Deep Learning for Natural Language Processing

N-gram language models

Marco Kuhlmann
Department of Computer and Information Science
Linköping University

WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

N-gram language models

Eisenstein § 6.1

- 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

- 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 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

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

  maximise the likelihood of the observations given the parameters

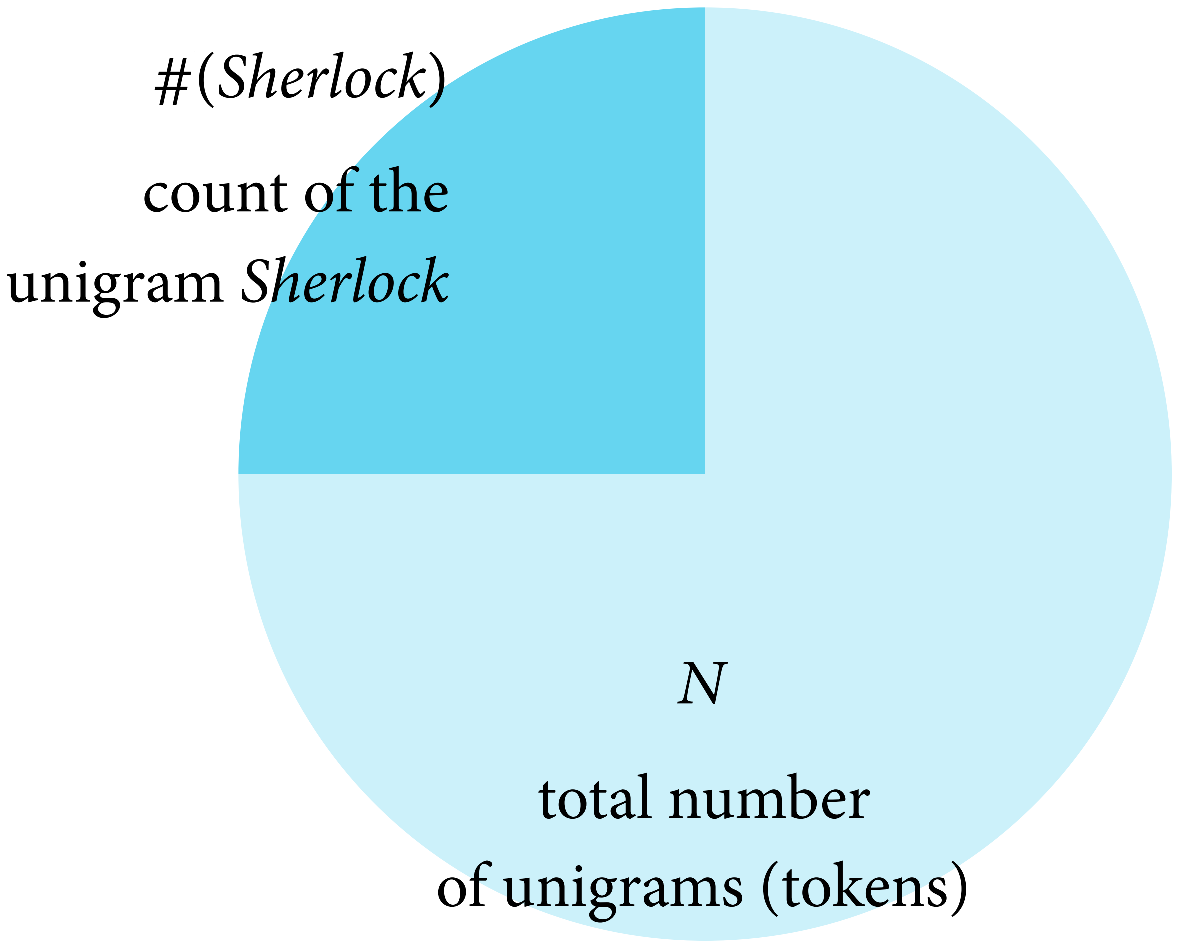
- We want to find model parameters (here, probabilities) that maximise the likelihood of some text data.

