COMP4901K/Math4824B
Machine Learning for Natural Language Processing

Lecture 12: Language Models

Instructor: Yangqiu Song

Slides credits: Hongning Wang

Today's lecture

1. How to represent a document?
   - Make it computable
2. How to infer the relationship among documents or identify the structure within a document?
   - Knowledge discovery

What is a statistical LM?

- A model specifying probability distribution over <u>word sequences</u>
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001

- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

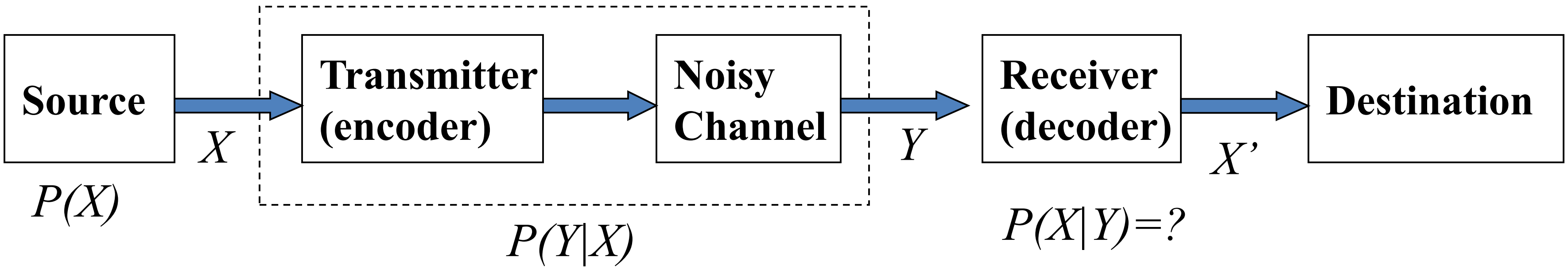
Why is a LM useful?

- Provide a principled way to quantify the uncertainties associated with natural language
- Allow us to answer questions like:
  - Given that we see "*John*" and "*feels*", how likely will we see "*happy*" as opposed to "*habit*" as the next word?
    (speech recognition)
  - Given that we observe "baseball" three times and "game" once in a news article, how likely is it about "sports" v.s. "politics"?
    (text categorization)
  - Given that a user is interested in sports news, how likely would the user use "baseball" in a query?
    (information retrieval)

Measure the fluency of documents

- How likely this document is generated by a given language model
  - If $p_{text-mining}(d) > p_{health}(d)$, d belongs to text mining related documents
  - If $p_{user_a}(d_1) > p_{user_a}(d_2)$, recommend d_1 to $user_a$

Source-Channel framework [Shannon 48]



$$\hat{X} = \arg\max_X \; p(X \mid Y) = \arg\max_X \; p(Y \mid X)\, p(X)$$ **(Bayes Rule)**

When X is text, p(X) is a language model

Many Examples:

Speech recognition:	**X=Word sequence**	**Y=Speech signal**
Machine translation:	**X=English sentence**	**Y=Chinese sentence**
OCR Error Correction:	**X=Correct word**	**Y= Erroneous word**
Information Retrieval:	**X=Document**	**Y=Query**
Summarization:	**X=Summary**	**Y=Document**

Language Model for Text

- Goal: Assign useful probabilities P(X) to sentences/ documents X
  - Input: many observations of training sentences X
  - Output: system capable of computing P(X)

- Probabilities should broadly indicate plausibility of sentences
  - P(I saw a van) $\gg$ P(eyes awe of an)
  - Not grammaticality: P(artichokes intimidate zippers) ≈ 0
  - In principle, "plausible" depends on the domain, context, speaker...

Language model for text

We need independence assumptions!

- Probability distribution over word sequences
  - $p(w_1\, w_2\, \ldots\, w_n) =$
    $p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \ldots p(w_n|w_1, w_2, \ldots, w_{n-1})$
  - Complexity - $O(V^{n^*})$
    - n^* - maximum ~~document~~ length
      sentence

