Statistical Learning Models for Text and Graph Data
Lecture 2: Language Models

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

September 6, 2019

∗Contents are based on materials created by Hongning Wang, Julia Hockenmaier, Dan Jurafsky, Dan Klein, Noah Smith, Slav Petrov, Yejin Choi, Gregor Heinrich, and Michael Collins

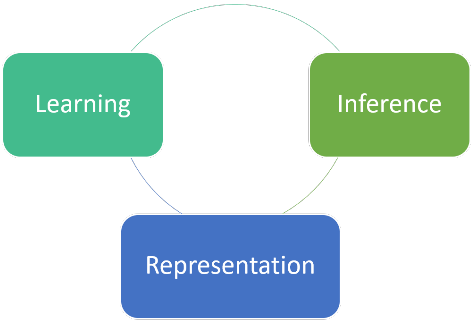
Reference Content

- Noah Smith. CSE 517: Natural Language Processing https://courses.cs.washington.edu/courses/cse517/16wi/
- Julia Hockenmaier. CS447: Natural Language Processing. http://courses.engr.illinois.edu/cs447
- Hongning Wang. CS6501 Text Mining. http://www.cs.virginia.edu/~hw5x/Course/Text-Mining-2015-Spring/_site/
- Dan Jurafsky. cs124/ling180: From Languages to Information. http://web.stanford.edu/class/cs124/
- Dan Klein. CS 288: Statistical Natural Language Processing. https://people.eecs.berkeley.edu/~klein/cs288/sp10/

Reference Content (Cont'd)

- Slav Petrov. Statistical Natural Language Processing. `https://cs.nyu.edu/courses/fall16/CSCI-GA.3033-008/`
- Chris Manning. CS 224N/Ling 237. Natural Language Processing. `https://web.stanford.edu/class/cs224n/`
- Yejin Choi. CSE 517 (Grad) Natural Language Processing. `http://courses.cs.washington.edu/courses/cse517/15wi/`
- Michael Collins. COMS W4705: Natural Language Processing. `www.cs.columbia.edu/~mcollins/courses/nlp2011/`

Course Organization



- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

What is a Statistical Language Model (LM)?

- A model specifying probability distribution over word sequences
  - P("Today is Wednesday") $\approx$ 0.001
  - P("Today Wednesday is") $\approx$ 0.0000000000001
  - P("The eigenvalue is positive") $\approx$ 0.00001
- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

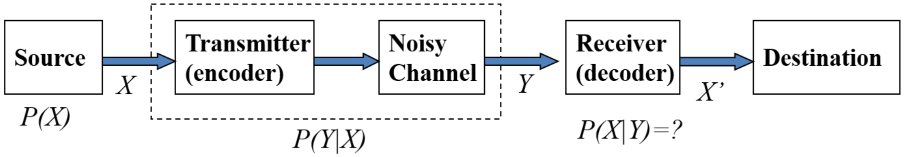
Why is a LM Useful?

- 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 $P_{machine-learning}(d) > P_{health}(d)$, document d belongs to machine learning related topics
  - If $P_{user_a}(d_1) > P_{user_a}(d_2)$, recommend d_1 to $user_a$

Source-Channel Framework [Shannon '48]



$$\hat{X} = \arg\max_X P(X|Y) = \arg\max_X P(Y|X)P(X) \text{ (Bayes Rule)}$$

When X is text, $P(X)$ is a language model

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

- Probability distribution over word sequences (chain rule) $P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_1, w_2, \ldots, w_{n-1})$
- Complexity – $O(V^{n^*})$
    - V: vocabulary size
    - n^*: maximum document (or sentence) length
    - We need independence assumptions!

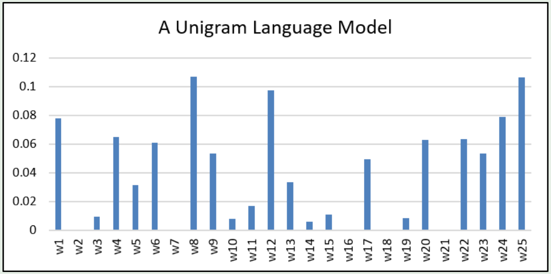
Example

- 475,000 main headwords in Webster's Third New International Dictionary
- Average English sentence length is 14.3 words
- A rough estimate: $O(475,000^{14}) \approx 3.38e^{66}TB$

Unigram Language Model

- Generate a piece of text by generating each word independently
  - $P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2)\ldots P(w_n)$
- Essentially a multinomial distribution over the vocabulary
- The simplest and most popular choice!

Example (Unigram Language Model)



More Sophisticated LMs

N-gram language models

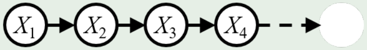
- Assumes each word depends only on the last n-1 words
  - bigram $P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_{n-1})$
  - trigram $P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_{n-1}, w_{n-2})$
- Such independence assumptions are called Markov assumptions (of order n-1)
  $P(w_i|w_1, \ldots, w_{i-1}) = P(w_i|w_{i-n+1}, \ldots, w_{i-1})$

Markov Models

- Value of X at a given time is called the state
- Parameters: called transition probabilities, specify how the state evolves over time (also, initial state probabilities)
- Stationarity assumption: transition probabilities the same at all times

Example (First-order Markov Chain)

"Markov" generally means that given the present state, the future and the past are independent

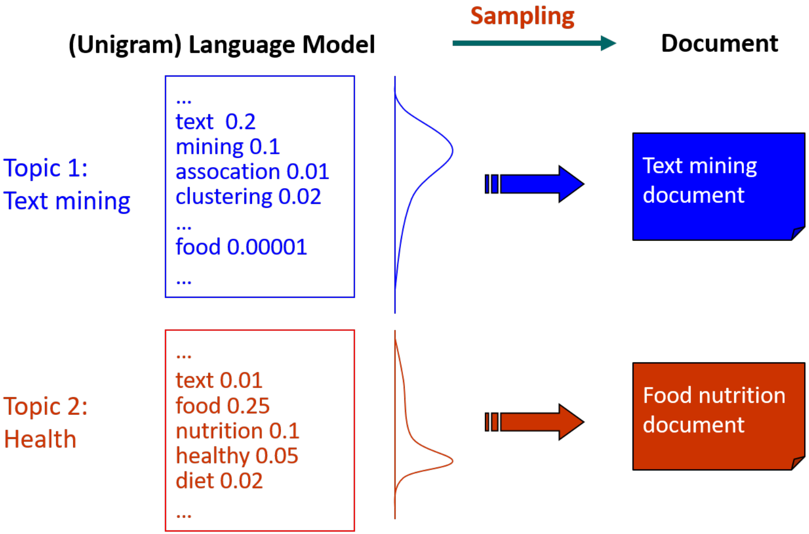


$$P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)\ldots P(X_n|X_{n-1}) = P(X_1)\prod_{t=1}^{n} P(X_t|X_{t-1})$$

Why Just Unigram Models (in most cases)?

