A Brief History of Time
(if all you care about is smoothing for n-gram models)

Stanley Chen

Carnegie Mellon

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

February 18, 1999

The Problem: Speech Recognition

goal: find

$$
\begin{aligned}
\arg\max_{T} \mathrm{Pr}(\text{text } T \mid \text{acous. sig. } A) &= \arg\max_{T} \frac{\mathrm{Pr}(A \mid T)\mathrm{Pr}(T)}{\mathrm{Pr}(A)} \\
&= \arg\max_{T} \mathrm{Pr}(A \mid T)\mathrm{Pr}(T)
\end{aligned}
$$

The Problem: Language Modeling

goal: estimate

$$\begin{aligned}
\text{Pr(text } T) &= \text{Pr(words } w_1 \cdots w_l) \\
&= \Pr(w_1) \times \Pr(w_2|w_1) \times \cdots \times \Pr(w_l|w_1 \cdots w_{l-1}) \\
&\quad \times \Pr(\text{END}|w_1 \cdots w_l) \\
&= \prod_{i=1}^{l+1} \Pr(w_i|w_1 \cdots w_{i-1}) \\
&\approx \prod_{i=1}^{l+1} \Pr(w_i|w_{i-2} w_{i-1})
\end{aligned}$$

Smoothing: Who Needs It?

Estimating $\Pr(w_i|w_{i-2}w_{i-1}) \Rightarrow \Pr_{\text{tri}}(w_i)$

- maximum likelihood (ML)

$$p_{\text{tri}}^{\text{ML}}(w) = \frac{c_{\text{tri}}(w)}{c_{\text{tri}}(\bullet)}$$

- zero probabilities are bad for business
- basic observation
  - probabilities of zero-count words are too low
  - probabilities of nonzero-count words are too high
- smoothing!

Smoothing, The Early Years: Big Bang – birth of Elvis

+1 smoothing (1920)

- add one to each count

$$p^{+1}_{\text{tri}}(w) = \frac{c_{\text{tri}}(w) + 1}{c_{\text{tri}}(\bullet) + |\text{vocab}|}$$

- example: $w \in \{\text{sun rises}, \text{sun explodes}\}$

$$p^{+1}(\text{sun explodes tomorrow}) = \frac{0+1}{10000+2} \approx 0.0001$$

- not good for large vocab, little data, *e.g.*, 10000 word vocab, training set of 100 words

$$p^{+1}_{\text{uni}}(\textit{the}) = \frac{5+1}{100+10000} \approx 0.0006$$

Smoothing, The Early Years: Big Bang – birth of Elvis

$+\delta$ smoothing (1920?)

- add $\delta < 1$ to each count

$$p_{\text{tri}}^{+\delta}(w) = \frac{c_{\text{tri}}(w) + \delta}{c_{\text{tri}}(\bullet) + \delta \times |\text{vocab}|}$$

- woo-hoo!

Smoothing, The Middle Ages: birth of Elvis – release of Terminator 2

(Good)-Turing estimate (1953)

- all words with same count should receive same probability
- count mass of words with $r+1$ counts should be assigned to words with r counts

$$r^{\mathsf{GT}} = \text{corrected count of word with } r \text{ counts}$$

$$[\#\ r\text{-count wds}] \times r^{\mathsf{GT}} = [\#\ (r+1)\text{-count wds}] \times (r+1)$$

Good-Turing Estimate (1953)

derivation — leave-one out estimation

- for each word token in data
- call that word the test set, remaining data the training set
- see how often word in test set has r counts in training set
- this will happen every time word left out has $r+1$ counts in original data

$$\text{(corrected mass of } r\text{-cnt wds)} = [\#\ (r+1)\text{-cnt wds}] \times (r+1)$$

Good-Turing Estimate (1953)

- accurate when plenty of data
- example: meeting new people at speech lunch
  - seen 100×20 people (tokens) at lunch = 2000 data points
  - seen 20 people (types) exactly once

$$\text{(corrected zero-count mass)} = (\#\ \text{1-count wds}) \times 1$$

  - prob that I've never seen a given person at speech lunch before $\approx \frac{20}{2000} = 0.01$
- example: estimate out-of-vocabulary rate

Smoothing, The Middle Ages: birth of Elvis – release of Terminator 2

Katz smoothing (1987)