Chain rule: from conditional probability to joint probability

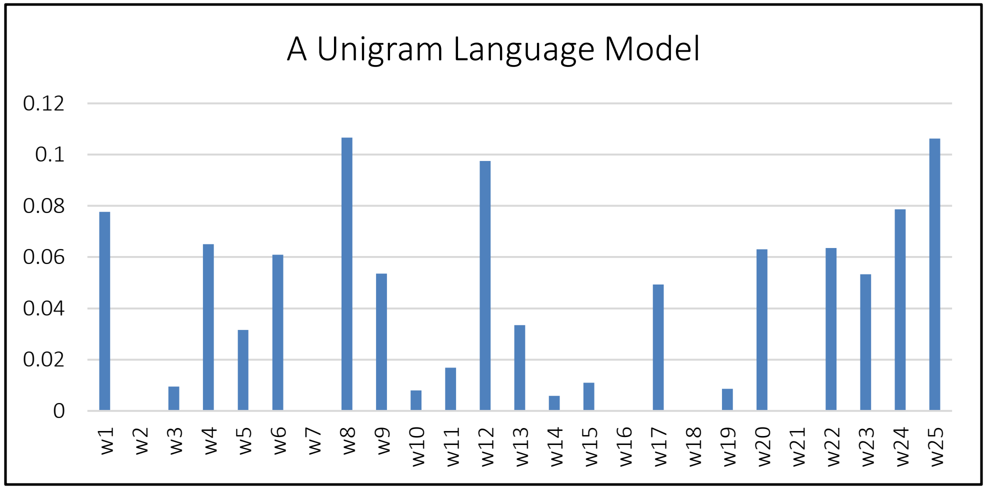
- 475,000 main headwords in Webster's Third New International Dictionary
- Average English sentence length is 14.3 words
- A rough estimate: $O(475000^{14})$

How large is this? $\approx 3.38e^{66}TB$

Unigram language model

- Generate a piece of text by generating each word independently
  - $p(w_1\, w_2\ \dots\ w_n) = p(w_1)p(w_2)\dots p(w_n)$
  - Looks familiar?
- Essentially a multinomial distribution over the vocabulary

The simplest and most popular choice!



More sophisticated LMs

- N-gram language models
  - Conditioned only on the past n-1 words
  - E.g., bigram: $p(w_1 \ \ldots \ w_n) =$
    $p(w_1)p(w_2|w_1)\,p(w_3|w_2) \ \ldots p(w_n|w_{n-1})$
- Such independence assumptions are called Markov assumptions (of order n-1)

$$p(w_n|w_1, w_2, \ldots, w_{n-1}) = p(w_n|w_{i-n+1}, \ldots, w_{n-1})$$

Why just unigram models?

- Difficulty in moving toward more complex models
  - They involve more parameters, so need more data to estimate
  - They increase the computational complexity significantly, both in time and space
- Capturing word order or structure may not add so much value for "topical inference"
- But, using more sophisticated models can still be expected to improve performance ...

Generative view of text documents

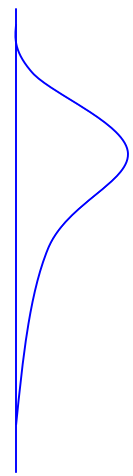
(Unigram) Language Model θ
p(w| θ)

Sampling

Document

Topic 1:
Text mining

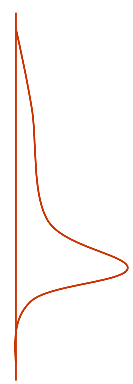
...
text 0.2
mining 0.1
assocation 0.01
clustering 0.02
...
food 0.00001
...



Text mining document

Topic 2:
Health

...
text 0.01
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
...