- 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

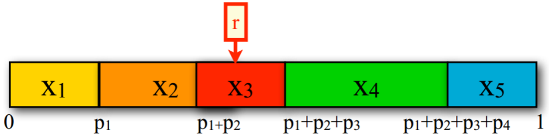


Computer Simulation

Sample from a discrete distribution $P(X)$, assuming n outcomes in the event space X

Algorithm 1 Sample from a distribution $P(X)$

```
1: for $t = 1$ to $T$ do
2:     Divide the interval $[0, 1]$ into $n$ intervals according to the probabilities of the outcomes
3:     Generate a random number $r$ between 0 and 1
4:     Return $x_i$ where $r$ falls into $[\sum_0^{i-1} p_i, \sum_0^{i} p_i]$
5: end for
```



Generating Text from Language Models

Example

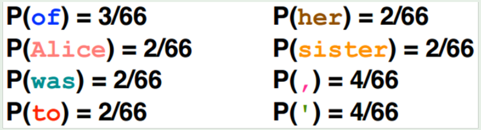
P(of) = 3/66	P(her) = 2/66
P(Alice) = 2/66	P(sister) = 2/66
P(was) = 2/66	P(,) = 4/66
P(to) = 2/66	P(') = 4/66

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

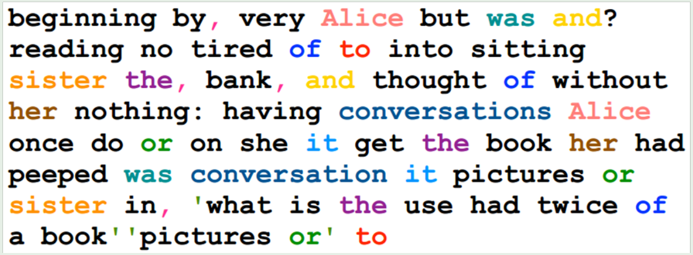
Example



P(of) = 3/66	P(her) = 2/66
P(Alice) = 2/66	P(sister) = 2/66
P(was) = 2/66	P(,) = 4/66
P(to) = 2/66	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
```

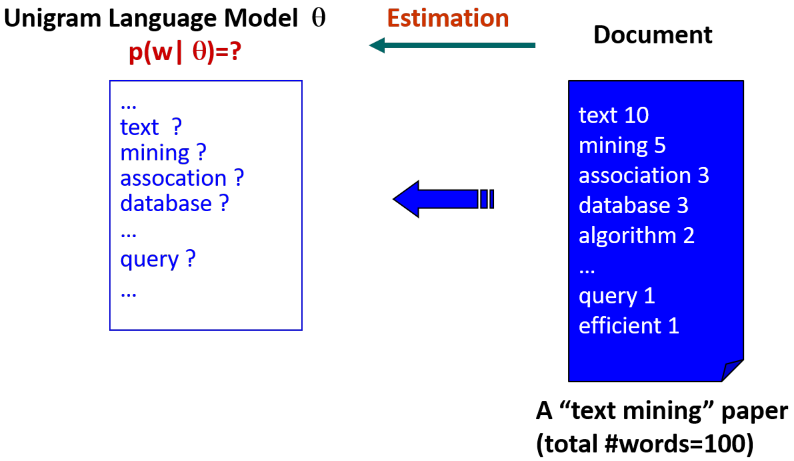
N-gram Language Models Will Help

Example (Generated from language models of New York Times)

- Unigram
  - Months the my and issue of year foreign new exchanges 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.

Overview

Estimation of Language Models



Unigram Language Model θ

p(w| θ)=?

Estimation

Document

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

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

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

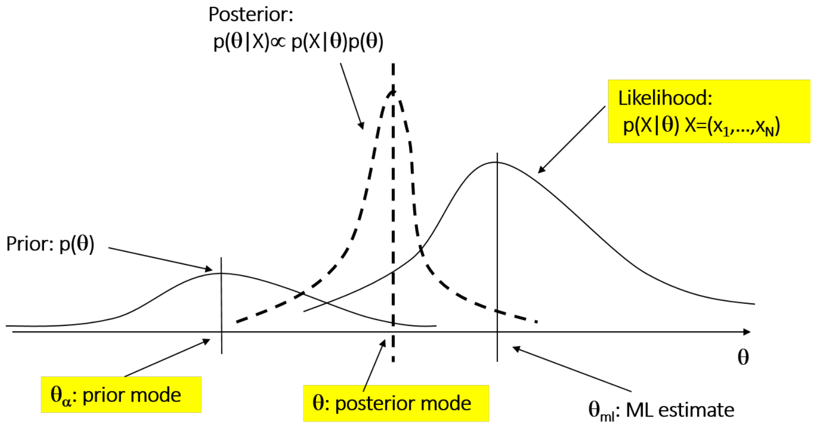
Parameter Estimation

- General setting
  - Given a (hypothesized & probabilistic) model that governs the random experiment
  - The model gives a probability of any data $P(\mathcal{X}|\boldsymbol{\theta})$ that depends on the parameter $\boldsymbol{\theta}$
  - Now, given actual sample data $\mathcal{X} = \mathbf{x}_1, \ldots, \mathbf{x}_n$, what can we say about the value of $\boldsymbol{\theta}$?
- Intuitively, take our best guess of $\boldsymbol{\theta}$
  - "best" means "best explaining/fitting the data"
- Generally an optimization problem

Maximum Likelihood vs. Bayesian

- Maximum likelihood estimation
  - "Best" means "data likelihood reaches maximum"
$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{X}|\boldsymbol{\theta})$$
  - Issue: small sample size
- Bayesian estimation
  - "Best" means being consistent with our "prior" knowledge and explaining data well
$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathcal{X}) = \arg\max_{\boldsymbol{\theta}} P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta})$$
  - A.k.a, maximum a posterior estimation
  - Issue: how to define prior?

Illustration of Bayesian Estimation


Posterior:
$p(\theta|X) \propto p(X|\theta)p(\theta)$
Likelihood:
$p(X|\theta)$ $X=(x_1,...,x_N)$
Prior: $p(\theta)$
θ
θ_α: prior mode
θ: posterior mode
θ_{ml}: ML estimate

Corpora

- A corpus is a collection of text
  - Annotated in some way: supervised learning
  - Sometimes just lots of text without any annotations: unsupervised learning
  - Balanced vs. uniform corpora
- Examples
  - Newswire collections: 500M+ words
  - Brown corpus: 1M words of tagged balanced text
  - Penn Treebank: 1M words of parsed WSJ
  - Canadian Hansards: 10M+ words of aligned French / English sentences
  - The Web: billions of words of who knows what

Unigram Modeling

- Data corpus: a collection of words, $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$
- Model: multinomial distribution $P(\mathcal{W}|\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_V)$, where
  - $\theta_i = P(v_i)$
  - $v_i \in \mathcal{V}$
  - $\mathcal{V}$ is the vocabulary
  - $|\mathcal{V}| = V$
- Count of words in corpus $\mathbf{u} = (u_1, \ldots, u_V)$ where $u_i = c(v_i)$ is the count of v_i shown in $\mathcal{W}$, $\sum_i u_i = N$

Unigram Modeling

- "Bag of words" assumes the words are sampled from a multinomial distribution $\mathbf{u} \sim \mathrm{Multi}(\boldsymbol{\theta})$

$$P(\mathbf{u}|\boldsymbol{\theta}) = \begin{pmatrix} N \\ \mathbf{u} \end{pmatrix} \prod_{i=1}^{V} \theta_i^{u_i} \triangleq \mathrm{Mult}(\mathbf{u}|\boldsymbol{\theta}, N), \textit{where} \begin{pmatrix} N \\ \mathbf{u} \end{pmatrix} = \frac{N!}{\prod_i u_i!}$$

If we focus on a single trial, we have:

