TDDE16/732A92 Text Mining (2017)

Language modelling

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Language models

- A **statistical language model** is a probability distribution over sequences of words in some language.

  assumes a fixed vocabulary of atomic words

- Language models can also be defined on characters or sounds.

- Recent years have seen the rise of **neural language models**, which use word embeddings instead of atomic words.

Language models in information retrieval

- In the query likelihood model, we associate a separate language model d with every document in the collection.
- We use this model to derive probabilities of the form $P(q|d)$ and rank documents by computing

$$P(d \mid q) = \frac{P(q \mid d)P(d)}{P(q)} \qquad \text{(Bayes' rule)}$$

- Intuitively, we ask: 'How likely is it that the query q is observed as a random sample from the language model for d?'

Language models in predictive text input

ping guo gong si → 蘋果公司

ping →

凴塀琕聠箳評蓱諤鵧乒
俜凭凴呯坪塀娉屏屛岼
帡帲幈平慿憑枰檘泙洴
涄淜焩玶瓶甁甹砯竮竼
箳絣聠胓艵苹荓萍蓱蘋
蚲蛢評评軿輧郱頩鮃鲆

Language models for language identification

- **Language identification** is the problem of detecting the main language of a text document d.

  Note that one and the same document could use multiple languages.

- We rank languages based on the probability of observing d as a random sample of the candidate language L.

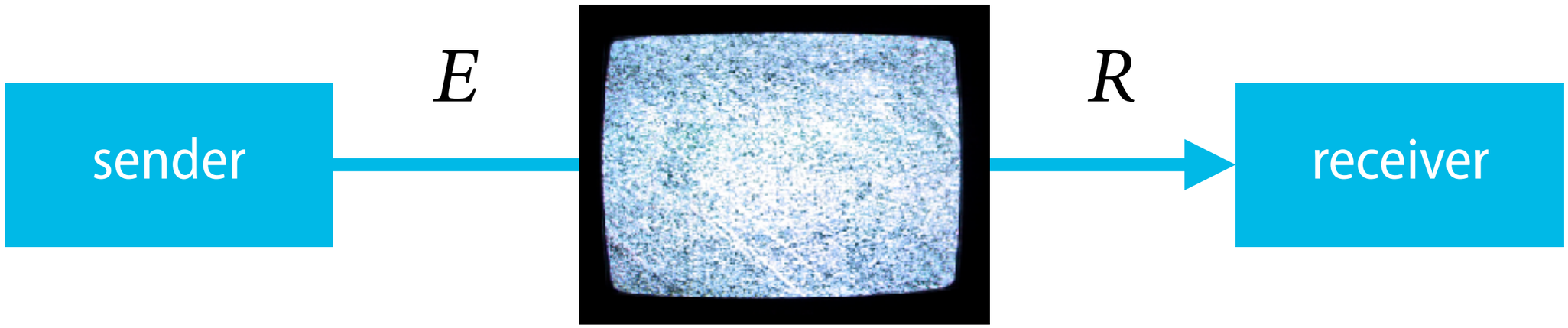
  alternative view: compute the perplexity between two language models

- The most successful systems for language identification use character-based language models.

‘When I look at an article in Russian, I say:
“This is really written in English,
but it has been coded in some strange symbols.
I will now proceed to decode.”’

Warren Weaver (1894–1978)

Language models in machine translation

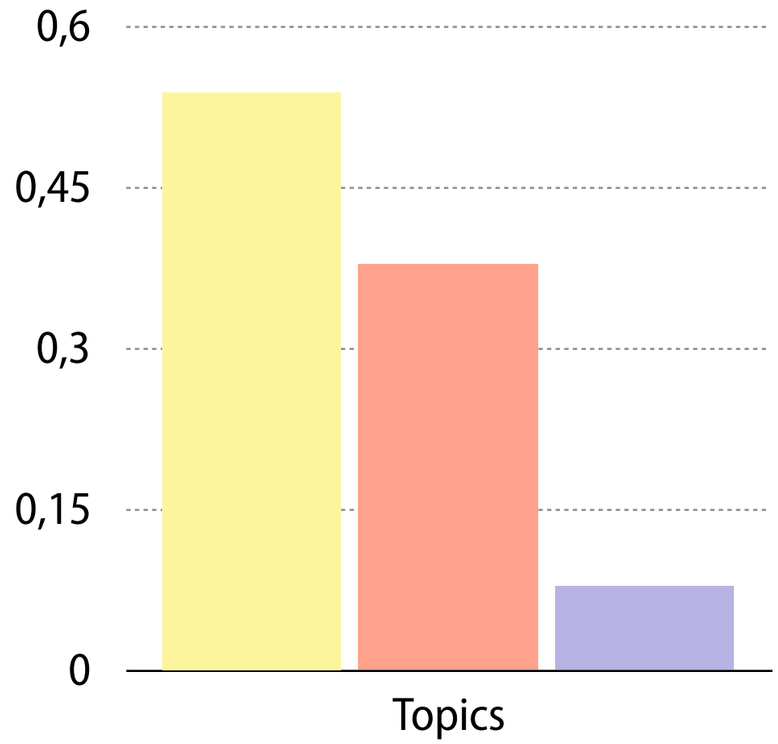


‘Russian is noisy English.’

$$\text{argmax}_E \, P(E|R) = \text{argmax}_E \, P(R|E) \, P(E)$$

Topic models are language models

How many genes does an organism need to survive? Last week at the genome meeting here, two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes.



