LANGUAGE MODELS

Gabriella Pasi
pasi@disco.unimib.it

Based on material of
Dan Jurafsky and Chris Manning

Language Models

- A *text* can be represented as a *language model* to represent its topics
  - words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model
- A LM model assigns probabilities to sequences of words
  - p("*Today is Wednesday*")
  - p("*Today Wednesday is*")
- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

Language models: why ?

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)

Language models: why ?

- Text categorization
  - Given that we observe “baseball” three times and “game” once in a news article, how likely is it about “sports” v.s. “politics”?

- Information retrieval
  - Given that a document is centered on the topic of sport, how likely would a query “generated” by this document?

+ Summarization, question-answering, etc., etc.!!

Language Models

- **Goal**: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

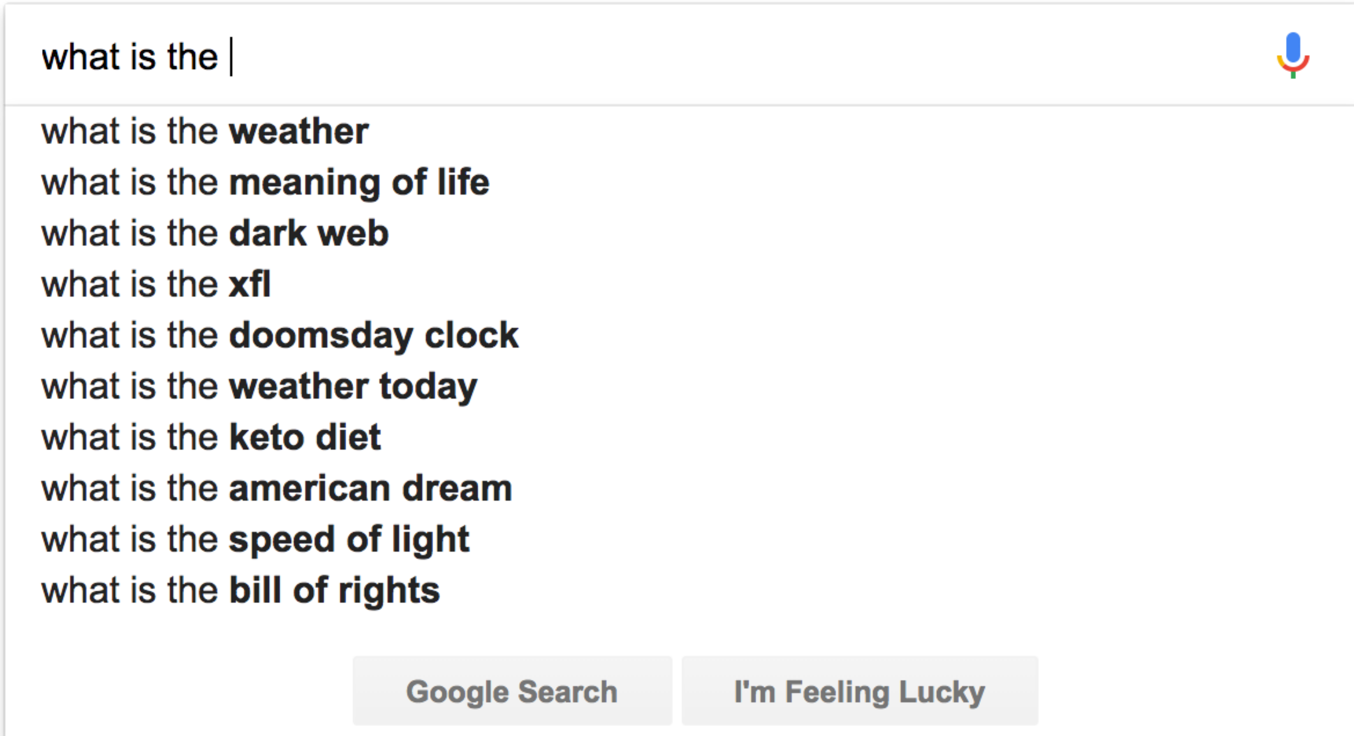
- **Related task**: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- So, a model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

You use Language Models every day!



Notation

- To represent the probability of a particular random variable X_i taking on the value “the”, or $P(X_i =$ “the”$)$, we will use the simplified notation $P(the)$.
- a sequence of n words is denoted either as $w_1 \ldots w_n$ or as w_1^n
- the joint probability of each word in a sequence having a particular value: $P(X = w_1, Y = w_2, Z = w_3, \ldots, W = w_n)$ is denoted as $P(w_1, w_2, \ldots, w_n)$.

