Specialization Module

Speech Technology

Timo Baumann
baumann@informatik.uni-hamburg.de

UNIVERSITÄT HAMBURG, DEPARTMENT OF INFORMATICS
NATURAL LANGUAGE SYSTEMS GROUP

Language Modelling

The Speech Recognition Task

- Given a language $\mathcal{L}$
- and a sensory impression (observation) **O**
  - sequence of (MFCC) parameters over sliding windows
- we search $\hat{\mathbf{W}}$ in $\mathcal{L}$ such that
  - $\hat{\mathbf{W}} = \arg\max \mathbf{W} : P(\mathbf{W}|\mathbf{O})$
    the *most likely* word sequence given the observation
  - $\hat{W} = \arg\max W : P(O|Ph) \times P(Ph|W) \times \mathbf{P(W)}$

What information do you/humans use when estimating the likelihood of word sequences?

small groups, 3 minutes

Language Modelling

assigns a probability to every word sequence $\mathbf{W}$ in $\mathcal{L}$

- $\mathcal{L}$ is a *closed* language
  - the vocabulary of $\mathcal{L}$ is fixed
  - only word sequences in $\mathcal{L}$ can be recognized
    - no handling of *out-of-vocabulary* words (OOV)
  - no matter what the input,
    a word sequence in $\mathcal{L}$ will be recognized
  - Example: let $\mathcal{L}$ contain all German even numbers
    I say "drei", the recognizer considers "zwei" or "dreißig"
    - how to reject hypotheses when OOV words are spoken?

Language Modelling

assigns a probability to every sentence **W** in $\mathcal{L}$

- two types
  - structural: weighted grammar (PCFG)
    - cannot (easily) be learned from data → manually constructed
    - no probabilities for partial sentences, only for complete sentences
      → this makes the speech recognition search less efficient
    - simplifies natural language understanding (NLU)
      → often used in applied spoken dialogue systems
  - surface-based: N-Gram model
    - next word's probability computed from previous N-1 words
    - probability of the sequence is approximated by concatenating the probabilities of subsequences of length N

Deriving the N-Gram Model:

- problem is data sparsity, we simply can't estimate P(W) for many sentences by looking at data
- however, $P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2) \ldots$
  $$P(w_n|\underbrace{w_1,w_2, \ldots, w_{n-1}}_{\text{word history}})$$
- assumption: recent history ismore relevant than more distant history → limit history to a fixed number of words

Definition of the N-Gram Model

$\mathbf{W} = w_1, w_2, \ldots w_n = w_{1..n}$

- using the chain rule of probability, we get:

  $P(W) = \prod_{k=1..n} P(w_k|w_{1..k-1})$

  - each word's probability depends on its contextual history
- N-Grams approximate the contextual history:

  $P(w_k|w_{1..k-1}) \approx P(w_k|w_{k-N..k-1})$
- the larger N, the better the approximation
- however, the larger N,
  the larger the original problem of data sparsity

A simple example:

"the dog barks"

- simplest form: unigrams (N=1)

  $P(\text{the dog barks}) \approx P(\text{the}) \times P(\text{dog}) \times P(\text{barks})$

  – not accurate as context is completely ignored
    "dog" is more likely than e.g. "from" after "the"
    "the dog" vs. "the from"

- context: bigrams/trigrams

  $P(\text{the dog barks}) \approx P(\text{the}|\langle s\rangle) \times P(\text{dog}|\text{the}) \times P(\text{barks}|\text{dog}) \times P(\langle /s\rangle|\text{barks})$

A simple example:

"the dog barks"

- simplest form: unigrams (N=1)

  $P(\text{the dog barks}) \approx P(\text{the}) \times P(\text{dog}) \times P(\text{barks})$

  - not accurate as context is completely ignored
    "dog" is more likely than e.g. "from" after "the"
    "the dog" vs. "the from"