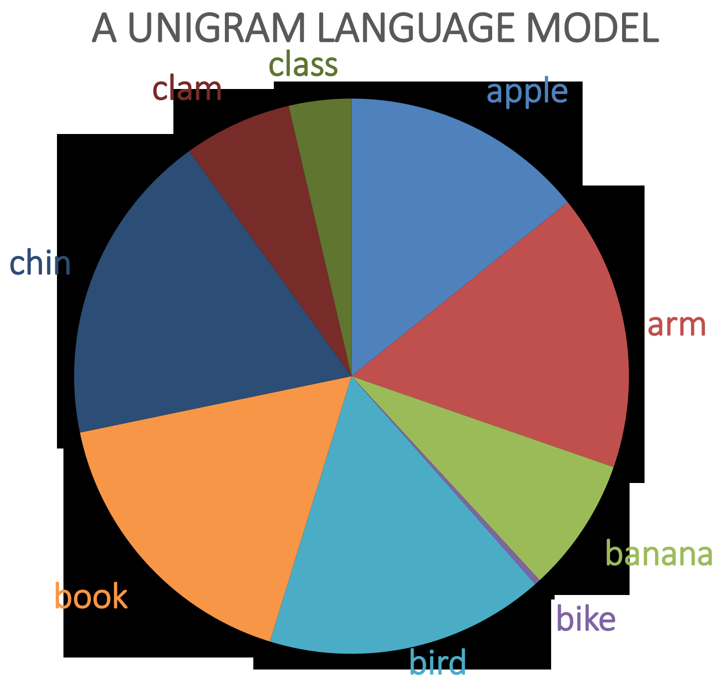


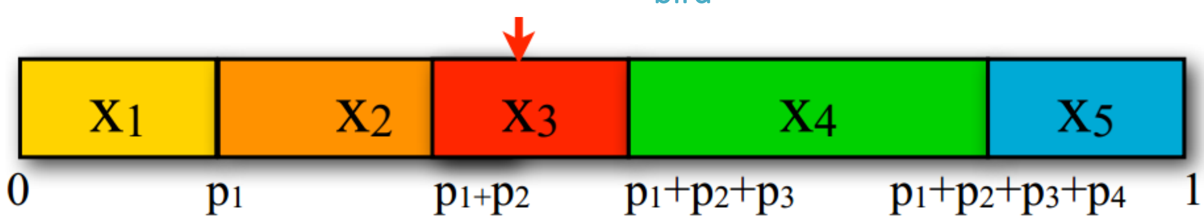
Food nutrition document

How to generate text from an N-gram language model?

- Sample from a discrete distribution $p(X)$
  - Assume n out

1. Divide the i ding to the probabilitie
2. Generate a l 1
3. Return x_i w





Generating text from language models

P(of) = 3/66
P(Alice) = 2/66
P(was) = 2/66
P(to) = 2/66

P(her) = 2/66
P(sister) = 2/66
P(,) = 4/66
P(') = 4/66

Under a unigram language model:

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

Generating text from language models

P(of) = 3/66
P(Alice) = 2/66
P(was) = 2/66
P(to) = 2/66

P(her) = 2/66
P(sister) = 2/66
P(,) = 4/66
P(') = 4/66

Under a unigram language model:

The same likelihood!

beginning by, very Alice but was and?
reading no tired of to into sitting
sister the, bank, and thought of without
her nothing: having conversations Alice
once do or on she it get the book her had
peeped was conversation it pictures or
sister in, 'what is the use had twice of
a book''pictures or' to

N-gram language models will help

- Unigram

  Generated from language models of New York Times

  – Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a q acquire to six executives.
- Bigram
  – Last December through the way to preserve the Hudson corporation N.B.E.C. Taylor would seem to complete the major central planners one point five percent of U.S.E. has already told M.X. corporation of living on information such as more frequently fishing to keep her.
- Trigram
  – They also point to ninety nine point six billon dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

Turing test: generating Shakespeare

A	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Every enter now severally so, let • Hill he late speaks; or! a more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
B	• What means, sir. I confess she? then all sorts, he is trim, captain. •Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. •What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the firs •E SCIgen - An Automatic CS Paper Generator com- mand of fear not a liberal largess given away, Falstaff! Exeunt
C	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
D	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry, 'tis a noble Lepidus.

Estimation of language models

Unigram Language Model θ

p(w| θ)=?

…
text ?
mining ?
assocation ?
database ?
…
query ?
…

Estimation

Document

text 10
mining 5
association 3
database 3
algorithm 2
…
query 1
efficient 1

A “text mining” paper
(total #words=100)

Estimation of language models

- Maximum likelihood estimation

~~Unigram~~ Language Model θ

$p(w|\theta)=?$

Estimation

Document

...
10/100 → text ?
5/100 → mining ?
3/100 → assocation ?
3/100 → database ?
...
1/100 → query ?
...

text 10
mining 5
association 3
database 3
algorithm 2
...
query 1
efficient 1

A "text mining" paper
(total #words=100)

Parameter estimation

- General setting:
  - Given a (hypothesized & probabilistic) model that governs the random experiment
  - The model gives a probability of any data $p(X|\theta)$ that depends on the parameter θ
  - Now, given actual sample data X={x_1,…,x_n}, what can we say about the value of θ?
- Intuitively, take our best guess of θ -- "best" means "best explaining/fitting the data"
- Generally an optimization problem

Maximum likelihood vs. Bayesian

- Maximum likelihood estimation
  - "Best" means "data likelihood reaches maximum"

$$\widehat{\boldsymbol{\theta}} = \mathbf{argmax}_{\boldsymbol{\theta}} \mathbf{P}(\mathbf{X}|\boldsymbol{\theta})$$

