Introduction to Language Modeling
(Parsing and Machine Translation Reading Group)

Christian Scheible

October 23, 2008

Outline

Language Models

What?

Probability distribution for word sequences:
$p(w_1, ..., w_n) = p(w_1^n)$

$\Rightarrow$ How likely is a sentence to be produced?

Why?

- Machine translation
- Speech recognition
- ...

N-gram Models

- Conditional probabilities for words given their N predecessors
- Independence assumption!

$\longrightarrow$ Approximation

Example

$p(\text{My dog finds a bone}) \approx$
$p(\text{My})\, p(\text{dog}|\text{my})\, p(\text{finds}|\text{my}, \text{dog})\, p(\text{a}|\text{dog}, \text{finds})\, p(\text{bone}|\text{finds}, \text{a})$

Why Smoothing?

Smoothing: Adjusting maximum likelihood estimates to produce more accurate probabilities

Why Smoothing?

- Probabilities: relative frequencies (from corpus)
  $p(w_1, ..., w_n) = \frac{f(w_1, ..., w_n)}{N}$
- But not all combinations of $w_1, ... w_n$ are covered

$\longrightarrow$ Data sparseness leads to probability 0 for many words

Example

p(My dog found two bones)
p(My dog found three bones)
p(My dog found seventy bones)
p(My dog found eighty sausages)
p(My dog found ... ...)

Smoothing Models – Overview

- Laplace Smoothing
- Additive Smoothing
- Good-Turing Estimate
- Katz Backoff
- Interpolation (Jelinek-Mercer)
- Absolute Discounting
- Witten-Bell Smoothing
- Kneser-Ney Smoothing

Laplace Smoothing (1)

- Add 1 to the numerator
- Add the size of the vocabulary (V) to the denominator

$$p(w_1, ..., w_n) = \frac{f(w1, ..., w_n) + 1}{N + |V|}$$

Laplace Smoothing (2)

Problems

- Intended for uniform distributions, language data usually produces Zipf distributions
- Overestimates new items, underestimates known items
- Unrealistically alters probabilities of items with high frequencies

MLE	Empirical	Laplace
0	0.000027	0.000137
1	0.448	0.000247
2	1.25	0.000411
3	2.24	0.000548
4	3.23	0.000685
5	4.21	0.000822
6	5.23	0.000959
7	6.21	0.00109
8	7.21	0.00123
9	8.26	0.00137

Additive Smoothing

- Add λ to the numerator
- Add $\lambda \times |V|$ to the denominator

$$p(w_1, ... w_n) = \frac{f(w1, ... w_n) + \lambda}{N + \lambda|V|}$$

Problems

- λ needs to be determined
- Same issues as Laplace smoothing

Good-Turing Estimate (1)

- Idea: Reallocate the probability mass of events that occur $n+1$ times to the events that occur n times
- For each frequency f, produce an adjusted (expected) frequency f^*

$$f^* \approx (f+1)\frac{E(n_{f+1})}{E(n_f)} \approx (f+1)\frac{n_{f+1}}{n_f}$$

$$p_{GT}(w_1, ..., w_n) = \frac{f^*(w_1, ..., w_n)}{\sum_X f^*(X)}$$

Good-Turing Estimate (2)

MLE	Test Set	Laplace	Good-Turing
0	0.000027	0.000137	0.000027
1	0.448	0.000274	0.446
2	1.25	0.000411	1.26
3	2.24	0.000548	2.24
4	3.23	0.000685	3.24
5	4.21	0.000822	4.22
6	5.23	0.000959	5.19
7	6.21	0.00109	6.21
8	7.21	0.00123	7.24
9	8.26	0.00137	8.25

Problems

- What if n_r is 0?

$\longrightarrow$ n_r need to be smoothed

- No combination with other distributions

Katz Backoff (1)

- Combines probability distributions
- Idea: Higher- and lower-order distributions
- Recursive smoothing up to unigrams

$$p(w_i | w_k, ..., w_{i-1}) = \begin{cases} \frac{f(w_k, ..., w_n)\delta}{f(w_k, ..., w_{n-1})} & \text{if } f(...) > 0 \\ \alpha\, p(w_i | w_{k+1}, ... w_{i-1}) & \text{else} \end{cases}$$

α ensures that the sum of all probabilities equals 1

Katz Backoff (2)

Example

Seen: $p(\text{two bones})$
Unseen: $p(\text{some bones})$

Known: $p(bones|two)$
Unknown: $p(bones|some)$

Known: $p(bones)$!

The higher-order frequency is related to the lower-order frequency.
Important:
$p(bones|some) < p(bones|two)$

Interpolation (Jelinek-Mercer)

- Another method to combine probability distributions
- Weighted average

$$p(w_i|w_k, ..., w_{i-1}) = \sum_{j=0}^{k} \lambda_j \; p(w_i|w_{k+j}, ..., w_{i-1})$$

About λ_j

- The sum of all λ_j is 1
- Use held-out data to determine the best set of values

Absolute Discounting (1)

- Subtract a fixed number from all frequencies
- Uniformly assign the sum of these numbers to new events

$$p(w_1, ..., w_n) = \begin{cases} \dfrac{f(w_1, ..., w_n) - \delta}{N} & \text{if } f(w_1, ..., w_n) - \delta > 0 \\[2ex] \dfrac{(A - n_0)\,\delta}{n_0\, N} & \text{else} \end{cases}$$

n_n: the number of events that occurred n times

Absolute Discounting (2)

How to determine δ?

$$\delta = \frac{n_1}{n_1 + 2n_2}$$

Problems

- Still wrong results for events with frequency 1

$\longrightarrow$ Solution: Different discounts for n_1, n_2, n_{3+}

Witten-Bell Smoothing (1)