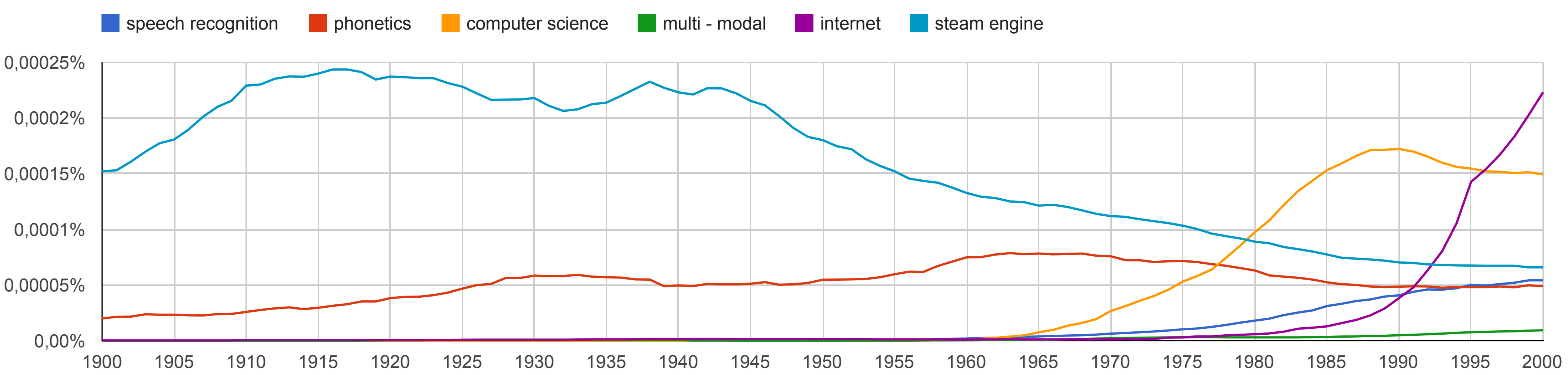
- context: bigrams/trigrams

  $P(\text{the dog barks}) \approx P(\text{the}|\langle s \rangle) \times P(\text{dog}|\text{the}) \times P(\text{barks}|\text{dog}) \times P(\langle /s \rangle|\text{barks})$

start/end
markers

Relative Frequencies: Counting Words over Time

- probability of words is estimated by counting their relative occurrence in large amounts of textual data



Counts from Google N-Grams: http://books.google.com/ngrams

From Counts to Probabilities

$$P\left(w_n \mid w_{n-1}\right) \approx \frac{Count\left(w_{n-1} w_n\right)}{Count\left(w_{n-1}\right)}$$

From Counts to Probabilities

- count occurrence of N-gram $w_1..w_n$ in data
- divide by count of $w_1..w_{n-1}$ in data
- for bigrams: $P(w_n|w_{n-1}) \approx \frac{Count(w_{n-1}w_n)}{Count(w_{n-1})}$
- what happens if some count is zero?

An example trigram

- when looking at the Billion Word Corpus:

  P('s|the world) = .33
  P(.|the world) = .14
  P(,|the world) = .10
  P(and|the world) = .02

  P(*everything else*|the world) = .41

- vocabulary limited to 100000 words;
  99996 words share less than half the probability

- among those words are things like:
  symbols (42 times in first 10 million words),
  Sinatra (19 times in first 10 million words),
  introspection (3 times in first 10 million words)