Adapted from Blei (2012)

Topic models are language models

human	evolution	computer
genome	evolutionary	models
dna	species	information
genetic	organisms	data
genes	life	computers
sequence	origin	system
gene	biology	network
molecular	groups	systems
sequencing	phylogenetic	model
map	living	parallel
information	diversity	methods
genetics	group	networks
mapping	new	software
project	two	new
sequences	common	simulations

Structure of this lecture

- Unigram models
- General n-gram models
- Evaluation of n-gram models

Unigram models

Unigram model

A **unigram language model** views a document *d* as a 'bag of words':

document

count of *w* in *d*

$$P(d = w_1 \cdots w_N) = \prod_{i=1}^{N} P(w_i) = \prod_{w \in V} P(w)^{c(w,d)}$$

vocabulary

The bag of words

the gorgeously elaborate continuation of the lord of the rings trilogy is so huge that a column of words cannot adequately describe co-writer/director peter jackson's expanded vision of j.r.r. tolkien's middle-earth

positive

… is a sour little movie at its core an exploration of the emptiness that underlay the relentless gaiety of the 1920's as if to stop would hasten the economic and global political turmoil that was to come

negative

The bag of words

a adequately cannot
co-writer/director column
continuation describe
elaborate expanded gorgeously
huge is j.r.r. jackson lord
middle-earth of of of of peter
rings so that the the the tolkien
trilogy vision words

positive

… 1920's a an and as at come
core economic emptiness
exploration gaiety global
hasten if is its little movie of of
political relentless sour stop
that that the the the the to to
turmoil underlay was would

negative

The bag of words

word	count
of	4
the	3
words	1
vision	1
trilogy	1
…	

positive

word	count
the	4
to	2
that	2
of	2
would	1
…	

negative

Estimation of unigram models

- The standard method for estimating unigram models is **maximum likelihood estimation**.

  maximise the likelihood of the observations given the parameters

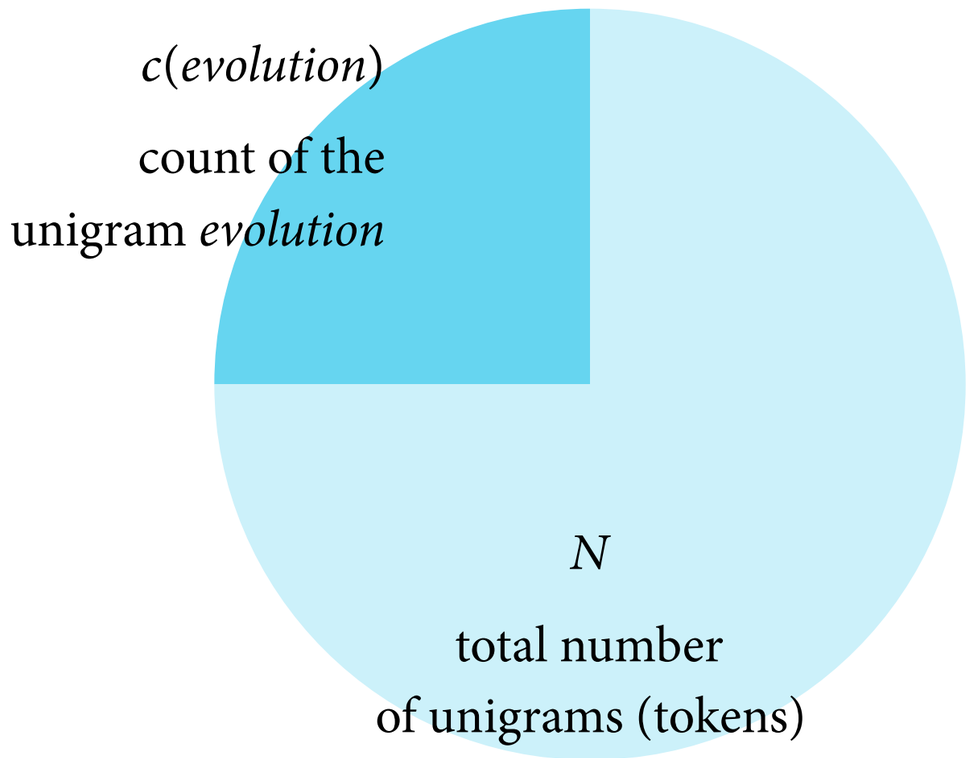
- We want to find model parameters (word probabilities) that maximise the likelihood of some text data and sum up to one.