- Instance of Interpolation
- λ_k: Probability of using the higher-order model

$$p_{WB}(w_i|w_k, ..., w_{i-1}) =$$
$$\lambda_k \; p_{ML}(w_i|w_k, ..., w_{i-1}) + (1 - \lambda_k) \; p_{WB}(w_i|w_{k+1}, ..., w_{i-1})$$

Witten-Bell Smoothing (2)

- Thus: $1 - \lambda_k$: Probability of using the lower-order model
- Idea: This is related to the number of different words that follow the *history* $w_k, ..., w_{i-1}$

$N_{1+}(w_k, ..., w_{i-1}, \bullet) =$ No. of different words that follow $w_k, ..., w_{i-1}$

$$1 - \lambda_k = \frac{N_{1+}(w_k, ..., w_{i-1}, \bullet)}{N_{1+}(w_l, ..., w_{i-1}, \bullet) + \sum_{w_i} f(w_k, ...w_i)}$$

Kneser-Ney Smoothing

- Extension of Absolute Discounting:
  New way to build the lower-order distribution

Problem Case

"San Francisco":

- *Francisco* is common but only occurs after *San*
- Absolute Discounting will yield high $p(\mathrm{Francisco}|\textit{new word})$
- Not intended!

- Unigram probability should not be proportional to its frequency but to the **number of different words it follows**

Modified Kneser-Ney Smoothing (1)

Idea

Similarities to Absolute Discounting

- Interpolation with lower-order model
- Discounts

$$p_{kn}(w_1|w_k, ..., w_{i-1}) =$$

$$\frac{f(w_k, ..., w_i) - D(f)}{\sum_{w_i} f(w_k, ..., w_i)} + \gamma(w_k, ..., w_{i-1})\, p_{kn}(w_i|w_{k+1}, ..., w_{i-1})$$

Modified Kneser-Ney Smoothing (2)

Discount function D(f)

- Different discounts for frequencies 1, 2 and 3+

$$D(f) = \begin{cases} 0 & \text{if } n = 0 \\ D_1 = 1 - 2Y\frac{n_2}{n_1} & \text{if } n = 1 \\ D_2 = 2 - 3Y\frac{n_3}{n_2} & \text{if } n = 2 \\ D_{3+} = 3 - 4Y\frac{n_4}{n_3} & \text{if } n \geq 3 \end{cases}$$

Modified Kneser-Ney Smoothing (3)

Gamma

Ensures that the distribution sums to 1.

$$\gamma(w_k, ..., w_{i-1}) =$$

$$\frac{D_1 N_1(w_k, ..., w_{i-1}) + D_2 N_2(w_k, ..., w_{i-1}) + D_3 N_3(w_k, ..., w_{i-1})}{\sum_{w_i} f(w_k, ..., w_i)}$$

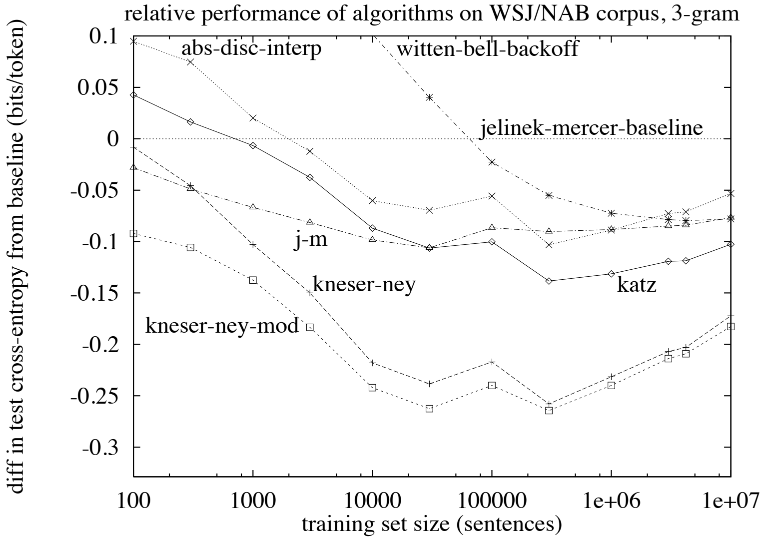
Evaluation

Cross-Entropy Evaluation

- **Cross-entropy (H):** The average number of bits needed to identify an event from a set of possibilities
- In our case: How good is the approximation our smoothing method produces?
- Calculate cross-entropy for each method on test data

$$H(p, q) = -\sum_{x} p(x) log\ q(x)$$

Evaluation


relative performance of algorithms on WSJ/NAB corpus, 3-gram
diff in test cross-entropy from baseline (bits/token)
training set size (sentences)
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
100
1000
10000
100000
1e+06
1e+07
abs-disc-interp
witten-bell-backoff
jelinek-mercer-baseline
j-m
kneser-ney
katz
kneser-ney-mod

References

1 Stanley F. Chen and Joshua Goodman
An empirical study of smoothing techniques for language modeling
Proceedings of the 34th annual meeting of the Association for Computational Linguistics, 1996

2 Joshua Goodman
The State of the Art in Language Model Smoothing
http://research.microsoft.com/~joshuago/lm-tutorial-public.ppt

3 Bill MacCartney
NLP Lunch Tutorial: Smoothing
http://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf

4 Christopher Manning and Hinrich Schütze
Foundations of statistical natural language processing
MIT Press, 1999

5 Helmut Schmid
Statistische Methoden in der Maschinellen Sprachverarbeitung
http://www.ims.uni-stuttgart.de/~schmid/ParsingII/current/snlp-folien.pdf