Zipf's law

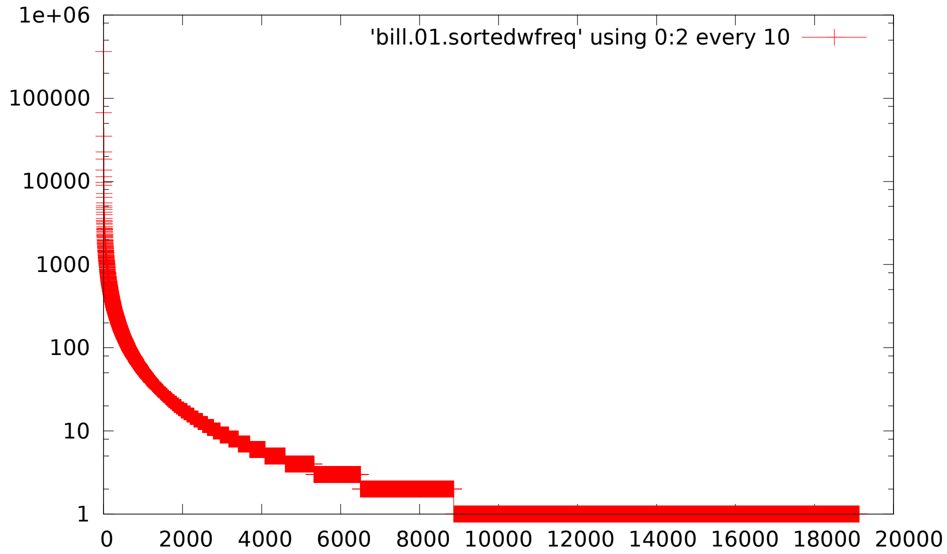
the frequent occurrence of rare events

- language uses few symbols very often and the vast majority of symbols very infrequently