  - Issue: small sample size

ML: Frequentist's point of view

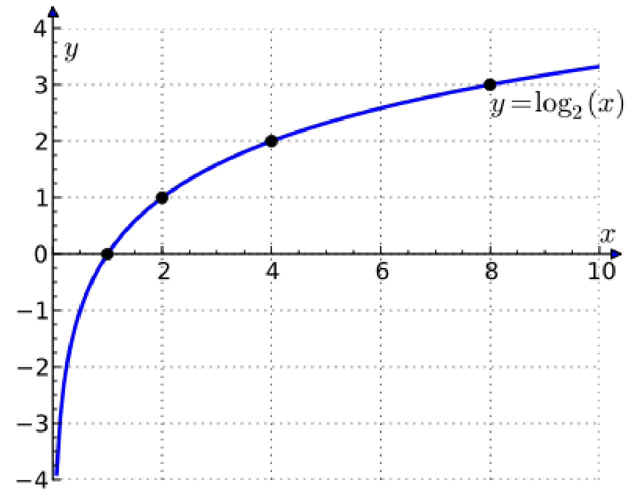
- Maximum a Posterior estimation (MAP) estimation
  - "Best" means being consistent with our "prior" knowledge and explaining data well

$$\widehat{\boldsymbol{\theta}} = \mathbf{argmax}_{\boldsymbol{\theta}} \boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{X}) = \mathbf{argmax}_{\boldsymbol{\theta}} \mathbf{P}(\mathbf{X}|\boldsymbol{\theta})\mathbf{P}(\boldsymbol{\theta})$$

  - Issue: how to define prior?

MAP: Bayesian's point of view

Maximum likelihood e



- Data: a collection of words, $w_1, w_2, \ldots, w_n$
- Model: multinomial distribution $\mathrm{p}(W)$ wit $p(w_i)$
- Maximum likelihood estimator: $\hat{\theta} = argmax_{\theta} p(W|\theta)$

$$p(W|\theta) = \binom{N}{c(x_1), \ldots, c(x_N)} \prod_{i=1}^{V} \theta_i^{c(x_i)} \propto \prod_{i=1}^{V} \theta_i^{c(x_i)} \Rightarrow \log p(W|\theta) = \sum_{i=1}^{V} c(x_i) \log \theta_i$$

$$\Rightarrow L(W, \theta) = \sum_{i=1}^{V} c(x_i) \log \theta_i + \lambda \left(\sum_{i=1}^{V} \theta_i - 1 \right)$$

Using Lagrange multiplier approach, we'll tune $\boldsymbol{\theta_i}$ to maximize $\boldsymbol{L(W, \theta)}$

$$\Rightarrow \frac{\partial L}{\partial \theta_i} = \frac{c(x_i)}{\theta_i} + \lambda \rightarrow \theta_i = -\frac{c(x_i)}{\lambda}$$

Set partial derivatives to zero

$\Rightarrow$ Since $\sum_{i=1}^{V} \theta_i = 1$ we have $\lambda = -\sum_{i=1}^{V} c(x_i)$

Requirement from probability

$$\Rightarrow \theta_i = \frac{c(x_i)}{\sum_{i=1}^{V} c(x_i)}$$

ML estimate

Maximum likelihood estimation

- For N-gram language models
  - $p(w_i|w_{i-1}, \ldots, w_{i-n+1}) = \frac{c(w_i, w_{i-1}, \ldots, w_{i-n+1})}{c(w_{i-1}, \ldots, w_{i-n+1})}$
    - $c(\emptyset) = N$

Length of document or total number of words in a corpus

For the ease of notations, we denote w_i as $w_i = x_i$ as the unique word type in vocabular

Problem with MLE

- Unseen events
  - We estimated a model on 440K word tokens, but:
    - Only 30,000 unique words occurred
    - Only 0.04% of all possible bigrams occurred
  - ➢ This means any word/N-gram that does not occur in the training data has zero probability!
  - ➢ No future documents can contain those unseen words/N-grams

