Smoothing

L645

Dept. of Linguistics, Indiana University
Fall 2009

Smoothing – Definitions

- the N-gram matrix for any given training corpus is **sparse**
  - i.e., not all n-grams will be present
  - MLE produces bad estimates when the counts are small
- **smoothing** = re-evaluating zero and small probabilities, assigning very small probabilities for zero-N-grams
  - if non-occurring N-grams receive small probabilities, the probability mass needs to be redistributed!
  - smoothing also sometimes called **discounting**

Types Vs. Tokens

- **token**: single item
  type: abstract class of items
- example: words in text
  - token: each word
  - type: each **different** word, i.e., wordform
- sentence: `the man with the hat`

tokens:	the, man, with, the, hat	# of tokens = 5
types:	the, man, with, hat	# of types = 4

Basic Techniques

An overview of what we'll look at:

- Add-One Smoothing (& variations)
  - Laplace's, Lidstone's, & Jeffreys-Perks laws
- Deleted estimation: validate estimates from one part of corpus with another part
- Witten-Bell smoothing: use probabilities of seeing events for the first time
- Good-Turing estimation: use ratios between $n+1$ and n-grams

Following Manning and Schütze, we'll use n-gram language modeling as our example

Basics of n-grams

n-grams are used to model language, capturing some degree of grammatical properties

- Thus, we can state the probability of a word based on its history:

  (1) $P(w_n|w_1...w_{n-1})$

- n-gram probabilities are estimated as follows

  (2) $P(w_n|w_1...w_{n-1}) = \frac{P(w_1...w_n)}{P(w_1...w_{n-1})}$

- To avoid data sparsity issues, bigrams and trigrams are commonly used
- We can use maximum likelihood estimation (MLE) to obtain basic probabilities:

  (3) $P(w_n|w_1...w_{n-1}) = \frac{C(w_1...w_n)}{C(w_1...w_{n-1})}$

But MLE probabilties do nothing to handle unseen data

Add-One Smoothing

Idea: pretend that non-existent bigrams are there once

- to make the model more just: assume that for each bigram we add one to the count
- . . . turns out not to be a very good estimator

Add-One Smoothing
Laplace's Law

- **unigram probabilities**:

  N = number of tokens
  C(x) = frequency of x
  V = vocabulary size; number of types
- standard probability for word w_x: $P(w_x) = \frac{C(w_x)}{N}$
- adjusted count: $C^*(w_x) = (C(w_x) + 1)\frac{N}{N+V}$
  - $\frac{N}{N+V}$ is a normalizing factor, $N + V$ is the new "size" of the text
- $p^*(w_x)$: estimated probability
  - probability: $p^*(w_x) = \frac{(C(w_x)+1)\frac{N}{N+V}}{N} = \frac{c(w_x)+1}{N+V}$

Test Corpus: Windows Haiku Corpus

- Haiku: Japanese poem, each poem has only 17 syllables; 5 syllables in the first line, 7 in the second, 5 in the third
- corpus: 16 haikus, 253 tokens, 165 words
- Windows NT crash'd.
  I am the Blue Screen of Death.
  No-one hears your screams.
- Yesterday it work'd.
  Today it is not working.
  Windows is like that.
- Three things are certain:
  Death, taxes and lost data.
  Guess which has occurred.

Test Corpus: Add-One Smoothing

word	freq.	**unsmoothed**: $\frac{C(w)}{N}$	**add-one**: $\frac{C(w)+1}{N+V}$
.	35	0.1383	0.0860
,	8	0.0316	0.0215
the	7	0.0277	0.0191
The	4	0.0158	0.0119
that	3	0.0119	0.0095
on	2	0.0079	0.0072
We	1	0.0040	0.0048
operator	0	0.0000	0.0024

Add-One Smoothing: Bigrams

- $P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$
- $p^*(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$

Bigrams Example

bigram	freq. bigram	freq. w_{n-1}	**unsmoothed**: $\frac{C(w_{n-1}w_n)}{C(w_{n-1})}$	**add-one**: $\frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$
. END	35	35	1.0000	0.1800
START The	3	35	0.0857	0.0200
START You	2	35	0.0571	0.0150
is not	2	7	0.2857	0.0174
Your ire	1	2	0.5000	0.0120
You bring	1	3	0.3333	0.0119
not found	1	4	0.2500	0.0118
is the	0	7	0	0.0058
This system	0	2	0	0.0060

Lidstone's & Jeffreys-Perks Laws

Because Laplace's law overestimates non-zero events, variations were created:

- Lidstone's law: instead of adding one, add some smaller value λ

(4) $P(w_1...w_n) = \frac{C(w_1...w_n)+\lambda}{N+V\lambda}$

- Jeffreys-Perks law: set λ to be $\frac{1}{2}$ (the expectation of maximized MLE):

(5) $P(w_1...w_n) = \frac{C(w_1...w_n)+\frac{1}{2}}{N+\frac{1}{2}}$

Problems: How do we guess λ? And still not good for low frequency n-grams . . .