How to compute $P(w_n|w_1,w_2 \ldots w_{n-1})$?

- Let us start by computing $P(w_n|w_1,w_2 \ldots w_{n-1})$, the probability of a word w_n given a sequence of words.
- For example: *P*(*the*|*its water is so transparent that*)
- Relative frequency counts: given a ***very large corpus***, count the number of times we see *its water is so transparent that*, and count the number of times it is followed by *the*.

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

Too many possible sentences!

How to compute P(W) ?

- Similarly, if we aim to know the probability P(W) of a sentence W (i.e., the joint probability of an entire sequence of words like *its water is so transparent)*, we could do it by asking "out of all possible sequences of five words, how many of them are *its water is so transparent*?"
- To do so, we would have to get the count of *its water is so transparent* and divide by the sum of the counts of all possible five word sequences.
- *That seems rather a lot to estimate*!!!

How to compute P(W) *practically*

- For example, how to compute this joint probability:

  - P(its, water, is, so, transparent, that)

- Intuition: let’s rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

  $p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**

- More variables:

  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

So, we can compute a joint probability by multiplying a number of conditional probabilities but … this seems not help! However…… we can approximate the "history"

Markov Assumption

Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

N-grams Language Models

- *Unigram language model*
  - probability distribution over the words in a language
  - generation of text consists of pulling words out of a “bucket” according to the probability distribution and replacing them
  - *PROBABILITIES OF WORDS IN A SEQUENCE DO NOT DEPEND ON PREVIOUS WORDS*
- N-gram language model
  - some applications use bigram and trigram language models where probabilities depend on previous words
  - BIGRAM LM: the probability of a word in a sequence depend on the word that preceeds it
  - TRIGRAM LM: the probability of a word in a sequence depend on the two worda that preceed it

N-grams Language Models

- Example of a 4gram LM (prediction based on the previous three words)

~~*as the proctor started the clock, the*~~ *students opened their* ______

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count(students opened their } \boldsymbol{w})}{\text{count(students opened their)}}$$

<u>For example, suppose that in the corpus:</u>

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Sparsity Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if *"students opened their $\boldsymbol{w}$"* never occurred in data? Then $\boldsymbol{w}$ has probability 0!

(Partial) Solution: Add small δ to the count for every $\boldsymbol{w} \in V$. This is called *smoothing*.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* $\boldsymbol{w}$!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: Increasing *n* makes sparsity problems *worse.* Typically we can't have *n* bigger than 5.

Recap: Language Models

A language model is **well-formed** over alphabet Σ if $\sum_{s \in \Sigma^*} P(s) = 1$.

Generic Language Model

"Today is Tuesday"	0.01
"The Eigenvalue is positive"	0.001
"Today Wednesday is"	0.00001
…	

Unigram Language Model

"Today"	0.1
"is"	0.3
"Tuesday"	0.2
"Wednesday"	0.2
…	

Bigram Language Model

"Today"	0.1
"is" | "Today"	0.4
"Tuesday" | "is"	0.8
…	

How to handle sequences?

- Chain Rule (requires long chains of cond. prob.):

$$P(t_1 t_2 t_3 t_4) = P(t_1) P(t_2 | t_1) P(t_3 | t_1 t_2) P(t_4 | t_1 t_2 t_3)$$

- Bigram LM (pairwise cond. prob.):

$$P_{bi}(t_1 t_2 t_3 t_4) = P(t_1) P(t_2 | t_1) P(t_3 | t_2) P(t_4 | t_3)$$

- Unigram LM (no cond. prob.):

$$P_{uni}(t_1 t_2 t_3 t_4) = P(t_1) P(t_2) P(t_3) P(t_4)$$

Recap: language models

How do we build probabilities over sequence of terms?