Dealing with Unknown Words: The Simple Solution

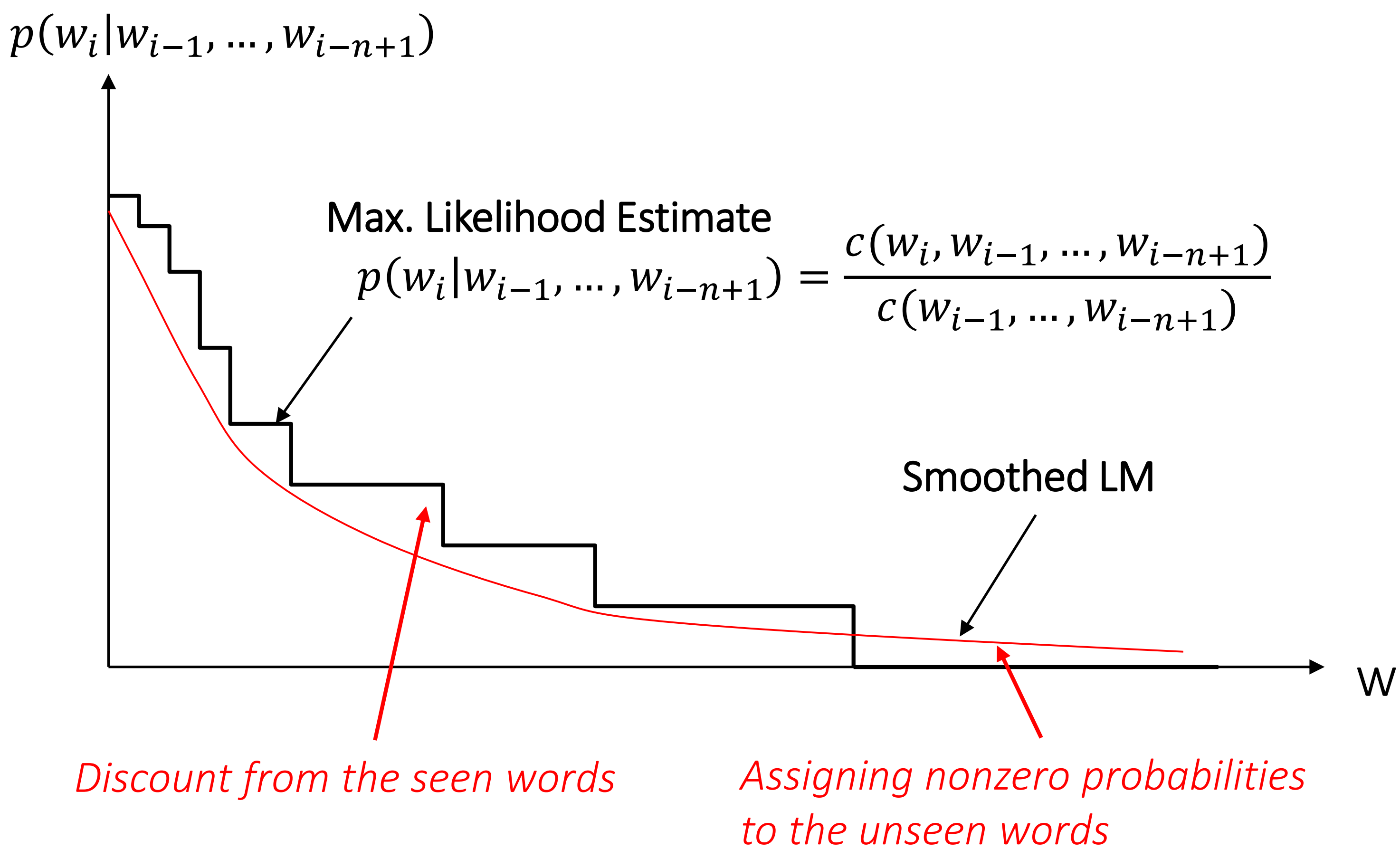
- Training:
  - Assume a fixed vocabulary (e.g. all words that occur at least twice (or n times) in the corpus)
  - Replace all other words by a token <UNK> (or a special OOV)
  - Estimate the model on this corpus
- Testing:
  - Replace all unknown words by <UNK>
  - Run the model

- This requires a large training corpus to work well!
- Note: You cannot fairly compare two language models that apply different UNK treatments!

Smoothing

- If we want to assign non-zero probabilities to unseen words
  - Unseen words = new words, new N-grams
  - Discount the probabilities of observed words
- General procedure
  1. Reserve some probability mass of words seen in a document/corpus
  2. Re-allocate it to unseen words

Illustration of N-gram language model smoothing



Smoothing methods

- Additive smoothing
  - Add a constant δ to the counts of each word
    - Unigram language model as an example

Counts of w in d

"Add one", Laplace smoothing

$$p(w|d) = \frac{c(w,d) + \delta}{|d| + \delta|V|}$$

Vocabulary size

Length of d (total counts)

  - Problems?
    - Hint: all words are equally important?