- Good-Turing estimate with *back-off*
  - use bigram distribution to help estimate probability of words with zero count

$$p^{\text{Katz}}_{\text{tri}}(w) = \begin{cases} \frac{c^{\text{GT}}_{\text{tri}}(w)}{c_{\text{tri}}(\bullet)} & \text{if } c_{\text{tri}}(w) > 0 \\ (1-\lambda) \times p^{\text{Katz}}_{\text{bi}}(w) & \text{if } c_{\text{tri}}(w) = 0 \end{cases}$$

- back-off recursively

Smoothing, The Middle Ages: birth of Elvis – release of Terminator 2

Jelinek-Mercer smoothing (1980), a.k.a. linear interpolation

- don't back-off, interpolate
  - use bigram distribution to help estimate probability of words with *nonzero* count
- linear discounting

$$p_{\text{tri}}^{\text{JM}}(w) = \begin{cases} \frac{\lambda c_{\text{tri}}(w)}{c_{\text{tri}}(\bullet)} + (1-\lambda) \times p_{\text{bi}}^{\text{JM}}(w) & \text{if } c_{\text{tri}}(w) > 0 \\ (1-\lambda) \times p_{\text{bi}}^{\text{JM}}(w) & \text{if } c_{\text{tri}}(w) = 0 \end{cases}$$

$$= \lambda \times \frac{c_{\text{tri}}(w)}{c_{\text{tri}}(\bullet)} + (1-\lambda) \times p_{\text{bi}}^{\text{JM}}(w)$$

Smoothing, The Middle Ages: birth of Elvis – release of Terminator 2

Witten-Bell smoothing (1990), a.k.a. Moffat smoothing

- instance of linear interpolation/Jelinek-Mercer smoothing
- parameterless

$$p_{\mathrm{tri}}^{\mathrm{WB}}(w) = \lambda \times \frac{c_{\mathrm{tri}}(w)}{c_{\mathrm{tri}}(\bullet)} + (1-\lambda) \times p_{\mathrm{bi}}^{\mathrm{WB}}(w)$$

$$1-\lambda = \frac{(\#\ \text{wd types following hist})}{(\#\ \text{wd types following hist}) + c_{\mathrm{tri}}(\bullet)}$$

Smoothing, The Middle Ages: birth of Elvis – release of Terminator 2

absolute discounting (Ney, Essen, and Kneser; 1994)

$$p_{\mathsf{tri}}^{\mathsf{abs}}(w) = \begin{cases} \frac{c_{\mathsf{tri}}(w) - D}{c_{\mathsf{tri}}(\bullet)} & \text{if } c_{\mathsf{tri}}(w) > 0 \\ (1-\lambda) \times p_{\mathsf{bi}}^{\mathsf{abs}}(w) & \text{if } c_{\mathsf{tri}}(w) = 0 \end{cases}$$

Interlude: The Story So Far

	shadowing		discounting		
	b-off	interp	GT	lin	abs
W-B		$\surd$		$\surd$	
abs. disc.	$\surd$				$\surd$
Katz	$\surd$		$\surd$		
J-M		$\surd$		$\surd$	