P(t1, t2, t3, t4) = P(t1) x P(t2|t1) x P(t3|t1, t2) x P(t4|t1, t2, t3)

Unigram language model –simplest ; no conditioning context

P(t1, t2, t3, t4) = P(t1) x P(t2) x P(t3) x P(t4)

Bigram language model – condition on previous term

P(t1, t2, t3, t4) = P(t1) x P(t2|t1) x P(t3|t2) x P(t4|t3)

Trigram language model …

Unigram model is the most common in IR

- Often sufficient to judge the topic of a document
- Data sparseness issues when using richer models
- Simple and efficient implementation

N-gram models

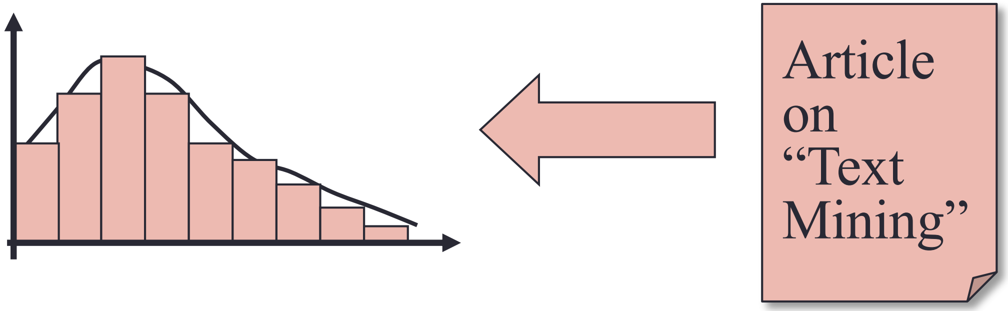
- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

    "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

Text representation with unigram LM

LM for topic 1: *IR&DM*

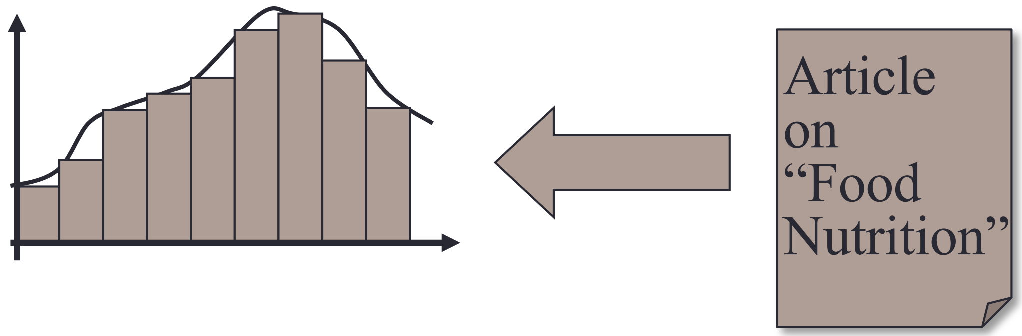
text	0.2
mining	0.1
n-gram	0.01
cluster	0.02
…	
healthy	0.000001
…	