Add one smoothing for bigrams

Original:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Smoothed:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

After smoothing

- Giving too much to the unseen events

Original:

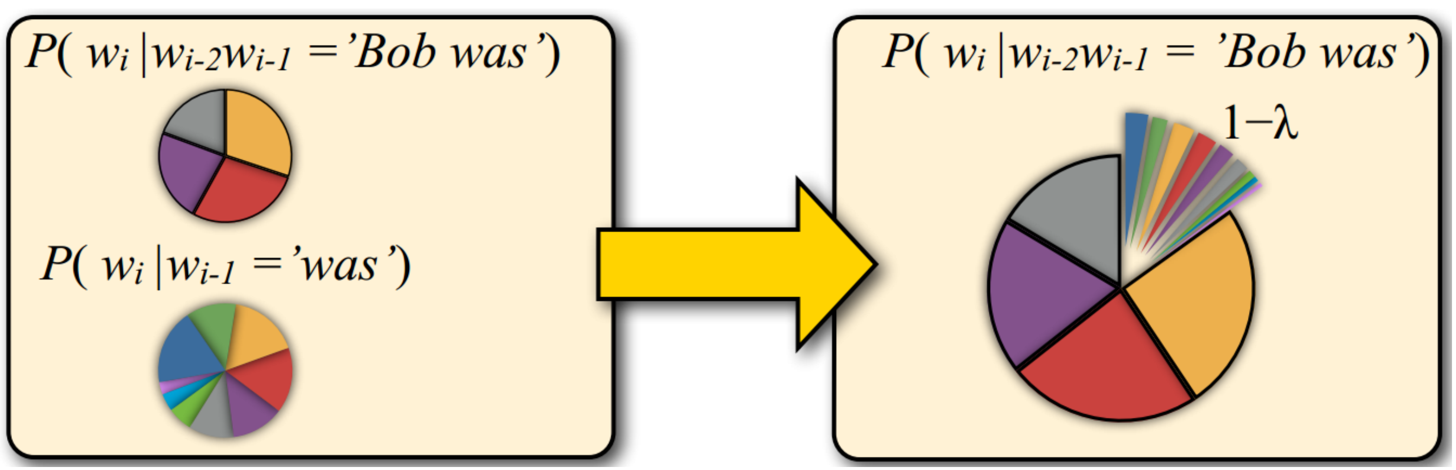
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Smoothed:

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Smoothing methods

- Linear interpolation
  - Use (N –1)-gram probabilities to smooth N-gram probabilities
    - We never see the trigram “Bob was reading”, but we do see the bigram “was reading”



Smoothing methods

- Linear interpolation
  - Use (N −1)-gram probabilities to smooth N-gram probabilities

ML N-gram

$$\bar{p}(w_i|w_{i-1},\dots,w_{i-n+1}) = \lambda p_{ML}(w_i|w_{i-1},\dots,w_{i-n+1}) \\ +(1-\lambda)\bar{p}(w_i|w_{i-1},\dots,w_{i-n+2})$$

Smoothed N-gram

Smoothed (N-1)-gram

  - Further generalize it

$$\bar{p}(w_i|w_{i-1},\dots,w_{i-n+1}) = \lambda_1 p_{ML}(w_i|w_{i-1},\dots,w_{i-n+1}) \\ +\lambda_2\bar{p}(w_i|w_{i-1},\dots,w_{i-n+2}) \\ \dots\dots\dots \\ +\lambda_n\bar{p}(w_i)$$

$$s.t. \sum_{i=1}^{n} \lambda_i = 1$$

Smoothing methods

- Linear interpolation
  - Estimating λs
    - Using a hold-out data set to find the optimal λs
      - An evaluation metric is needed to define "optimality"
    - Define λ as a function of $w_{i-1}, \dots, w_{i-n+2}$

Language model evaluation

- Train the models on the same training set
  - Parameter tuning can be done by holding off some training set for validation
- Test the models on an unseen test set
  - This data set must be disjoint from training data
- Language model A is better than model B
  - If A assigns <u>higher probability</u> to the test data than B