- It turns out that we can solve this problem by simply counting word occurrences and normalising over the data.

  formal derivation uses Laplace multipliers

MLE of unigram probabilities

$$P(evolution)$$



$c(evolution)$

count of the
unigram *evolution*

N

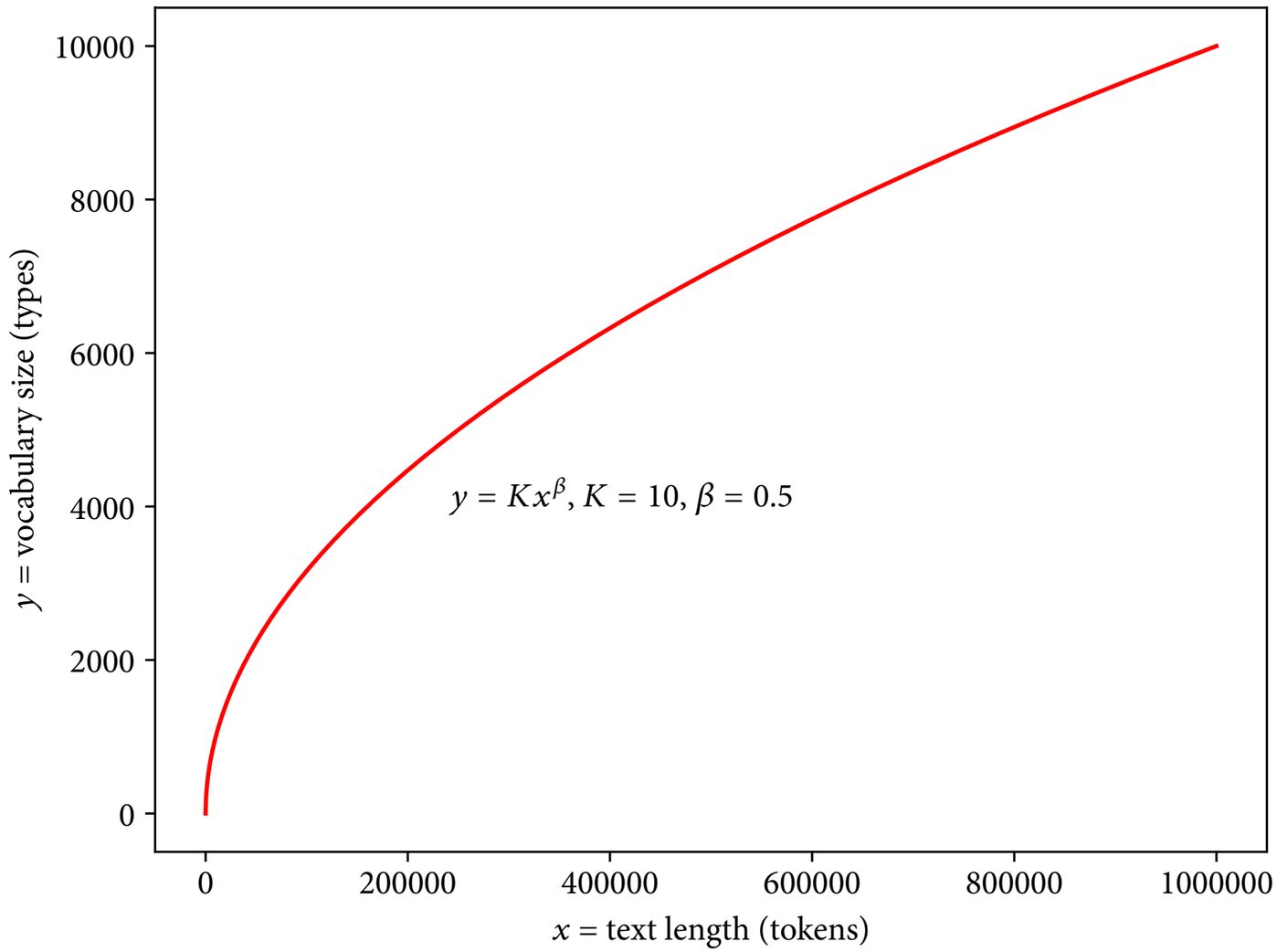
total number
of unigrams (tokens)

The problem of unseen words

- Even with very large text collections, there will always be words in the vocabulary that we never observe.
- When we later compute the probability of a document that contains such **unseen words**, that probability will be zero.