different θ_d for different d

LM for topic 2: *Health*

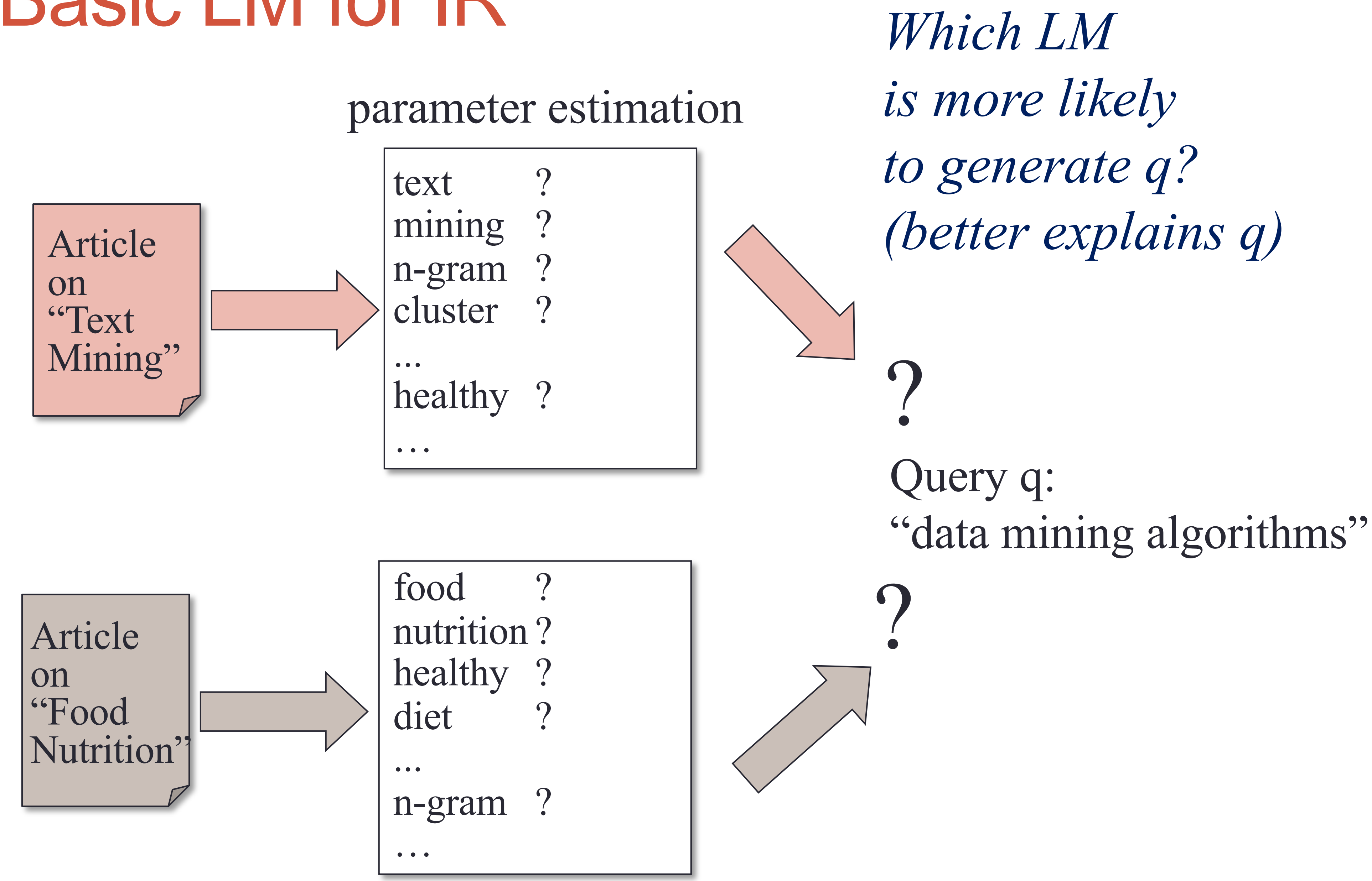
food	0.25
nutrition	0.1
healthy	0.05
diet	0.02
…	
n-gram	0.00002
…	



LMs for Retrieval

- 3 possibilities:
  - probability of generating the query text from a document language model
  - probability of generating the document text from a query language model
  - comparing the language models representing the query and document topics
- We will see this when will will present IR models

Basic LM for IR



Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Estimating Probabilities

- Obvious estimate for unigram probabilities is

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

- *Maximum likelihood estimate*
  - makes the observed value of $f_{q_i;D}$ most likely
- If query words are missing from document, score will be zero
  - Missing 1 out of 4 query words same as missing 3 out of 4

Smoothing

- Document texts are a *sample* from the language model
  - Missing words should not have zero probability of occurring
- *Smoothing* is a technique for estimating probabilities for missing (or unseen) words
  - lower (or *discount*) the probability estimates for words that are seen in the document text
  - assign that “left-over” probability to the estimates for the words that are not seen in the text

Neural Language Models

- To overcome some limitations of Statistical LM, neural LM have been definied:
  - Fixed window neural LM
  - RNN (recurrent NN) LM
  - BERT (Bidirectional Encoder Representations from Transformers)
  - BERT's variants
  - ….

Evaluation: How good is our model?

- Does our language model "prefer" good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences than those sentences that "rarely observed" or "ungrammatical" ?
- 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 that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.

(Extra Slide not in video)
Training on the test set

- We can't allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- "Training on the test set"
- Bad science!
- And violates the honor code

Extrinsic evaluation of N-gram models

- Best evaluation for comparing two language models A and B
  - Put each model in a specific NLP task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
    - Time-consuming; can take days or weeks
- So
    - Sometimes use **intrinsic** evaluation: **perplexity**
    - Bad approximation
        - unless the test data looks **just** like the training data
        - So **generally only useful in pilot experiments**
    - But is helpful to think about.

Perplexity

The best language model is one that best predicts unseen words in a test set

- Gives the highest P(sentence)

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

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

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

Chain rule:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Lower perplexity = better model

Example Perplexity Values of different N-gram language models trained using 38 million words and tested using 1.5 million words from *The Wall Street Journal dataset*

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109