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

  formal derivation uses Laplace multipliers

MLE of unigram probabilities

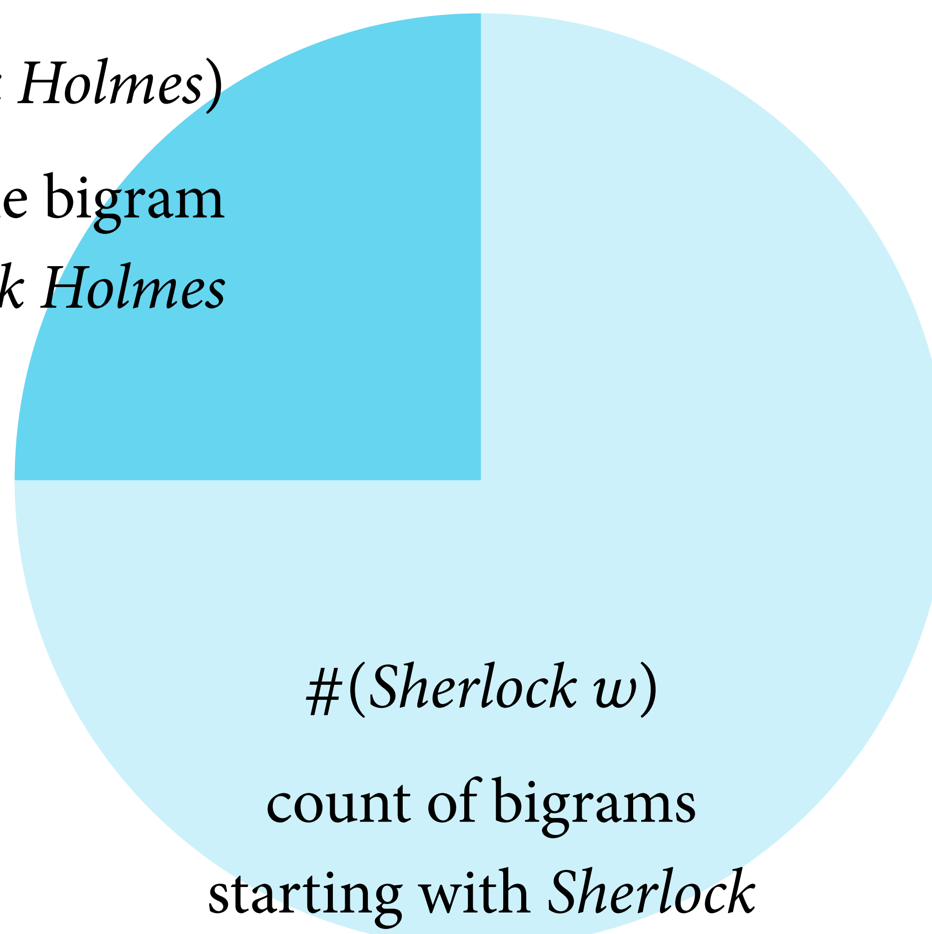
$P(Sherlock)$



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

MLE of bigram probabilities

$P(Holmes \mid Sherlock)$



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

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

Sparsity problems

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

Possible solutions: smoothing, discounting

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

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

Possible solutions: back-off, interpolation

Example attributed to Abigail See

Smoothing

Eisenstein § 6.2.1

- 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, …

Interpolation

Eisenstein § 6.2.3

- In an **interpolated language model**, probabilities are weighted sums of probabilities across progressively simpler models.

- For example, for an interpolated trigram model the probability $\bar{P}(w_3 \mid w_1 w_2)$ would be defined as

$$\lambda_3 P(w_3 \mid w_1 w_2) + \lambda_2 P(w_3 \mid w_2) + \lambda_1 P(w_3)$$

- The interpolation weights (λ) can be learned; in particular, they can be estimated from held-out data.

Out-of-vocabulary words

- A new text may contain words that are **out-of-vocabulary**. For these, none of the proposed solutions will help.
- A simple way to deal with out-of-vocabulary words is to restrict the vocabulary to the most frequent words, and to convert all other tokens to a special 'unknown word' token, UNK.
- Then, when we compute the probability of a new text, we first replace every out-of-vocabulary word with UNK.