  Slogan: Zero probabilities destroy information!

Heaps' law


10000
8000
6000
4000
2000
0
0
200000
400000
600000
800000
1000000
y = vocabulary size (types)
x = text length (tokens)
$y = Kx^{\beta}, K = 10, \beta = 0.5$

Smoothing

- In the context of language models, the term **smoothing** refers to techniques aimed at making a model more robust.

- Intuitively, we 'spread the probability mass' more evenly over the words in the vocabulary than MLE would do.

- Which smoothing techniques work best for a given application is an empirical question.

Unknown words

- In addition to unseen words, a new text may even contain **unknown words**. For these, smoothing will not help.

  out-of-vocabulary

- One way to deal with this is to introduce a special word type UNK, and to smooth it like any other word type in the vocabulary.

- When we compute the probability of a document, then we first replace every unknown word with UNK.

Notation

N	number of word tokens
$c(w)$	count of unigram (word) w
V	number of word types
V_k	number of word types seen exactly k times
V_{k+}	number of word types seen at least k times

Source: Chen and Goodman (1998)

Laplace smoothing

- In **Laplace smoothing**, also called Lidstone smoothing, we hallucinate k extra occurrences of every word type:

$$P(w) = \frac{c(w) + k}{N + kV}$$

- The parameter $k \geq 0$ is the smoothing parameter. In the case where $k = 1$, this is called **add-one smoothing**.

  Note that k does not need to be an integer.

Rewriting the formula for additive smoothing

The formula for add-k smoothing of unigrams,

$$P(w) = \frac{c(w) + k}{N + kV}$$

can be written as a linear interpolation of the maximum-likelihood estimate and the uniform distribution over word types:

$$P(w) = \lambda \frac{c(w)}{N} + (1 - \lambda)\frac{1}{V} \qquad \text{where} \qquad \lambda = \frac{N}{N + kV}$$

A problem with additive smoothing

We count the number of occurrences of the word *the* in a sample of 55,708,861 words of English.

Without smoothing		With add-one smoothing	
Observed count	Probability estimate	Probability estimate	Expected count
3,579,493	0.0643	0.0641	3,570,938

The smoothed model has no explanation for the 8,555 extra occurrences in the sample.

Chiang (2016)

A problem with additive smoothing

- We have only a constant amount of probability mass that we can distribute among the word types.
- Therefore, although we are adding to the count of every word type, we are not adding to the probability of every word type.

  probabilities still need to sum to one

- We take away a certain percentage of the probability mass from each word type and redistribute it equally to all word types.

Witten–Bell smoothing

Writing V_{1+} for the number of word types seen at least once,

$$P(w) = \lambda \frac{c(w)}{N} + (1 - \lambda)\frac{1}{V} \qquad \text{where} \qquad \lambda = \frac{N}{N + V_{1+}}$$

gives us **Witten–Bell smoothing**. This can be viewed as an adaptive form of additive smoothing and can also be written as

$$P(w) = \frac{c(w) + k}{N + kV} \qquad \text{where} \qquad k = \frac{V_{1+}}{V}$$

Good–Turing estimation, intuition

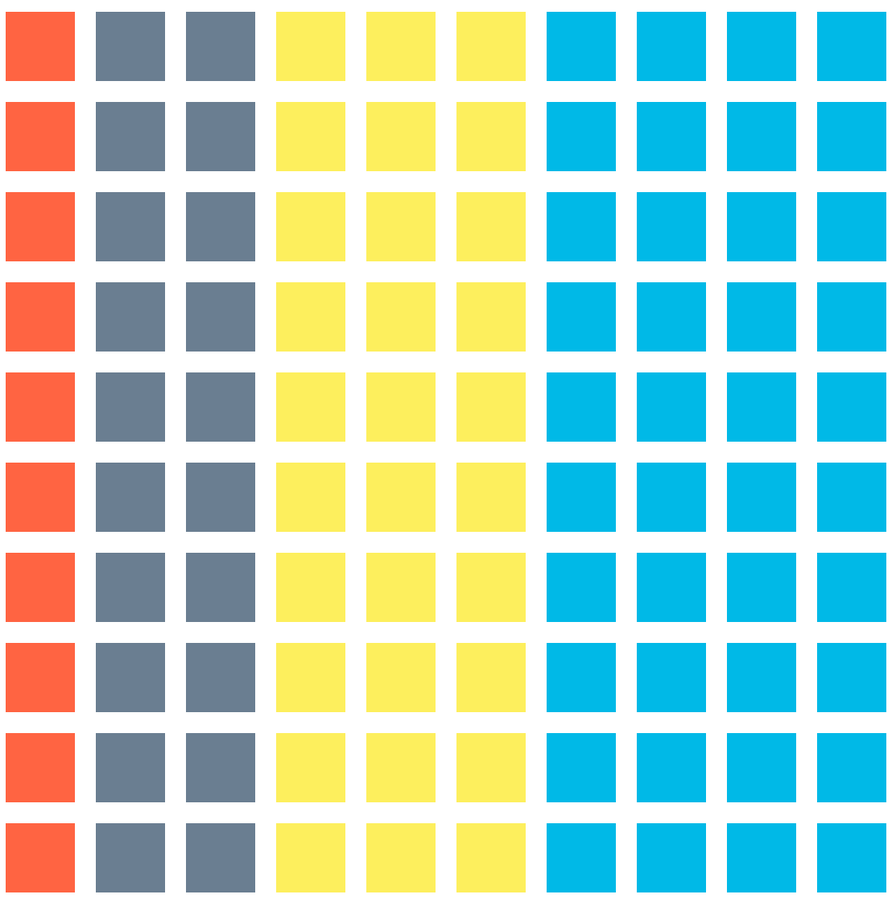
Derive probabilities for unseen words from the counts of words that we have seen only once.