Interlude: Performance

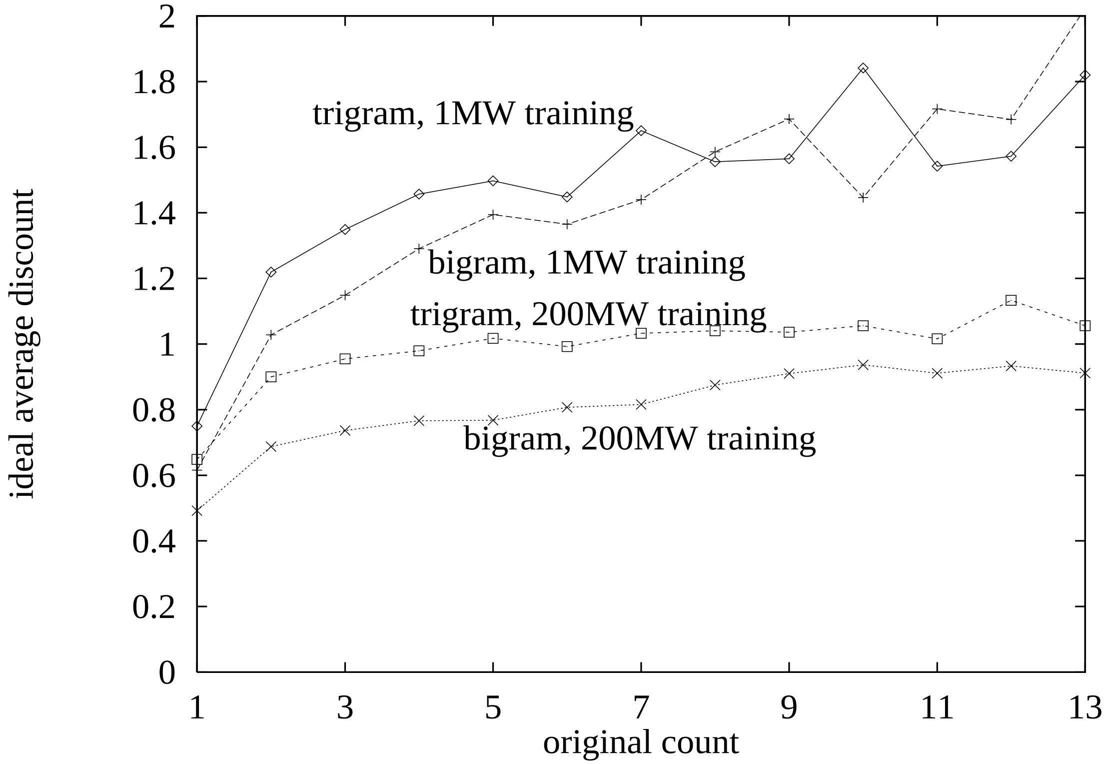
(CHEN and Goodman; 1996)

- loser smoothing algorithms
  - $+1$ smoothing, $+\delta$ smoothing
- mediocre smoothing algorithms
  - Witten-Bell smoothing, absolute discounting
- OK smoothing algorithms ($\sim$0–7% PP reduction, $\sim$0.0–0.5% absolute WER reduction over mediocre)
  - Katz smoothing, Jelinek-Mercer smoothing

Interlude: Lessons

- interpolation is better than back-off
  - bigram model helps estimate trigram probs even for words with nonzero trigram counts
- types of discounting
  - for back-off models, Good-Turing is good
  - for interpolated models, absolute discounting works better than linear discounting

Discounting



Carnegie Mellon

Smoothing, Modern Era: release of Terminator 2 — now

old way

- what process are we going to use to smooth probabilities?

new way

- what properties do we want our smoothed model to have?

Smoothing, Modern Era: release of Terminator 2 — now

constraining marginals

- given trigram model $p_{\text{tri}}(w_i|w_{i-2}w_{i-1})$, calculate how often this model predicts a given word should occur:

$$
\begin{aligned}
p_{\text{tri}}(\textit{the}) &= \sum_{w_{i-2}w_{i-1}} p_{\text{tri}}(w_{i-2}w_{i-1}\textit{the}) \\
&= \sum_{w_{i-2}w_{i-1}} p(w_{i-2}w_{i-1})\, p_{\text{tri}}(\textit{the} \mid w_{i-2}w_{i-1})
\end{aligned}
$$

- this should equal (or be close to) frequency of *the* in training data
- marginals of smoothed model should match marginals of training data

Smoothing, Modern Era: release of Terminator 2 — now

Kneser-Ney smoothing (1995)

- absolute discounting

$$p_{\text{tri}}^{\text{KN}}(w) = \begin{cases} \frac{c_{\text{tri}}(w) - D}{c_{\text{tri}}(\bullet)} & \text{if } c_{\text{tri}}(w) > 0 \\ (1-\lambda) \times p_{\text{bi}}^{\text{KN}}(w) & \text{if } c_{\text{tri}}(w) = 0 \end{cases}$$

- $p_{\text{bi}}^{\text{KN}}(w)$ is chosen so that unigram and bigram marginals of training data are met exactly

Kneser-Ney smoothing (1995)

- bigram probabilities $p^{\text{KN}}_{\text{bi}}(w)$
  - *not* proportional to how often bigram occurs
  - proportional to how many word types that bigram follows
- intuition: backoff to bigram only when trigram count not present; bigram prob should be proportional to prob of bigram occurring in novel trigram
- example: unigram probability of *Francisco*
  - *San Francisco* occurs in training data, like, a million times
  - regular model: set p_{uni}(*Francisco*) to be high, will be assigned high prob after a lot of words
  - Kneser-Ney: set p_{uni}(*Francisco*) to be low, will be assigned low prob after all words except *San*, in which case bigram prob is used