$$P(w|\boldsymbol{\theta}) = P(w = v_i) = \prod_{i=1}^{V} \theta_i^{\delta_{w=v_i}} \triangleq \mathrm{Mult}(w|\boldsymbol{\theta})$$

- Maximum likelihood estimator: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{W}|\boldsymbol{\theta})$

$$P(\mathcal{W}|\boldsymbol{\theta}) = \prod_{j=1}^{N} P(w_j|\boldsymbol{\theta}) = \prod_{i=1}^{V} P(v_i)^{u_i} = \prod_{i=1}^{V} \theta_i^{u_i}$$

Maximum Likelihood Estimation: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{W}|\boldsymbol{\theta})$

$$P(\mathcal{W}|\boldsymbol{\theta}) = \prod_i^V \theta_i^{u_i}$$

(log likelihood)

$$\Rightarrow \log P(\mathcal{W}|\boldsymbol{\theta}) = \sum_i^V u_i \log \theta_i$$

(Lagrange multiplier to make $\boldsymbol{\theta}$ be a distribution)

$$\Rightarrow L(\mathcal{W}, \boldsymbol{\theta}) = \log P(\mathcal{W}|\boldsymbol{\theta}) = \sum_i^V u_i \log \theta_i + \lambda(\sum_i \theta_i - 1)$$

(Set partial derivatives to zero)

$$\Rightarrow \frac{\partial L}{\partial \theta_i} = \frac{u_i}{\theta_i} + \lambda = 0$$

Since $\sum_i^V \theta_i = 1$, we have $\lambda = -\sum_i^V u_i$

$$\Rightarrow \theta_i = \frac{u_i}{\sum_i^V u_i} = \frac{u_i}{N} \; (\textit{Maximum Likelihood Estimation}\,, \textit{MLE})$$

The Problems with Unigram Modeling

- Pros:
  - Easy to understand
  - Cheap
  - Good enough for information retrieval (maybe)
- Cons:
  - "Bag of words" assumption is linguistically inaccurate
    - P(the the the the) $\gg$ P(I want ice cream)
  - Data sparseness; high variance in the estimator
  - "Out of vocabulary" problem

N-gram model

- Markov modeling

$$\begin{aligned} & P(w_1, \ldots, w_N) \\ = \; & \prod_{i=1}^{N} P(w_i | w_1, \ldots, w_{i-1}) \; (\textit{chain rule}) \\ = \; & \prod_{i=1}^{N} P(w_i | w_{i-1}, \ldots, w_{i-n+1}) \; (\textit{Markov model}) \end{aligned}$$

  - (n - 1)th-order Markov assumption $\equiv$ n-gram model
    - Unigram model is the n = 1 case
    - For a long time, trigram models (n = 3) were widely used
    - 5-gram models (n = 5) are not uncommon now in machine translation systems

- Parameter estimation

$$P(w_i | w_{i-1}, \ldots, w_{i-n+1}) = \frac{c(v^1 = w_i, \ldots, v^n = w_{i-n+1})}{c(v^1 = w_{i-1}, \ldots, v^{n-1} = w_{i-n+1})}$$

v^j is a unique word v at position j

Estimating N-gram models: A Running Example

Example (Bigram Model)

- Bracket each sentence by special start and end symbols:
  $\langle s \rangle$ Alice was beginning to get very tired ... $\langle s \rangle$
  (We only assign probabilities to strings $\langle s \rangle$...$\langle s \rangle$)
- Count the frequency of each n-gram
  c($\langle s \rangle$, Alice) = 1, c(Alice, was) = 1,
- Normalize to get the probability
  $P(w_i|w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$
  $P(was|Alice) = \frac{c(was, Alice)}{c(Alice)}$
- This is called a relative frequency estimate of $P(w_i|w_{i-1})$

The Problems with N-gram Modeling

- The curse of dimensionality: the number of parameters grows exponentially in n
- Pros:
  - Easy to understand
  - Cheap (with modern hardware; Lin and Dyer (2010))
  - Good enough for machine translation, speech recognition, ...
- Cons:
  - Markov assumption is linguistically inaccurate
    - (But not as bad as unigram models!)
  - Data sparseness; high variance in the estimator
    - most n-grams will never be observed, even if they are linguistically plausible
  - "Out of vocabulary" problem

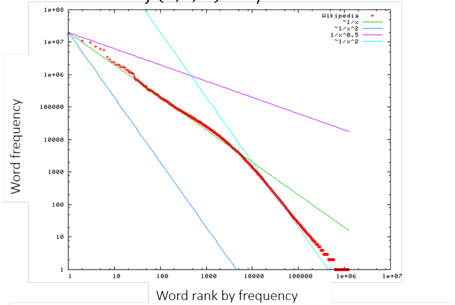
Overview

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

- In natural language:
  - A small number of events (e.g. words) occur with high frequency
  - A large number of events occur with very low frequency
  - Zipfs law: the long tail

$f(k;s,N) \propto 1/k^s$



A plot of word frequency in Wikipedia (Nov 27, 2006)

Dealing with Unseen Events

- Relative frequency (maximum likelihood) estimation assigns all probability mass to events in the training corpus
- But we need to reserve some probability mass to events that don't occur in the training data
    - Unseen events = new words, new bigrams
- Important questions:
    - What possible events are there?
    - How much probability mass should they get?

Dealing with Unseen Events

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

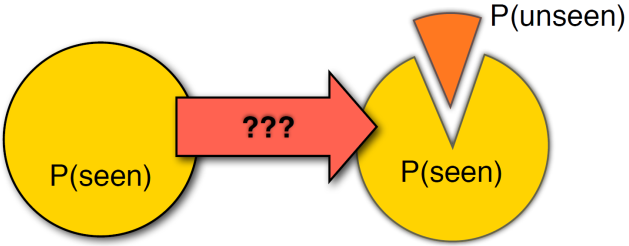