Towards Deleted Estimation
Held-Out Estimation

To get an idea as to whether a smoothing technique is effective, we can use held-out estimation.

- Split the data into training data and held-out data
- Use the held-out data to see how good the training estimates are

Using bigrams as an example:

- Say that there are N_r bigrams with frequency r in the training data
- Count up how often all these bigrams together occur in the held-out data; call this T_r
- Average frequency in held-out data is thus $\frac{T_r}{N_r}$

Held-Out Estimation

Since N is the number of training instances, the probability of one of these n-grams is $\frac{T_r}{N_r N}$

This re-estimate can provide one of two different things:

- A reality check on the smoothing technique being used
- A better estimate to be used on the testing data
  - It is critical that the testing data be disjoint from both the held-out and the training data

Deleted Estimation

Held-Out Estimation keeps the held-out data separate from the training data

- But what if we split the training data in half?
  - We could train on one half and validate on the other
  - And then we could switch the training and validation portions
- With both of these estimates, we can average them to obtain even more reliable estimates

$$(6)\quad p_{del}(w_1 w_2) = \frac{T_r^1 + T_r^2}{N(N_r^1 + N_r^2)}$$

Deleted Estimation

Deleted Estimation turns out to be quite good ... but not for low frequency *n*-grams

What's wrong with low-frequency *n*-grams?

- Overestimates unseen objects (& underestimates one-time objects)
  - The *n*-grams that appear 0 times in one half of the training data are counted in the other
  - But as the size of the data increases, there are generally less unseen *n*-grams
    - In other words, the number of unseen objects is not linear, but deleted estimation assumes it is
  - Smaller training sets lead to more unseen events in the held-out data

Witten-Bell Discounting

Problems with Add-One Smoothing:

- add-one smoothing leads to sharp changes in probabilities
- too much probability mass goes to unseen events

Witten-Bell: think of unseen events as ones not having happened yet

- the probability of this event – when it happens – can be modeled by the probability of **seeing it for the first time**

First Time Probability

How do we estimate probability of an N-gram occurring for the first time?

- count number of times of seeing an N-gram for the first time in training corpus
- think of corpus as series of events: one event for each token and one event for each new type
- e.g. unigrams:

corpus:	a		man		with		a	hat
event:	a	**new**	man	**new**	with	**new**	a	hat . . .

- number of events: $N + T$

Witten-Bell Probabilities

- **total** probability mass for unseen events: $\sum_{x:C(w_x)=0} p^*(w_x) = \frac{T}{N+T}$
- probability for **one** unseen unigram: $p^*(w_x) = \frac{T}{Z(N+T)}$
  - divide total prob. mass up for all unseen events
  - number of all unseen unigrams: $Z = \sum_{x:C(w_x)=0} 1$
- discount total probability mass for unseen events from other events
  $p^*(w_x) = \frac{C(w_x)}{N+T} \qquad$ for $C(w_x) > 0$
- alternatively: **smoothed counts**:

$$C^*(w_x) = \begin{cases} \frac{T}{Z}\frac{N}{N+T} & \text{if } C(w_x) = 0 \\ C(w_x)\frac{N}{N+T} & \text{if } C(w_x) > 0 \end{cases}$$

Witten-Bell Smoothed Bigrams

Type counts are conditioned on previous word: use probability of bigram **starting with previous word**

- $T(w_x)$ = number of bigrams starting with w_x

Zero-count events:

- total prob. mass: $\sum_{i:C(w_{i-1}w_i)=0} p^*(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{N+T(w_{i-1})}$
- $p^*(w_i \mid w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N+T(w_{i-1}))}$ if $C(w_{i-1}w_i) = 0$
  - $Z(w_{i-1}) = \sum_{i:C(w_{i-1}w_i)=0} 1$

Non-zero-count events:

- $p^*(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{N+T(w_{i-1})}$ if $C(w_{i-1}w_i) > 0$
  - $N = C(w_{i-1})$

$T(w)$ And $Z(w)$ from Haikus

$Z(w)$ = number of unseen bigrams starting with w

complete number of bigram starting with w: V

$Z(w) = V - T(w) = 165 - T(w)$

word	$T(w)$	$Z(w)$
.	1	164
START	13	152
is	6	159
Your	2	163
You	2	163
not	4	161
This	2	163

Haiku Probabilities

bigram	unsmoothed	add-one	Witten-Bell
. END	1.0000	0.1800	0.9722
START The	0.0857	0.0200	0.0625
START You	0.0571	0.0150	0.0417
is not	0.2857	0.0174	0.1538
Your ire	0.5000	0.0120	0.2500
You bring	0.3333	0.0119	0.2000
not found	0.2500	0.0118	0.1250
is the	0	0.0058	0.0029
This system	0	0.0060	0.0031

Good-Turing-Smoothing

Idea: re-estimate probability *mass* assigned to N-grams with zero counts

- by looking at probability mass of **all** N-grams with count 1
- based on assumption of binomial distribution

Idea broken down:

- create classes N_c of N-grams which occur c times
- the size of class N_c is the frequency of frequency c

This works well for N-grams.