Measuring Model Quality

- The goal isn't to pound out fake sentences!
  - Obviously, generated sentences get “better” as we increase the model order
  - More precisely: using ML estimators, higher order is always better likelihood on train, but not test
- What we really want to know is:
  - Will our model prefer good sentences to bad ones?
  - Bad != ungrammatical!
  - Bad ≈ unlikely
  - Bad = sentences that our model really likes but aren't the correct answer

Perplexity

- Standard evaluation metric for language models
  - A function of the probability that a language model assigns to a data set
  - Rooted in the notion of cross-entropy in information theory

Perplexity

- The inverse of the likelihood of the test set as assigned by the language model, normalized by the number of words

$$PP(w_1, \dots, w_N) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i | w_{i-1}, \dots, w_{i-n+1})}}$$

N-gram language model

An experiment

- Models
  - Unigram, Bigram, Trigram models (with proper smoothing)
- Training data
  - 38M words of WSJ text (vocabulary: 20K types)
- Test data
  - 1.5M words of WSJ text
- Results

	Unigram	Bigram	Trigram
Perplexity	962	170	109

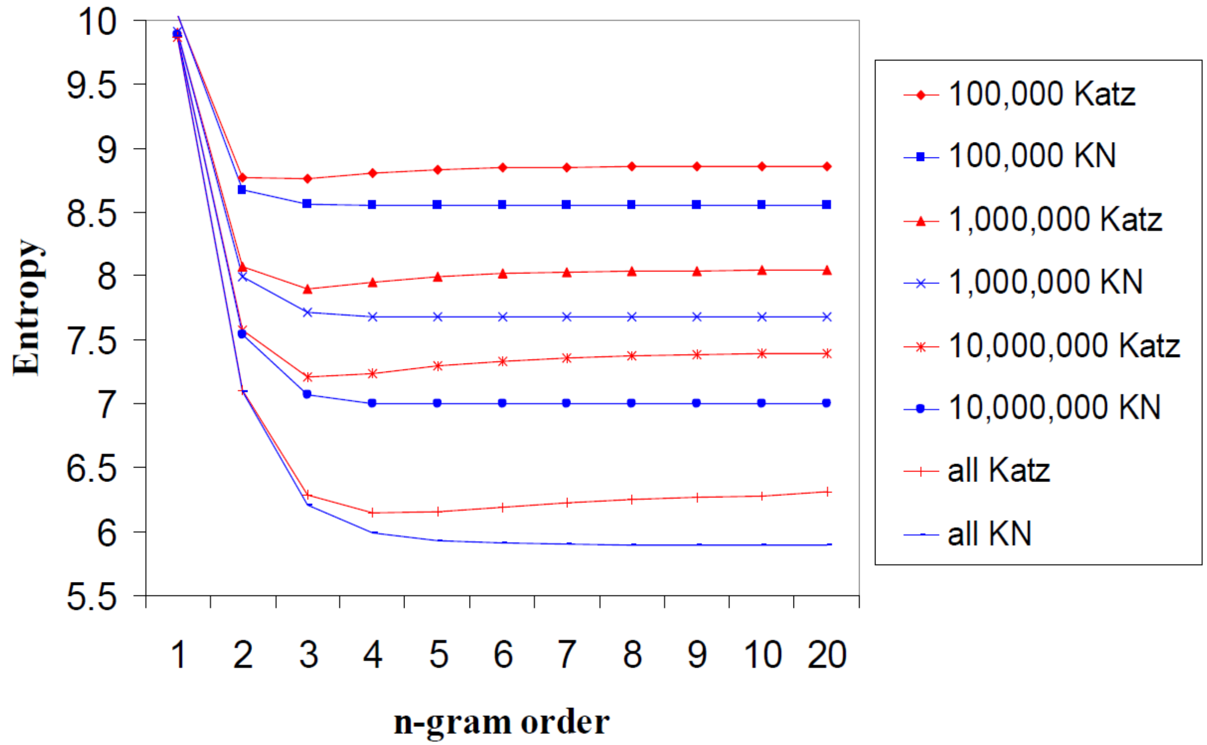
What Actually Works?

- Trigrams and beyond
  – Unigrams, bigrams generally useless for speech or machine translation
  – Trigrams much better (when there's enough data)
  – 4-, 5-grams really useful in MT, but not so much for speech
- Discounting (or smoothing)
  – Absolute discounting, Good-Turing, held-out estimation, Witten-Bell, etc.
- See Chen and Goodman (1996) reading for tons of graphs

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In ACL, pages 310{318.