Zipf's law

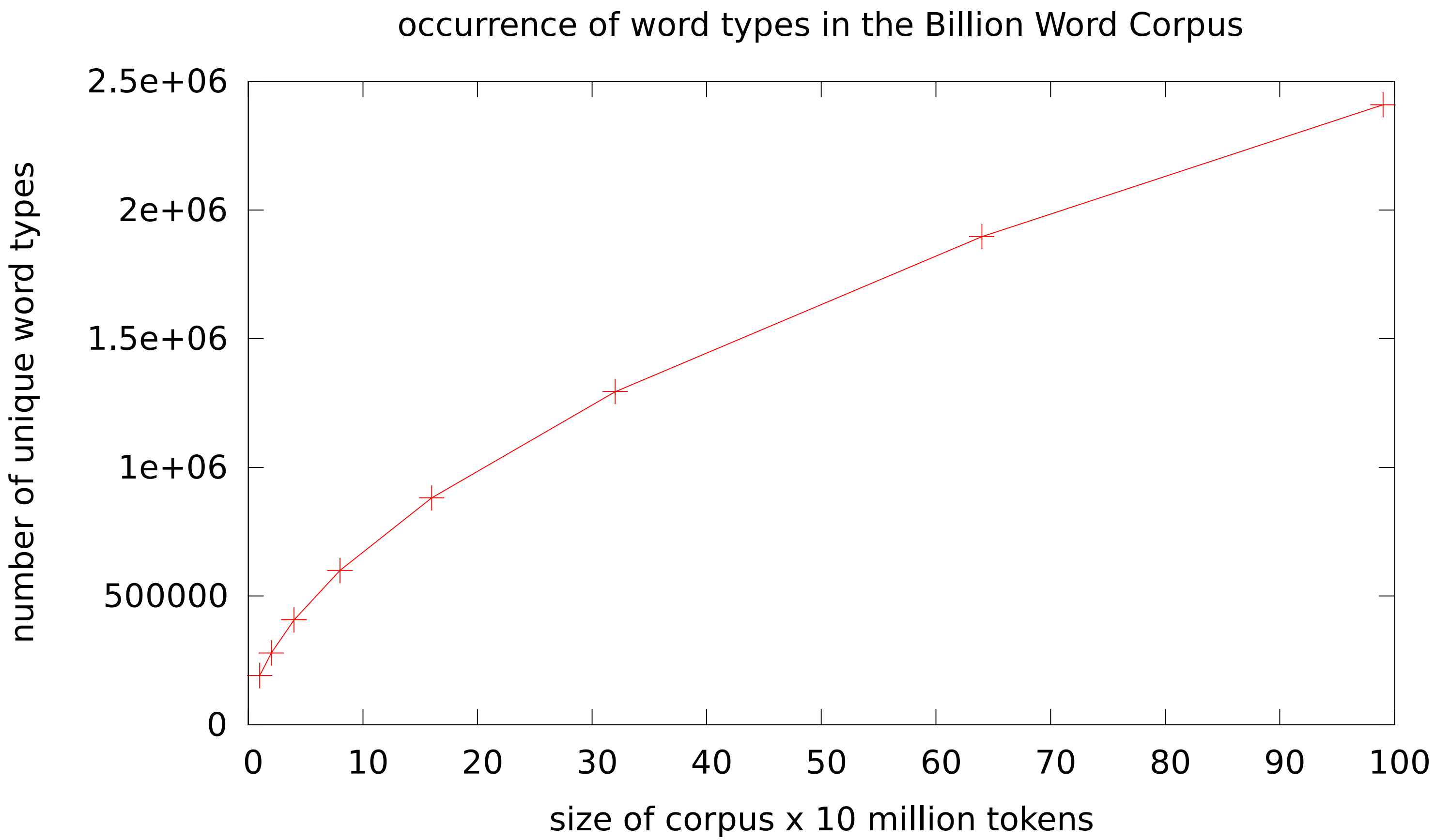
the frequent occurrence of rare events

- language uses few symbols very often and the vast majority of symbols very infrequently



Zipf's law

- most things that you will see in practice, you will never have observed in your training material
- not even the vocabulary is saturated at a billion words (half of the vocabulary, you've just seen once)



Data-Sparsity and Interpolation

- fix the vocabulary to some number that you like. There's nothing that you can do for the less frequent words.
  - change infrequent words to <UNK> (or some other tag)
  - remove sentences that contain infrequent words
- deal with data sparsity of N-grams for the remaining corpus
  - move some probability mass to non-occurring N-grams (discounting)
  - back-off to N-1 gram if N-gram count is zero
  - use a mix of N, N-1, N-2, … N-Grams,
carefully estimate ideal mixture parameters
  - use a mix of N-Gram models estimated on different data

Shifting Probability Mass to Unseen Events

- add count of 1 to every N-gram count before estimating probabilities (has largest effect on zero-occurrence N-grams, → Laplace discounting)
  - generalization add α instead of 1, estimate α on development data
- better: estimate the probability for an N-gram that does not occur in training based on N-grams that occurred once
  - generalization: of N-gram that occurred X times based on those that occurr X+1 times (→ Good-Turing discounting)

N-gram Backoff

- we may never have seen neither „*Scottish beer drinkers*“ nor „*Scotting beer eaters*“ in our corpus (e.g. American data)
  - simple discounting will assign identical probabilities, smarter discounting may do slightly better
- how about „*beer drinkers*“ vs. „*beer eaters*“?
- backoff to lower-order n-gram
  - however, now we are mixing probability spaces → add a weighing factor (backoff weight) to fix this, can be computed during model estimation