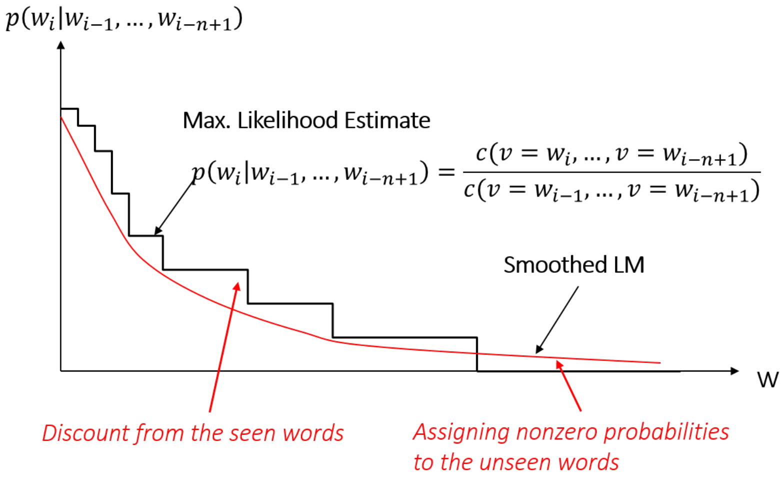


Illustration of N-gram Language Model Smoothing



What Unseen Events May Occur?

- Simple distributions:

$$P(X = x)$$

  (e.g. unigram models)
- Possibility:
  - The outcome x has not occurred during training (i.e. is unknown)
  - We need to reserve mass in $P(X)$ for x
- What outcomes x are possible?
- How much mass should they get?

What Unseen Events May Occur?

- Simple conditional distributions:
$$P(X = x|Y = y)$$
(e.g. bigram models)
- Possibility:
  - The outcome x has been seen, but not in the context of $Y = y$:
  - We need to reserve mass in $P(X|Y = y)$ for $X = x$
- The conditioning variable y has not been seen:
  - We have no $P(X|Y = y)$ distribution.
  - We need to drop the conditioning context $Y = y$ and use $P(X)$ instead.

What Unseen Events May Occur?

- Complex conditional distributions:
$$P(X = x|Y = y, Z = z)$$
(e.g. trigram models)
- Possibility:
  - The outcome x has been seen, but not in the context of $(Y = y, Z = z)$:
  - We need to reserve mass in $P(X|Y = y, Z = z)$ for $X = x$
- The joint conditioning event $(Y = y, Z = z)$ has not been seen:
  - We have no $P(X|Y = y, Z = z)$ distribution.
  - We need to drop z and use $P(X|Y = y)$ instead.

Examples

Example

- Training data: The wolf is an endangered species
- Test data: The wallaby is endangered

Unigram	Bigram	Trigram
$P(the)$	$P(the \mid \langle s \rangle)$	$P(the \mid \langle s \rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby \mid the)$	$\times\ P(wallaby \mid the, \langle s \rangle)$
$\times\ P(is)$	$\times\ P(is \mid wallaby)$	$\times\ P(is \mid wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered \mid is)$	$\times\ P(endangered \mid is, wallaby)$

Examples

Example

- Training data: The wolf is an endangered species
- Test data: The wallaby is endangered

Unigram	Bigram	Trigram
$P(the)$	$P(the \vert \langle s \rangle)$	$P(the \vert \langle s \rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby \vert the)$	$\times\ P(wallaby \vert the, \langle s \rangle)$
$\times\ P(is)$	$\times\ P(is \vert wallaby)$	$\times\ P(is \vert wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered \vert is)$	$\times\ P(endangered \vert is, wallaby)$

- Case 1:
  - $P(wallaby)$, $P(wallaby|the)$, $P(wallaby|the, \langle s \rangle)$
  - What is the probability of an unknown word (in any context)?

Examples

Example

- Training data: The wolf is an endangered species
- Test data: The wallaby is endangered

Unigram	Bigram	Trigram
$P(the)$	$P(the \mid \langle s \rangle)$	$P(the \mid \langle s \rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby \mid the)$	$\times\ P(wallaby \mid the, \langle s \rangle)$
$\times\ P(is)$	$\times\ P(is \mid wallaby)$	$\times\ P(is \mid wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered \mid is)$	$\times\ P(endangered \mid is, wallaby)$

- Case 2:
  - $P(endangered \mid is)$
  - What is the probability of a known word in a known context, if that word hasn't been seen in that context?

Examples

Example

- Training data: The wolf is an endangered species
- Test data: The wallaby is endangered

Unigram	Bigram	Trigram
$P(the)$	$P(the \mid \langle s \rangle)$	$P(the \mid \langle s \rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby \mid the)$	$\times\ P(wallaby \mid the, \langle s \rangle)$
$\times\ P(is)$	$\times\ P(is \mid wallaby)$	$\times\ P(is \mid wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered \mid is)$	$\times\ P(endangered \mid is, wallaby)$

- Case 3:
  - $P(is \mid wallaby)$, $P(is \mid wallaby, the)$, $P(endangered \mid is, wallaby)$
  - What is the probability of a known word in an unseen context?

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 $\langle UNK \rangle$ (or a special OOV)
  - Estimate the model on this corpus
- Testing:
  - Replace all unknown words by $\langle UNK \rangle$
  - 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!

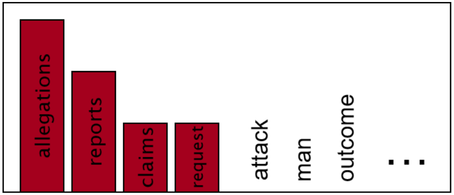
Overview

Dealing with Unknown Words

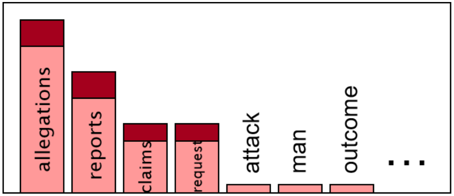
- Use a different estimation technique:
  - Add-one (Laplace) Smoothing
  - Good-Turing Discounting
  - Idea: Replace MLE estimate $P(w) = \frac{c(w)}{N}$
- Combine a complex model with a simpler model:
  - Linear Interpolation
  - Modified Kneser-Ney smoothing
