Statistical Learning for Text Data Analytics
Language Models

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

Spring 2018

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

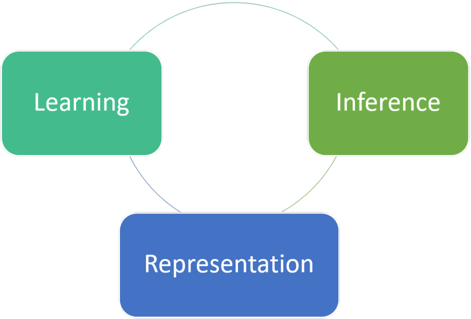
Reference Content

- 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 Topics



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

NLP applications: tasks introduced in Lecture 1

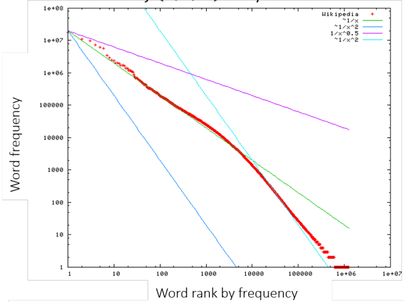
Overview

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

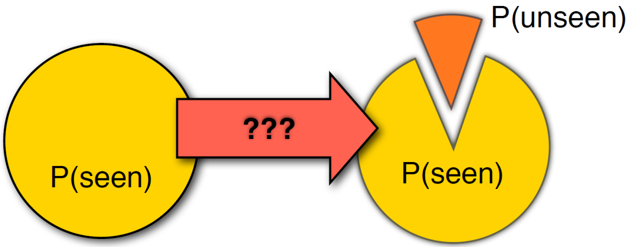
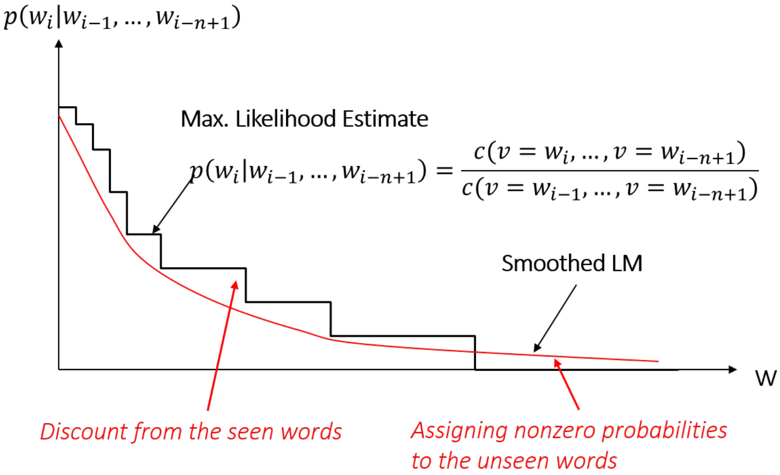


Illustration of N-gram Language Model Smoothing



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
$P(the)$	$P(the\|\langle s\rangle)$	$P(the\|\langle s\rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby\|the)$	$\times\ P(wallaby\|the, \langle s\rangle)$
$\times\ P(is)$	$\times\ P(is\|wallaby)$	$\times\ P(is\|wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered\|is)$	$\times\ P(endangered\|is, wallaby)$

- 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
$P(the)$	$P(the\|\langle s\rangle)$	$P(the\|\langle s\rangle)$
$\times\ P(wallaby)$	$\times\ P(wallaby\|the)$	$\times\ P(wallaby\|the, \langle s\rangle)$
$\times\ P(is)$	$\times\ P(is\|wallaby)$	$\times\ P(is\|wallaby, the)$
$\times\ P(endangered)$	$\times\ P(endangered\|is)$	$\times\ P(endangered\|is, wallaby)$

- Case 2:
  - $P(endangered|is)$
  - 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!

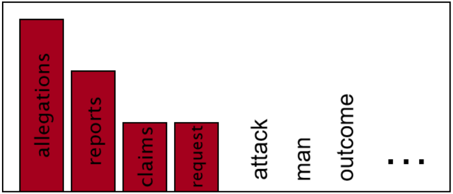
Overview

Dealing with Unknown Words

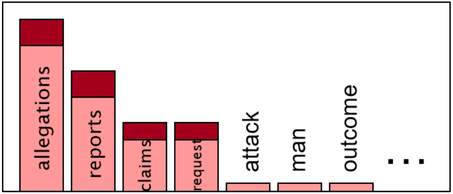
- Use a different estimation technique:
  - Add-one (Laplace) Smoothing
  - Good-Turing Discounting
  - Idea: Replace MLE estimate $P(w) = \frac{c(w)}{N}$
- Combine a complex model with a simpler model:
  - Linear Interpolation
  - Modified Kneser-Ney smoothing
  - Idea: use bigram probabilities $P(w_i|w_{i-1})$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

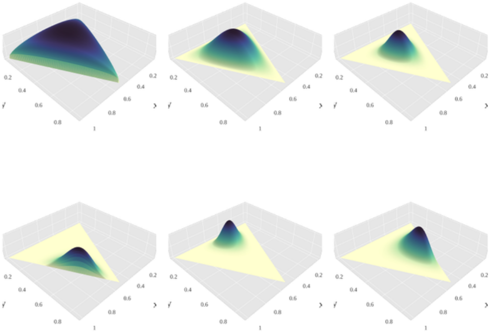
- Conjugate distribution
  - Adding a conjugate prior to a likelihood will result in a posterior in the same distribution family as the prior, then the prior and the likelihood are called conjugate distributions
  - Conjugate distribution makes us easier to formulate Bayesian belief and inference the 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 $\Gamma(x) = (x-1)!$
        - 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 $\sum_i \theta_i = 1$



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:

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

- We have MAP (maximum a posterior estimation) estimate as $\theta_i = \frac{u_i+\alpha_i-1}{\sum_i^V u_i + V(\alpha_i-1)}$ ($\alpha_i = 1$ equals to MLE)

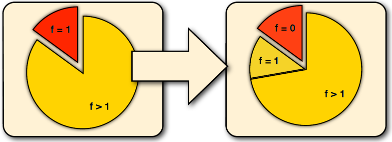
Overview

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

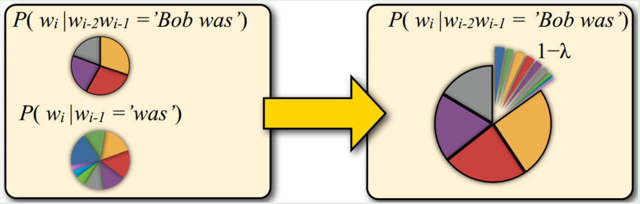
Overview

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

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

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

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

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.

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

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.