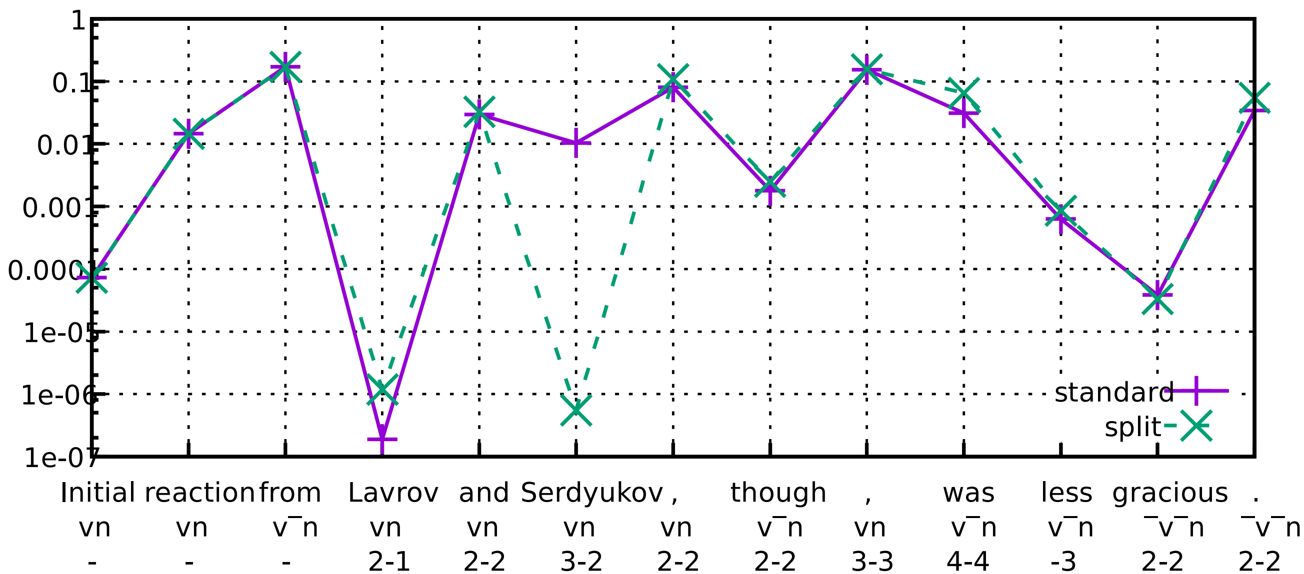
Example from Philipp Koehn's slides on this topic, see references.

Advanced smoothing methods

- even better: shift probability mass based on diversity of words predicted by a history → Witten-Bell discounting
- still better: shift mass based on diversity of histories → Kneser-Ney discounting
- combine with interpolation across model orders
- Kneser-ney discounting with interpolation usually works best
  - and by far outperforms LSTMs :-)

Combining Language Models with Different Characteristics

- previous slide: LSTMs are not as good as N-gram models
  - however, they make different kinds of mistakes
    - $P(W) = \lambda P_1(W) + (1\text{-}\lambda P_2(W))$
  - combination of two models is (almost) always better than each individual model (averaging effect)
    - reason: grave mistakes are improved by a larger magnitude than small improvements are reduced