V_1 = number of word types seen exactly once

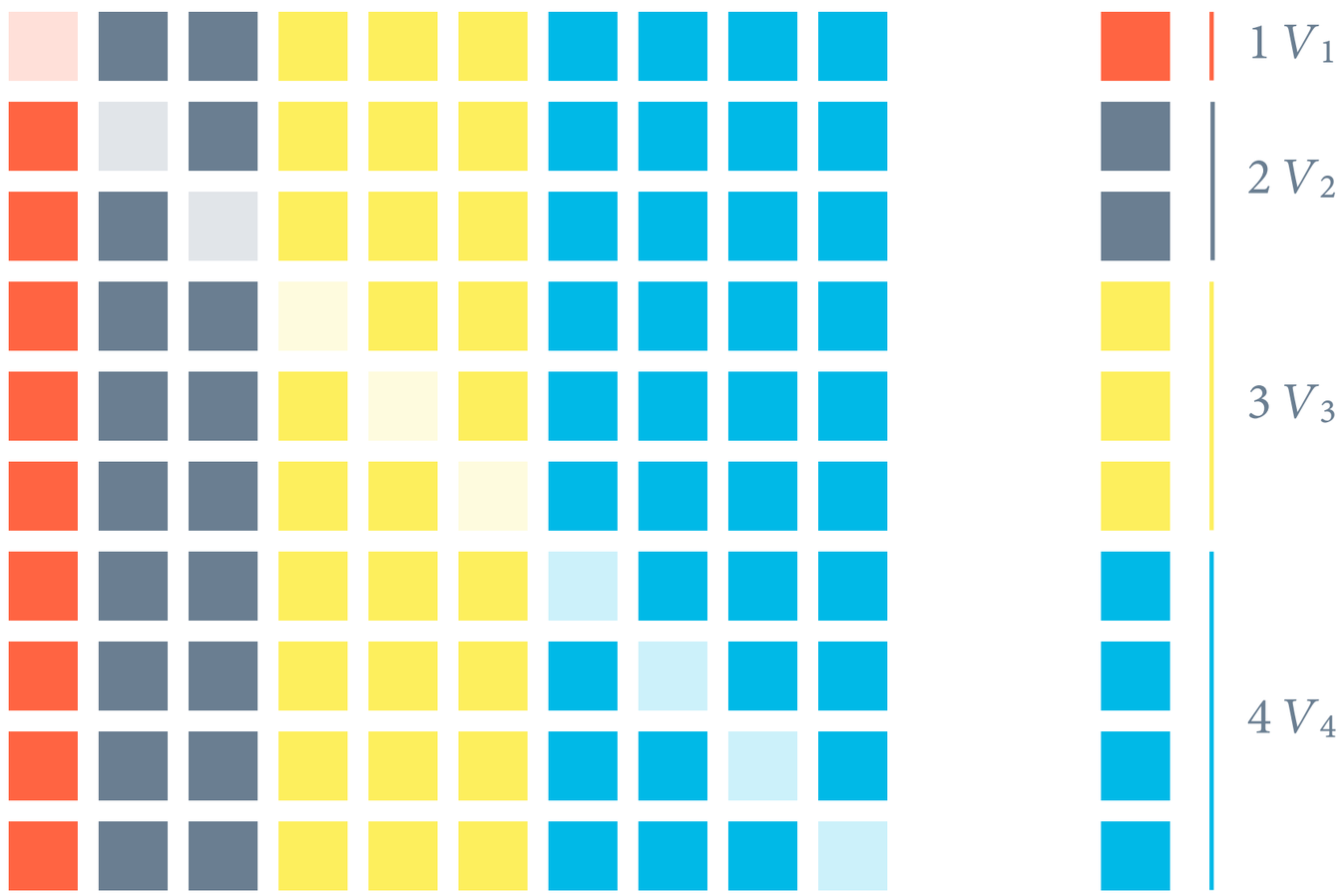
How likely is it that the next word is ■? 3/18 $V_1 = 3$

