all!

Possible solutions: smoothing, discounting

Possible solutions: back-off, interpolation

Example attributed to Abigail See

Smoothing

Jurafsky and Martin (2026), §3.6

- In **smoothing**, we "spread out the probability mass" over the possible outcomes more evenly than MLE would do.
- A substantial amount of research in language modelling has been devoted to the development of advanced smoothing techniques.

  additive smoothing, absolute discounting, Kneser–Ney smoothing, …

The relation between smoothing and perplexity

- When smoothing a language model, we are redistributing probability mass to outcomes we have never observed.
- This leaves a smaller fraction of the probability mass to the outcomes we actually *did* observe during training.
- The more probability we are taking away from observed outcomes, the higher the perplexity on the training data.