Evaluating Language Models

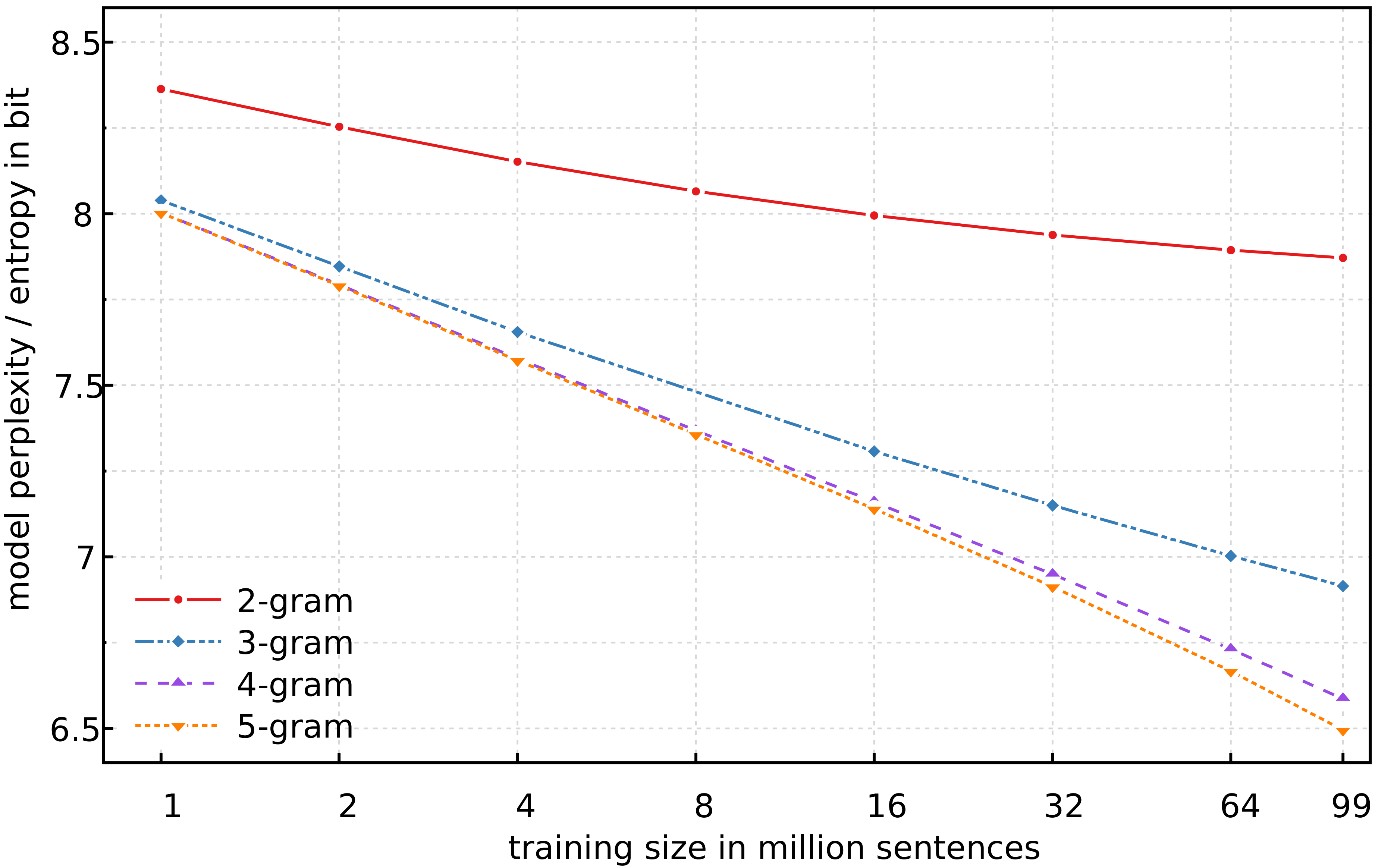
- LM should be a good source of information ...
  (→ information theory)
  ... *in general* (we approximate with *some* test material)
- test performance on unseen(!) material:
  - cross-entropy: estimate number of bits necessary to encode each word in a sentence given the language model's predictions:

    $$\hat{H} = -\frac{1}{m}\log_2\left(P\left(w_1, w_2, w_3, \ldots, w_m\right)\right)$$

  - above measure is in bit (frequent values ~5-10 bit)
  - more frequently used: 2^H is called *perplexity*
    - interpretation as average branching factor after each word

More Data is Better Data

one of the largest freely available corpora has barely enough data to saturate bigram training.



Summary

- $\hat{W} = \arg\max W : P(O|Ph) \times P(Ph|W) \times P(W)$
  - P(W): Word Sequence Model → N-Gram
- N-Gram training is simple (counting) and feasible on large amounts of data
- the limiting factor is often the data
  more degrees of freedom → less data per item → ...
- „more advanced“ approaches interpolate with Kneser-Ney interpolating 5-gram models to get high performance

Thank you.

baumann@informatik.uni-hamburg.de

https://nats-www.informatik.uni-hamburg.de/SLP16

Further Reading

- Introduction to Language Modelling:
  - D. Jurafsky & J. Martin (2009): *Speech and Language Processing*. Pearson International. InfBib: A JUR 4204x
- Particularly good explaination (in my view) including details in:
  - Philipp Koehn (2010): Statistical Machine Translation. Cambridge University Press. InfBib: A KOE 45521

Notizen

Desired Learning Outcomes

- know that N-gram models are a good representation of language and be able to explain why
- understand the problems arising from the estimation of probabilities from observations, in particular given Zipf's law
- remedies: smoothing, interpolation across N-gram order