  - Idea: use bigram probabilities $P(w_i|w_{i-1})$ to calculate trigram probabilities $P(w_i|w_{i-1}, w_{i-2})$ of w

Smoothing: Intuition

- When we have sparse statistics ($P(w|denied\ the)$):



- Steal probability mass to generalize better



Add-one (Laplace) Smoothing

- Assume every (seen or unseen) event occurred once more than it did in the training data
- Example: unigram probabilities
  - Estimated from a corpus with N tokens and a vocabulary (number of word types) of size V.
    MLE:
    $$\Rightarrow \theta_i = \frac{u_i}{\sum_i^V u_i} = \frac{u_i}{N}$$
    Add one:
    $$\Rightarrow \theta_i = \frac{u_i + 1}{\sum_i^V (u_i + 1)} = \frac{u_i + 1}{N + V}$$
    where $u_i = c(v_i)$ is the count of v_i shown in training set $\mathcal{W}$, $\sum_i u_i = N$

Add One Smoothing for Bigrams

Original:

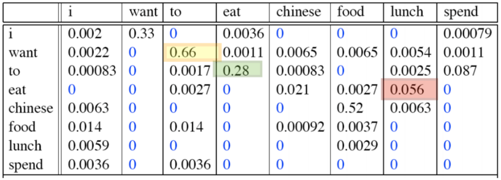
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

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

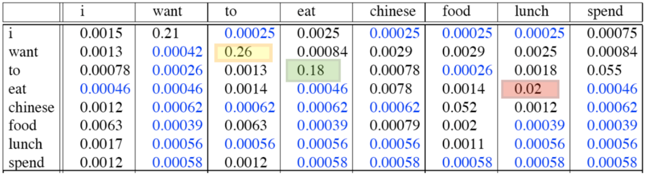
Add One Smoothing for Bigrams

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

Problem: Add-one moves too much probability mass from seen to unseen events!

Summary: Add-One smoothing

- Advantage:
  - Very simple to implement
- Disadvantage:
  - Takes away too much probability mass from seen events
  - Assigns too much total probability mass to unseen events

Example (The Shakespeare example)

- $V = 30,000$ word types; "the" occurs $25,545$ times
- Bigram probabilities for "the...":
  $P(w_i|w_{i-1} = the) = \frac{c(the, w_i)+1}{25{,}545+30{,}000}$

Overview

Generalization: Add-K smoothing

Problem: Add-one moves too much probability mass from seen to unseen events!

- Variant of Add-One smoothing
  - Add a constant k to the counts of each word
  - For any $k > 0$ (typically, $k < 1$), a unigram model is

$$\Rightarrow \theta_i = \frac{u_i + k}{\sum_i^V u_i + kV} = \frac{u_i + k}{N + kV}$$

- If $k = 1$
  - "Add one" Laplace smoothing
- This is still too simplistic to work well.

Any explanation?

Bayesian Interpretation

- Conjugate distribution
  - Adding a conjugate prior to a likelihood will result in a posterior in the same distribution family as the prior, then the prior and the likelihood are called conjugate distributions
  - Conjugate distribution makes us easier to formulate Bayesian belief and inference the model

Bayesian Interpretation

- The conjugate prior of a multinomial is Dirichlet distribution:
  $P(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}) \triangleq \frac{\Gamma(\sum_{i=1}^{V}\alpha_i)}{\prod_{i=1}^{V}\Gamma(\alpha_i)} \prod_{i=1}^{V} \theta_i^{\alpha_i - 1} \triangleq \frac{1}{\Delta(\boldsymbol{\alpha})} \prod_{i=1}^{V} \theta_i^{\alpha_i - 1}$
  - The "Dirichlet Delta function" $\Delta(\boldsymbol{\alpha})$ is introduced for convenience
  - $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_V)^\top \in \mathbb{R}^V$
  - The Gamma function satisfies $\Gamma(x+1) = x\Gamma(x)$
    - For integer variable, Gamma function is just factorial $\Gamma(x) = (x-1)!$
    - For real numbers, it is $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} \mathrm{d}t$
- The Dirichlet distribution can be seen as the *"distribution of a distribution"*
  - We can sample a multinomial distribution from Dirichlet distribution, satisfied the constraint $\sum_i \theta_i = 1$

Bayesian Interpretation

- The Dirichlet distribution can be seen as the *"distribution of a distribution"*
  - We can sample a multinomial distribution from Dirichlet distribution, satisfied the constraint $\sum_i \theta_i = 1$
  - In two variables case, multinomial reduces to binomial and Dirichlet reduces to Beta distribution

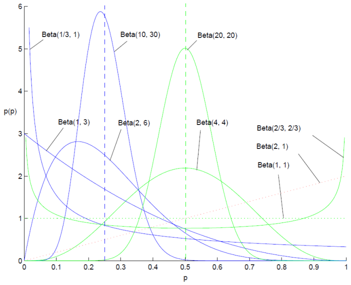


Figure copied from Heinrich (2008)

Bayesian Interpretation

- The Dirichlet distribution can be seen as the *"distribution of a distribution"*
  - We can sample a multinomial distribution from Dirichlet distribution, satisfied the constraint $\sum_i \theta_i = 1$
  - Three variables: distribution defined over a simplex

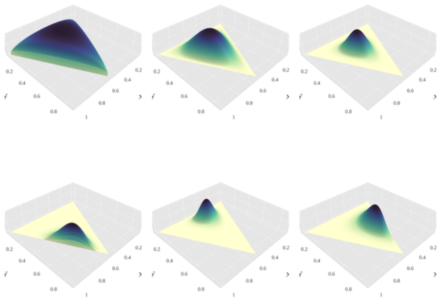


Figure copied from Wikipedia:
`https://en.wikipedia.org/wiki/Dirichlet_distribution`

Bayesian Estimation

- Remember Maximum likelihood estimator: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{W}|\boldsymbol{\theta})$