Good-Turing-Smoothing (2)

- smoothed count c: $c^* = (c+1)\frac{N_{c+1}}{N_c}$
  - $N_c = \sum_{b:c(b)=c} 1$
- smoothed count for unseen events: $c^* = \frac{N_1}{N_0}$

Haiku counts:

c	N_c	c^*
0	26980	0.0087
1	236	0.0593
2	7	0.4286
3	1	0
4	0	

Good-Turing-Smoothing (3)

- Problem: for highest count c, $N_{c+1} = 0$!!!
  - i.e. $c^* = (c+1)\frac{N_{c+1}}{N_c} = (c+1)\frac{0}{N_c} = 0$
- Solution: discount only for small counts $c \leq k$ (e.g. $k = 5$)
  - $c^* = c \qquad\qquad \text{for } c > k$
- New discounting:

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}} \qquad\qquad \text{for } 1 \leq c \leq k$$

Haiku Bigrams

- G-T = $\frac{c^*(w_{n-1}w_n)}{C(w_{n-1})}$
- $k = 3$

Haiku Bigrams

- G-T = $\frac{c^*(w_{n-1}w_n)}{C(w_{n-1})}$
- $k = 3$

bigram	count	orig.	add-1	W-B	G-T
. END	35	1.0000	0.1800	0.9722	1.0000
START The	3	0.0857	0.0200	0.0625	0.0857
START You	2	0.0571	0.0150	0.0417	0.0122
is not	2	0.2857	0.0174	0.1538	0.1837
Your ire	1	0.5000	0.0120	0.2500	0.0297
You bring	1	0.3333	0.0119	0.2000	0.0198
not found	1	0.2500	0.0118	0.1250	0.0148
is the	0	0	0.0058	0.0029	0.0012
This system	0	0	0.0060	0.0031	0.0044

Backoff

Idea: go back to "smaller" N-grams

- i.e. do not only use trigram prob. but also bigrams and unigrams
- no trigram found, use bigram; if no bigram found, use unigram

... can be used instead of smoothing

- need to weight contribution of specific N-gram:

$$P^*(w_i \mid w_{i-2}w_{i-1}) = \begin{cases} P(w_i \mid w_{i-2}w_{i-1}) & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 P(w_i \mid w_{i-1}) & \text{if } C(w_{i-2}w_{i-1}w_i) = 0 \text{ and} \\ & C(w_{i-1}w_i) > 0 \\ \alpha_2 P(w_i) & \text{otherwise} \end{cases}$$

Linear Interpolation
Simple linear interpolation

Simple linear interpolation involves mixing different pieces of information to derive a probability

- Called deleted interpolation with subset relations (e.g., bigrams and unigrams are subsets of trigrams)

(7) $\hat{P}(w_i|w_{i-2}w_{i-1}) =$
$\lambda_1 P(w_i|w_{i-2}w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i)$

- $\sum \lambda_i = 1$
- $0 \leq \lambda_i \leq 1$

Every trigram probability is a linear combination of the focus word's trigram, bigram, and unigram.

- Use EM algorithm on held-out data to calculate λ values

Linear Interpolation
General linear interpolation

More generally, we can condition the word on its history and each λ can be based on the history, too

(8)
$$\begin{aligned} P(w|h) &= \sum_i \lambda_i(h)P_i(w|h) \\ &= \lambda_1(h)P_1(w|h) + \lambda_2(h)P_2(w|h) + \lambda_3(h)P_3(w|h) \end{aligned}$$

- P_1 may focus on the trigram history, while P_2 uses the bigram, and so forth.
- So, instead of having one λ_1 for all trigrams, we have individualized it for each unique trigram
  - Useful, in that every trigram potentially behaves differently
  - But there's a big sparse data problem

Equivalence bins

To overcome the sparse data problem, λ's are calculated by putting them into equivalence bins

- One method (Chen and Goodman 1996) bases the bins on the number of different words which an $n-1$-gram has following it

$$(9) \quad \frac{C(w_1...w_{i-1})}{|w_i : C(w_1...w_i) > 0|}$$

  - $w_i : C(w_1...w_i) > 0$ means: the set of w_i such that the trigram exists

- So, *great deal* occurs 178 times, with 36 different words after it: average count = 4.94
- *of that* occurs 178 times, with 115 different words after it: 1.55
  - These histories will thus prompt different λ values