Good–Turing estimation, intuition

Good–Turing estimation, intuition



Good–Turing estimation, intuition



Good–Turing estimation

- In the sample, how often do we expect to see a token w that we have seen k times in the modified training set?

$$c^*(w) = V_1 \quad (k = 0) \qquad\qquad c^*(w) = \frac{(k+1)V_{k+1}}{V_k} \quad (k > 0)$$

- Since sample size is N, we get the following probability estimates:

$$P(w) = \frac{V_1}{N} \quad (k = 0) \qquad\qquad P(w) = \frac{(k+1)V_{k+1}}{NV_k} \quad (k > 0)$$

From Good–Turing to absolute discounting

Empirical count	Good–Turing count
0	0.0000270
1	0.45
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Source: Church and Gale (1991)

Absolute discounting

- We would like to decrease the expected count of a word (relative to its empirical count) by a constant value.
- In **absolute discounting**, we subtract $0 < d < 1$ from the count of every observed word type and re-distribute equally to all types:

$$P(w) = \frac{\max(0, c(w) - d)}{N} + \frac{dV_{1+}}{N}\frac{1}{V}$$

General n-gram models

N-gram models

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

  unigram, bigram, trigram, quadrigram

- An ***n*-gram model** is a language model where the probability of a word depends only on the $n - 1$ immediately preceding words.

Markov models

- The probability of each item depends only on the immediately preceding item.
- The probability of a sequence of items is the product of these conditional probabilities.
- For a well-defined model, we need to mark the beginning and the end of a sequence.

Андре́й Ма́рков
(1856–1922)

Image source: Wikipedia

Probability of a sequence of words

beginning-of-sentence

$$P(w_1 w_2 w_3) = P(w_1 \mid \text{BOS}) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_2) \cdot P(\text{EOS} \mid w_3)$$

end-of-sentence

Bigram models

A **bigram model** is a Markov model on sequences of words:

$$P(w_1 \cdots w_N) = P(w_1 \,|\, \text{BOS}) \cdot \prod_{i=2}^{N} \left[P(w_i \,|\, w_{i-1}) \right] \cdot P(\text{EOS} \,|\, w_N)$$

Thus the probability of a word depends only on the immediately preceding word.

Formal definition of an n-gram model

n	the model's order (1 = unigram, 2 = bigram, …)
V	a set of possible words (character); 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

Simple uses of n-gram models

- **Prediction**

  To predict the next word, we can choose the word that has the highest probability among all possible words w:

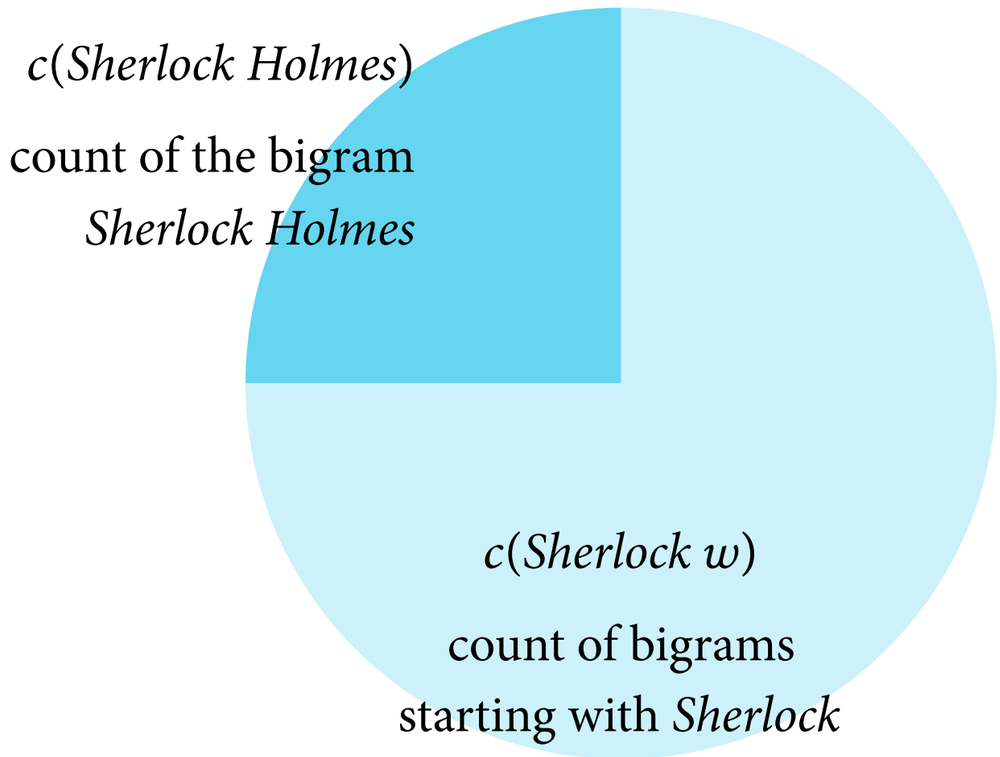
  $$\text{predicted word} = \text{argmax}_w \, P(w | \text{preceding words})$$

- **Generation**

  We can generate a random sequence of words where each word w is sampled with probability $P(w | \text{preceding words})$.