$$P(\mathcal{W}|\boldsymbol{\theta}) = \prod_{j=1}^{N} P(w_j|\boldsymbol{\theta}) = \prod_{i=1}^{V} P(v_i)^{u_i} = \prod_{i=1}^{V} \theta^{u_i} (\theta_i = \frac{u_i}{\sum_i^V u_i} = \frac{u_i}{N})$$

- The posterior of the parameters $\boldsymbol{\theta}$ based on the prior and the observation of N words:

$$\begin{aligned} P(\boldsymbol{\theta}|\mathcal{W},\boldsymbol{\alpha}) &= \frac{P(\mathcal{W}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha})}{P(\mathcal{W}|\boldsymbol{\alpha})} = \frac{\prod_{i=1}^{N} P(w_i|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha})}{\int_{\boldsymbol{\theta}} \prod_{i=1}^{N} P(w_i|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha})\mathrm{d}\boldsymbol{\theta}} \\ &= \frac{\prod_{i=1}^{N} P(w_i|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha})}{Z} \\ &= \frac{1}{Z}\prod_{i=1}^{V} \theta_i^{u_i} \frac{1}{\Delta(\boldsymbol{\alpha})} \prod_{i=1}^{V} \theta_i^{\alpha_i - 1} \\ &= \frac{1}{\Delta(\boldsymbol{\alpha}+\mathbf{u})} \prod_{i=1}^{V} \theta_i^{\alpha_i + u_i - 1} = \mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}+\mathbf{u}) \end{aligned}$$

- According to the property of Dirichlet distribution, the posterior is with mean $\theta_i = \frac{u_i+\alpha_i}{\sum_i^V u_i + V\alpha_i}$ and mode $\theta_i = \frac{u_i+\alpha_i-1}{\sum_i^V u_i + V(\alpha_i - 1)}$ (MAP, maximum a posterior estimation, estimation), and $\alpha_i = 1$ equals to MLE

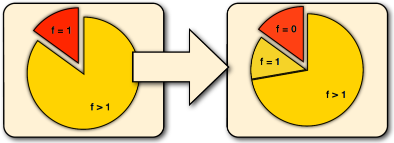
Overview

Good-Turing Smoothing

- Question: why the same discount for all n-grams?
- Good-Turing Discounting: invented during WWII by Alan Turing and later published by Good (1953)
- Motivation
  - $P(seen) + P(unseen) = 1$
  - MLE: $\Leftrightarrow \frac{N}{N} + 0 = 1$
  - Good tuning: $\Leftrightarrow \frac{2 \cdot N_2 + \ldots + m \cdot N_m}{\sum_{i=1}^{m} i \cdot N_i} + \frac{1 \cdot N_1}{\sum_{i=1}^{m} i \cdot N_i} = 1$
    - N_r: number of event types that occur r times $(c(w_1, \ldots, w_n) = r)$
    - N_1: number of event types that occur once $(c(w_1, \ldots, w_n) = 1)$
    - $N = \sum_{i=1}^{m} i \cdot N_i$: total number of observed event tokens
- Quick idea
  - Now, use the modified counts $c^*(w_1, \ldots, w_n) = (r+1)\frac{N_{r+1}}{N_r}$ for events that occur r times

Good-Turing Smoothing Intuition

- You are fishing (a scenario from Josh Goodman), and caught:
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- How likely is it that next species is trout?
  - 1/18
- How likely is it that next species is new (i.e. catfish or bass)
  - Let's use our estimate of things-we-saw-once to estimate the new things
  - 3/18 (because $N_1 = 3$)
- Assuming so, how likely is it that next species is trout?
  - Must be less than 1/18
  - How to estimate?



Relative Frequency Estimate   Good Turing Estimate

Good-Turing Smoothing: More Details

- **General principle**: Reassign the probability mass of all events that occur r times in the training data to all events that occur $r-1$ times

The probability mass of all words that appear $r-1$ times becomes:

$$\sum_{w:c(w)=r-1} P_{GT}(w) = \sum_{w':c(w')=r} P_{MLE}(w') = \sum_{w':c(w')=r} \frac{r}{N} = \frac{r \cdot N_r}{N}$$

- N_r events occur r times, with a total frequency of $r \cdot N_r$
- Good Turing smoothing uses the modified counts: replaces the original count c_r of $w_1, ..., w_n$ with a new count c_r^*
  - $c_r^*(w_1, ..., w_n) = \frac{(r+1) \cdot N_{r+1}}{N_r}$ where $c(w_1, ..., w_n) = r$
    - i.e., N_r events occur $\frac{(r+1) \cdot N_{r+1}}{N_r}$ times
  - $c_{r-1}^*(w_1, ..., w_n) = \frac{r \cdot N_r}{N_{r-1}}$ where $c(w_1, ..., w_n) = r-1$
  - ...
  - $\sum_{r=1}^{m} N_r c_r^*(w_1, ..., w_n) = \sum_{r=1}^{m} (r+1) \cdot N_{r+1} = N - \frac{N_1}{N}$
  - Then, our estimate of the missing mass is: $\frac{N_1}{N}$

Good-Turing Smoothing Example

- You are fishing (a scenario from Josh Goodman), and caught:
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- Unseen (bass or catfish)
  - $c = 0$
  - $P_{MLE} = 0/18 = 0$
  - $P_{GT}(unseen) = N_1/N = 3/18$
- Seen once (trout)
  - $c = 1$
  - $P_{MLE} = 1/18$
  - $c^*(trout) = 2 * N_2/N_1 = 2 * 1/3 = 2/3$
  - $P_{GT}(trout) = 2/3/18 = 1/27$

Problems with Good-Turing

- Problem 1:
  - What happens to the most frequent event?
- Problem 2:
  - We don't observe events for every k.
- Variant (tricks): Simple Good-Turing
  - Replace N_n with a fitted function $f(n) = a + b\log(n)$:
  - Requires parameter tuning (on held-out data):
    - Set a, b so that $f(n) \approx N_n$ for known values.
    - Use c_n^* only for small n

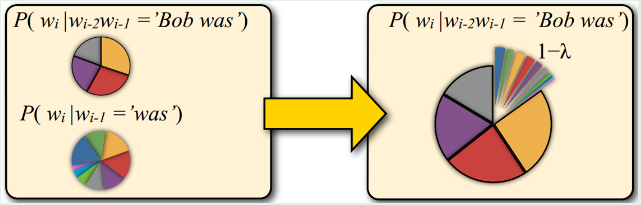
Overview

Linear Interpolation

- Linear interpolation: Use (n-1)-gram probabilities to smooth n-gram probabilities:
  $\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+1}) =$
  $\lambda P_{MLE}(w_i|w_{i-1},\ldots,w_{i-n+1}) + (1-\lambda)\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+2})$
  - $\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+1})$ is smoothed n-gram
  - $P_{MLE}(w_i|w_{i-1},\ldots,w_{i-n+1})$ is MLE result
  - $\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+2})$ is smoothed (n-1)-gram

Example (We never see the trigram "Bob was reading,")

But we might have seen the bigram "was reading", and we have certainly seen "reading" (or $\langle UNK\rangle$)



Linear Interpolation (Cont'd)

- Linear interpolation: further generalization

$$\begin{aligned} & \bar{P}(w_i|w_{i-1},\dots,w_{i-n+1}) \\ = \; & \lambda_1 P_{MLE}(w_i|w_{i-1},\dots,w_{i-n+1}) \\ + \; & \lambda_2 \bar{P}(w_i|w_{i-1},\dots,w_{i-n+2}) \\ + \; & \dots \\ + \; & \lambda_n \bar{P}(w_i) \end{aligned}$$

- Again $P_{MLE}(w_i|w_{i-1},\dots,w_{i-n+1})$ is MLE result

Linear Interpolation (Cont'd)

- Estimating λ_i's
  - Using a hold-out data set to find the optimal λ_i's
  - An evaluation metric is needed to define "optimality"
  - We will come back to this later

Absolute Discounting

- Absolute discounting
  - Subtract a constant δ from each nonzero n-gram count and then interpolate

$$\begin{aligned} & \bar{P}(w_i|w_{i-1},\ldots,w_{i-n+1}) \\ = \quad & \frac{\max(0,c(w_i,\ldots,w_{i-n+1})-\delta)}{c(w_{i-1},\ldots,w_{i-n+1})} + \lambda\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+2}) \end{aligned}$$

- If S seen word types (unique words in vocabulary) occur after $w_{i-1},\ldots,w_{i-n+1}$ in the training data, this reserves the probability mass $P(u) = \frac{\delta S}{c(w_{i-1},\ldots,w_{i-n+1})}$ to be reallocated according to $\bar{P}(w_i|w_{i-1},\ldots,w_{i-n+2})$
- We set $\lambda = \frac{\delta S}{c(w_{i-1},\ldots,w_{i-n+1})}$

Overview

Kneser-Ney Smoothing

- Observation: "San Francisco" is frequent, but "Francisco" only occurs after "San"
  - "Francisco" will get a high unigram probability, and so absolute discounting will give a high probability to "Francisco" appearing after novel bigram histories.
  - Better to give "Francisco" a low unigram probability, because the only time it occurs is after San, in which case the bigram model fits well.
- Solution: the unigram probability $P(w)$ should not depend on the frequency of w, but on the number of contexts in which w appears
  - $N_{+1}(\cdot, w)$: number of contexts in which w appears = number of word types (unique words in vocabulary) w' which precede w (w="Francisco", count "San" only once)
  - $N_{+1}(\cdot, \cdot) = \sum_w N_{+1}(\cdot, w)$
- Kneser-Ney smoothing: Use absolute discounting, but use $P(w) = N_{+1}(\cdot, w)/N_{+1}(\cdot, \cdot)$ to smooth bigram language model

Overview

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 higher probability 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 $\neq$ ungrammatical!
  - Bad $\approx$ unlikely
  - Bad $=$ sentences that our model really likes but aren't the correct answer

Measuring Model Quality (Cont'd)

- The Shannon Game (by Claude Shannon, 1916–2001):
  - How well can we predict the next word?

When I eat pizza, I wipe off the ______

grease	0.5
sauce	0.4
dust	0.05
...	
mice	0.0001
...	
the	$1e-100$

  - Unigrams are terrible at this game.
- How good are we doing?
  - Compute per word log likelihood (N words, M test sentences S_i):
  - An intuitive way: $I = \frac{1}{N}\sum_i^N \log P(S_i)$

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

- Perplexity of a probability distribution

$$2^{H(P)} = 2^{-\sum_x P(x) \log_2 P(x)}$$

  - $H(P)$: entropy
  - Perplexity of a random variable X may be defined as the perplexity of the distribution over its possible values x
  - In the special case where P models a uniform distribution over k discrete events, its perplexity is k