Smoothing, Modern Era: release of Terminator 2 — now

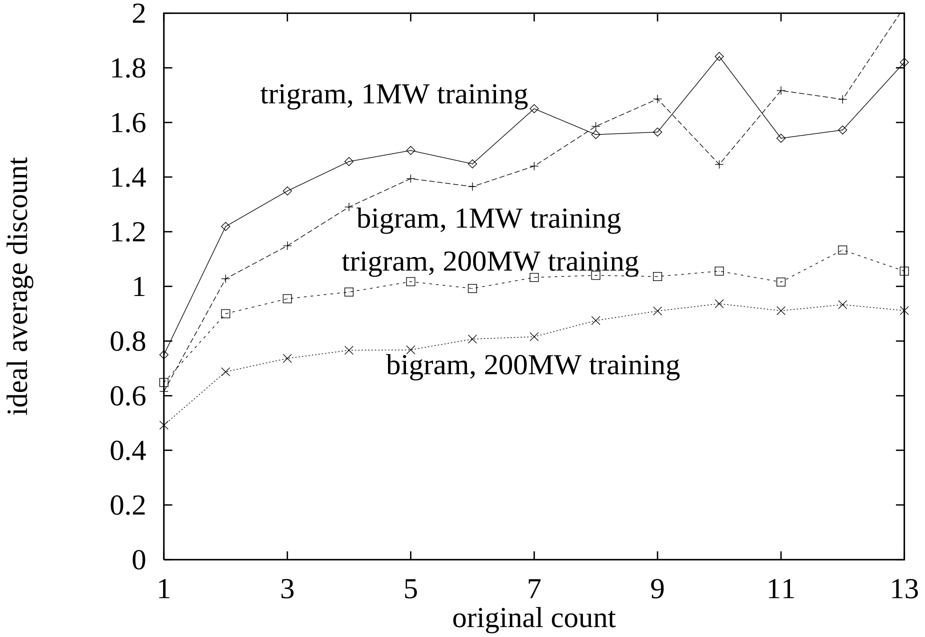
Performance gain (CHEN and Goodman; 1998)

- mediocre smoothing algorithms
  - Witten-Bell smoothing, absolute discounting
- OK smoothing algorithms (~0–7% PP reduction, ~0.0–0.5% absolute WER reduction over mediocre)
  - Katz smoothing, Jelinek-Mercer smoothing
- new, high-tech smoothing algorithms (~10–15% PP reduction, ~0.5% absolute WER reduction over OK)
  - Kneser-Ney smoothing, modified Kneser-Ney smoothing, Lafferty smoothing

Smoothing, Modern Era: release of Terminator 2 — now

modified Kneser-Ney smoothing (CHEN and Goodman; 1998)

- interpolated instead of back-off
- three discounts D_1, D_2, D_{3+} instead of one



Smoothing, Modern Era: release of Terminator 2 — now

Lafferty smoothing (CHEN and Rosenfeld; 1999)

- maximum entropy n-gram model — maximum likelihood exponential model

$$p_{\text{tri}}^{\text{exp}}(w) = \begin{cases} \frac{Z_{\text{bi}}}{Z_{\text{tri}}} \times e^{\lambda_w} \times p_{\text{bi}}^{\text{exp}}(w) & \text{if } c_{\text{tri}}(w) > 0 \\ \frac{Z_{\text{bi}}}{Z_{\text{tri}}} \times p_{\text{bi}}^{\text{exp}}(w) & \text{if } c_{\text{tri}}(w) = 0 \end{cases}$$

- Gaussian prior on λ_w to favor more uniform models
- marginal constraints are satisfied
- interpolated; *logarithmic* discounting
- real slow to build

Algorithm Summary

	shadowing		discounting					
	b-off	interp	GT	lin	abs	3-abs	log	marg
W-B		$\surd$		$\surd$				
abs. disc.	$\surd$				$\surd$			
Katz	$\surd$		$\surd$					
J-M		$\surd$		$\surd$				
K-N	$\surd$				$\surd$			$\surd$
mod K-N		$\surd$				$\surd$		$\surd$
ME		$\surd$					$\surd$	$\surd$

Conclusions

- ask what, not how
- use marginal constraints to design smoothing algorithms
- interpolated is better than backoff
  - bigram model helps estimate trigram probs even for words with trigram counts
- logarithmic discounting