Estimating bigram probabilities

$$P(Holmes|Sherlock)$$



Smoothing bigram probabilities

- When smoothing unigram probabilities, we 'took away' from the ML estimate and 'gave back' equally to all word types.
- For bigram probabilities, a better option is to give back to word types proportional to their unigram probability.
- Moreover, we will make the amount that we take away from the ML estimate dependent on the context.

Reminder: Witten–Bell smoothing for unigrams

Writing V_{1+} for the number of word types seen at least once,

$$P(w) = \lambda \frac{c(w)}{N} + (1 - \lambda)\frac{1}{V} \qquad \text{where} \qquad \lambda = \frac{N}{N + V_{1+}}$$

gives us **Witten–Bell smoothing**. This can be viewed as an adaptive form of additive smoothing and can also be written as

$$P(w) = \frac{c(w) + k}{N + kV} \qquad \text{where} \qquad k = \frac{V_{1+}}{V}$$

Smoothing bigram probabilities

Witten–Bell smoothing

$$P(w \mid u) = \lambda(u)\frac{c(uw)}{c(u\bullet)} + \big(1 - \lambda(u)\big)P(w) \qquad \lambda(u) = \frac{c(u\bullet)}{c(u\bullet) + V_{1+}(u\bullet)}$$

Absolute discounting

$$P(w \mid u) = \frac{\max\big(0, c(uw) - d\big)}{c(u\bullet)} + \frac{dV_{1+}(u\bullet)}{c(u\bullet)}P(w)$$

Kneser–Ney smoothing

- **Kneser–Ney smoothing** is a refinement of absolute discounting that uses better estimates of the lower-order *n*-grams.
- Some unigrams are frequent only in the context of some bigrams, and we should have limited trust in them in other contexts.

  I can't find my reading *glasses*. I can't find my reading *Francisco*.

- When backing off to a lower-order *n*-gram, we should use a special 'continuation probability' for that *n*-gram.

Adapted from Dan Jurafsky

Continuation probabilities

- The **continuation probability** for a unigram w should be proportional to the number of bigram contexts it completes:

$$P'(w) \propto \left|\{ u \mid c(uw) > 0 \}\right|$$

- To get a probability, we normalise these counts by the total number of bigram types.

- With this, a frequent word that only occurs in one context will have a small continuation probability.

Evaluation of n-gram models

Intrinsic and extrinsic evaluation

- **Intrinsic evaluation**

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

  in classification: accuracy, precision, recall

- **Extrinsic evaluation**

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

  predictive input, machine translation, speech recognition

Perplexity and entropy

The **perplexity** of a language model P on a test sample $x_1, \ldots x_N$ is

$$2^{-\frac{1}{N}\log_2 P(x_1,\ldots,x_N)}$$

This measure is not easy to understand intuitively.
We will therefore focus on the term in the exponent:

$$-\frac{1}{N}\log_2 P(x_1, \ldots, x_N)$$

This measure is known as the **entropy** of the test sample.

Intrinsic evaluation of language models, intuition

- Learn a language model from a set of training sentences and use it to compute the probability of a set of test sentences.

  Fundamental assumption: Both models are defined on the same vocabulary.

- If the language model is good, the probability of the test sentences should be high.

- In the following, for simplicity we assume that we have only one, potentially very long test sentence.

The problem with different sentence lengths

- Under a Markov model, the probability of a sentence is the product of the bigram probabilities.
- Therefore, all other things being equal, the probability of a sentence decreases with the sentence length.
- This makes it hard to compare the probability of the test data to the probability of the training data.

  Intuitively, we would like to average over sentence length.

From probabilities to surprisal

- Instead of computing probabilities for the test sentence, we will compute negative log probabilities.