- Perplexity of a probability model

$$2^{H(\hat{P},Q)} = 2^{-\sum_x \hat{P}(x) \log_2 Q(x)}$$

  - $H(\hat{P}, Q)$: cross entropy
  - $\hat{P}$ denotes the empirical distribution of the test sample (i.e., $\hat{P}(x) = n/N$ if x appeared n times in the test sample of size N)
  - Q: a proposed probability model
  - One may evaluate Q by asking how well it predicts a separate test sample $x_1, x_2, ..., x_N$ also drawn from unknown P

The Shannon Game Intuition for Perplexity

- How hard is the task of recognizing digits “0,1,2,3,4,5,6,7,8,9” at random
  - Perplexity 10
- How hard is recognizing (30,000) names at random
  - Perplexity 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

Perplexity as Branching Factor

- Language with higher perplexity means the number of words branching from a previous word is larger on average
- The difference between the perplexity of a language model and the true perplexity of the language is an indication of the quality of the model

Perplexity Per Word for Language Models

- Given a test corpus with N tokens, $w_1, \ldots, w_N$, and an n-gram model $P(w_i | w_{i1}, \ldots, w_{in+1})$ the perplexity $PP(w_1, \ldots, w_N)$ is defined as follows (Brown et al. (1992)):
- The inverse of the likelihood of the test set as assigned by the language model, normalized by the number of words

$$\begin{aligned} PP(w_1, \ldots, w_N) &= P(w_1, \ldots, w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1, \ldots, w_N)}} \\ &= \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i | w_1, \ldots, w_{i-1})}} \; (chain\ rule) \\ &= \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i | w_{i-1}, \ldots, w_{i-n+1})}} \; (n-gram\ model) \end{aligned}$$

- Minimizing perplexity = maximizing probability!
- Language model LM_1 is better than LM_2 if LM_1 assigns lower perplexity (= higher probability) to the test corpus $w_1, \ldots, w_N$
- Note: the perplexity of LM_1 and LM_2 can only be directly compared if both models use the same vocabulary.

Practical Issues

- Since language model probabilities are very small, multiplying them together often yields to underflow
- It is often better to use logarithms instead, so replace

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

with

$$PP(w_1, \ldots, w_N) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i | w_{i-1}, \ldots, w_{i-n+1})\right)$$

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

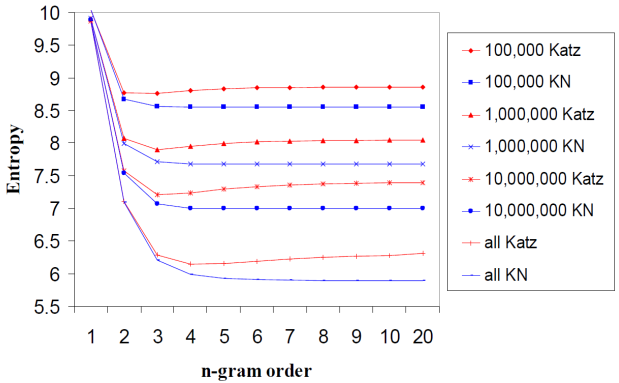
- Conclusion: The bigram is much better than the unigram, and the trigram is even better

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
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell, etc.
- See Chen and Goodman (1996) reading for tons of graphs

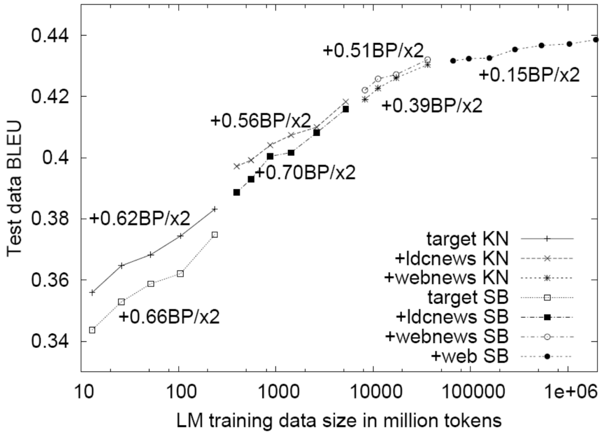
Data vs. Method?

- 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



http://www.aclweb.org/anthology/D07-1090.pdf

Overview

Further Reading

- Manning et al. (2008). Introduction to information retrieval. Chapter 12: Language models for information retrieval.
- Jurafsky and Martin (2017). Speech and Language Processing. Chapter 4: N-Grams. https://web.stanford.edu/~jurafsky/slp3/
- Chen and Goodman (1996). An empirical study of smoothing techniques for language modeling.
- Collins (2011). Course notes for COMS w4705: Language modeling, 2011. http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/lm.pdf
- Zhu (2010). Course notes for cs769: Language modeling, 2011. http://pages.cs.wisc.edu/~jerryzhu/cs769/lm.pdf

References

Brown, P. F., Pietra, V. J. D., Mercer, R. L., Pietra, S. A. D., and Lai, J. C. (1992). An estimate of an upper bound for the entropy of english. *Comput. Linguist.*, 18(1):31–40.

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

Collins, M. (2011). Course notes for coms w4705: Language modeling. Technical report, Columbia University.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40 (3 and 4):237–264.

Heinrich, G. (2008). Parameter estimation for text analysis. Technical Report Version 2.4, vsonix GmbH + University of Leipzig, Germany.

Jurafsky, D. and Martin, J. H. (2017). *Speech and Language Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Zhu, X. J. (2010). Course notes for cs769: Language modeling. Technical report, University of Wisconsin-Madison.