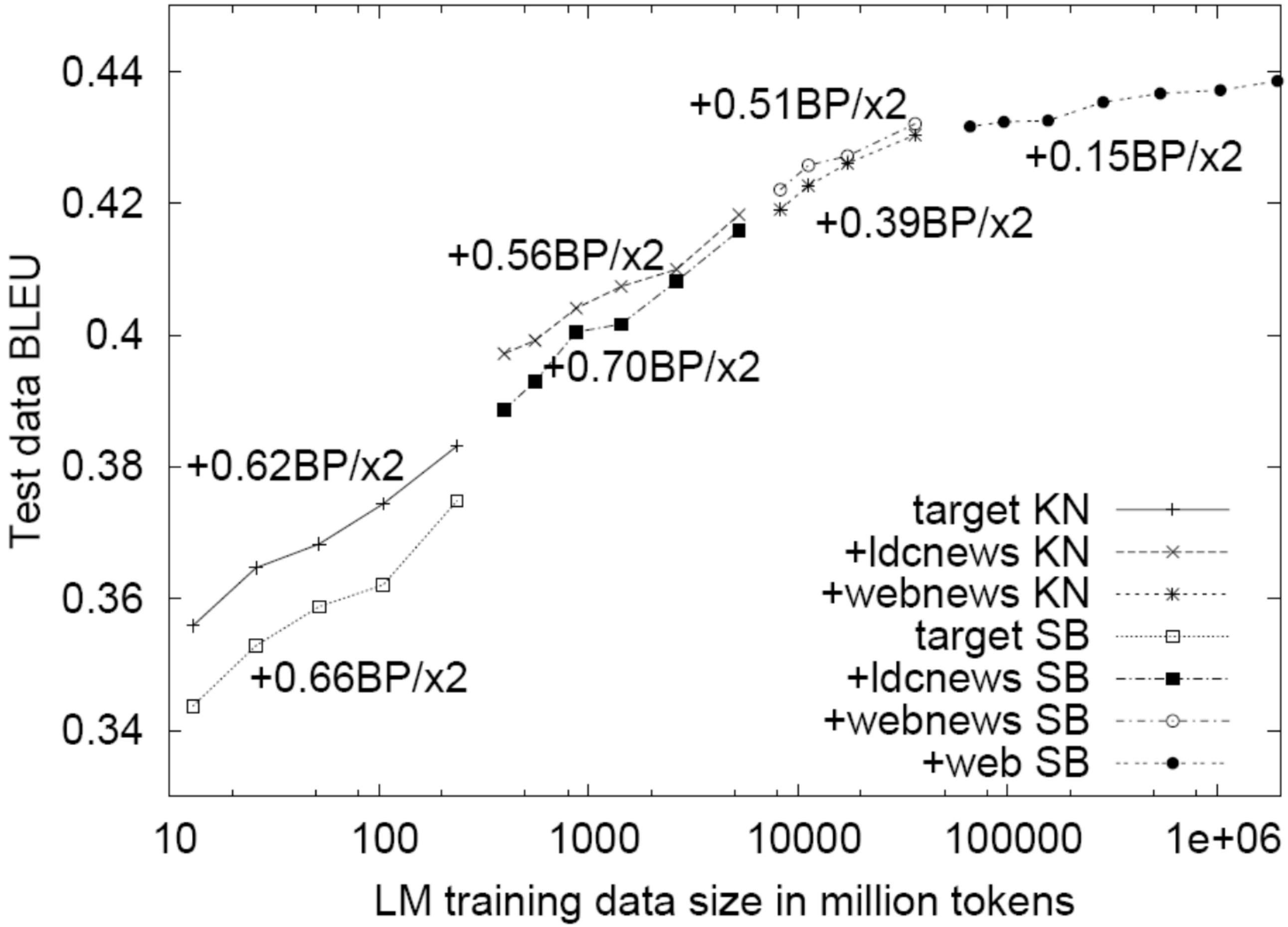
Data vs. Method?

- Having more data is better...
- ...but so is using a better estimator
- Another issue: n > 3 has huge costs in speech recognizers



Tons of Data?

- Tons of data closes gap, for extrinsic MT evaluation



Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, Jeffrey Dean. Large Language Models in Machine Translation. ACL, 2007 http://www.aclweb.org/anthology/D07-1090.pdf

What you should know

- N-gram language models
- How to generate text documents from a language model
- How to estimate a language model
- General idea and different ways of smoothing
- Language model evaluation

Further reading

- Speech and Language Processing
  - Chapter 4: N-Grams