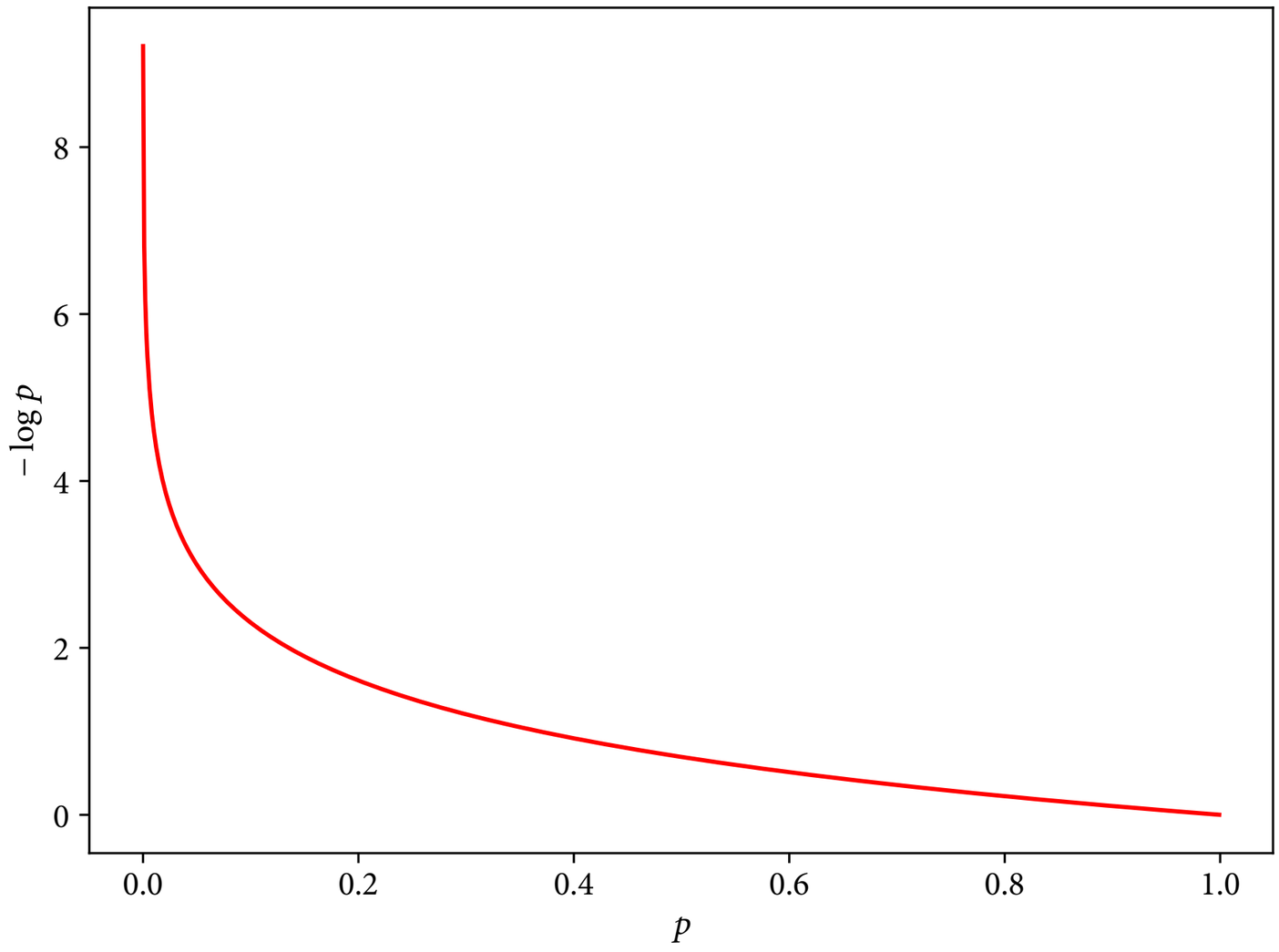
  $P(w|c)$ becomes $-\log P(w|c)$

- Intuitively, this measures how 'surprised' we are about seeing the test sentence, given our language model.

  high probability = low surprisal

- We can then simply divide by the number of words in the sentence to average over sentence length.

Negative log probabilities



Entropy is cross-entropy

- Formally, we compute the cross-entropy between two probability distributions: a language model and the empirical measure (a text).
- We view a language as a stochastic process that generates a sequence of tokens, each of which is a random variable.

  transcription of every word you utter in your life, universal corpus

- Under this view and certain assumptions, per-word cross-entropy is well approximated by average negative log probability.

  Shannon–McMillan–Breiman theorem (Asymptotic Equipartition Property)

From cross-entropy to model entropy

$$H(L,m) = -\lim_{|x|\to\infty} \frac{1}{|x|} \sum_{x} p_L(x) \log_2 m(x)$$

Asymptotic Equipartition Property

$$H(L,m) = -\lim_{|x|\to\infty} \frac{1}{|x|} \log_2 m(x)$$

pick a single, long enough sequence x

$$H(L,m) \approx -\frac{1}{|x|} \log_2 m(x)$$

Structure of this lecture

- Unigram models
- General n-gram models
- Evaluation of n-gram models