# Language Models for Speech Recognition

Arthur Kantor

Department of Computer Science
University of Illinois - Urbana Champaign

September 13, 2010

# Outline

# What is a Language Model?

- Language model: a distribution over possible word strings
- If we have a sequence $w_1, ..., w_l$ of $l$ words, the language model is the distribution

$$p(w_1, ..., w_l) = \prod_{i=1}^{l} p(w_i | w_1, ..., w_{i-1}) \approx \prod_{i=1}^{l} p(w_i | w_{i-n+1}, ..., w_{i-1}) = \prod_{i=1}^{l} p(w_i | h) \quad (1)$$

- Equation 1 assumes that words are conditionally independent, given they are separated by a long enough history $h$ of $n - 1$ words.
- $n$ is the order of the n-gram language model.
- If $i - n + 1 < 1$, we can simply pad the beginning of the text with a special <BEGINNING> token.

# Evaluating the model quality

## Language Model quality is measured with Cross-Entropy

$$H_{pq}(w|h) = -\sum_{w,h} q(w,h) \log p(w|h)$$

- $p(w,h)$ and $q(w,h)$ are the distributions over word sequences estimated from the training and development data, respectively.
- We can write

$$H_{pq}(w|h) = H_p(w|h) + D_{KL}(p(w|h)||q(w|h))$$

so we are minimizing the sum of conditional entropy of training distribution and the conditional KL-divergence between the training and development distributions.

## Relationship of cross-entropy and Word Error Rate

- Difficult to describe analytically
- Empirically, The WER and model perplexity are related by the power law [Klakow, Peters 2002]:

$$\log WER = a + bH_{pq}(w|h)$$

  where $a$ and $b$ are constants that depend on the data and the quality of the acoustic model.
- Relative WER improvement is proportional to decrease of cross entropy of the LM.
- On planned speech (Broadcast News corpus, DARPA 1996 and 1997 competitions), the relative WER improvement is 12%-20% for each bit decrease of cross-entropy

# The maximum likelihood language model

Let $C(x)$ be the number of times the word string $x$ is seen in the training corpus.

**Maximum Likelihood estimate**

$$p_{ML}(w|h) = \frac{C(h,w)}{C(h)}$$

That was easy, right?

However
$p_{ML}(w|h)$ is a poor estimate when the training data is sparse.

# The training data is sparse

Fisher corpus:

- 57036 words, $1.85 \times 10^{14}$ possible trigrams
- 21.9 million tokens cover at most 0.0000118% of trigrams

If training data sparsity is not a problem, you can make a higher-order LM with lower cross-entropy, and training data sparsity again becomes a problem.

# What's the problem?

- $p_{ML}(w|h)$ underestimates the probability of n-grams never seen in the training data.
  - Never-seen ngrams account for a large probability mass of the true n-gram distribution.
- $p_{ML}(w|h) = 0$ precludes the recognizer from hypothesizing $w|h$ even if the acoustic model fits perfectly.

**Solution: Smoothing**

Raise the probability of low-probability n-grams and lower the probability of high-probability n-grams

# Outline

# An old problem.

- Laplace considered smoothing in his "Will the sun rise tomorrow?" question.
- Sun not rising is a rare event, unobserved in the known past. What is the probability $p(\mathit{Sun\ not\ rising\ tomorrow})$?
- According to prior knowledge, two outcomes are possible: pretend they happened and add them as pseudocounts to the observed counts. $p(x) = \frac{C(x)+1}{C(x)+2}$
- Generalizing to $|V|$ objects so that $w \in V$, and allowing pseudocounts smaller than 1, we get

**additive smoothing**

$$p_{add}(w|h) = \frac{C(h,w)+\alpha}{C(h)+\alpha|V|} \qquad 0 < \alpha \leq 1$$

Simple, but yields poor models (discounts too much).

## Count-of-counts Definition

- Group n-grams by the number of times an n-gram was seen in the training data.
- Define $n_r$ be the total number of n-grams each of which has been $r$ times (count of counts)
- Define the event of encountering *any* n-gram that has been seen $r$ times in the training data as $M_r$.
- According to the ML distribution, the probability of seeing event $M_r$ is

$$p_{ML}(M_r) = \frac{n_r r}{N}$$

where $N$ is the total number of n-grams: $N = \sum_{r=1}^{\infty} n_r$

# Main Idea

Probability mass assigned to all n-grams observed $r$ times in training data is spread equally among the n-grams seen $r - 1$ times.
Good-Turing distribution $p_{GT}$ is defined to satisfy

$$p_{GT}(M_r) = p_{ML}(M_{r+1})`$$

The probability mass assigned to all unseen n-grams is $p_{GT}(M_0) = p_{ML}(M_1)$.

(see the board)

# Definition

Good-Turing smoothing adjusts the counts $r$ seen in the training data

$$
\begin{aligned}
p_{GT}(M_r) &= p_{ML}(M_{r+1}) \\
\frac{n_r r^*}{N} &= \frac{n_{r+1}(r+1)}{N} \\
r^* &= \frac{n_{r+1}}{n_r}(r+1)
\end{aligned}
$$

## Good-Turing Smoothing

$$p_{GT}(w_i, h) = \frac{r^*(h, w_i)}{N}$$

Definition requires that $n_r > 0$. In practice only n-grams with $r(h, w_i) < k$ are smoothed, and $p_{GT}(h, w_i)$ is re-normalized.

## Why this particular discount $r^*$?

- $r^*$ is the solution to

$$\frac{r^*}{N} \approx E(p_i | C(w_i) = r)$$

  where $w_i$ is one of $s$ n-grams, with true frequency $p_i$.
- $E(p_i | C(w_i) = r)$ is the expected probability for some n-gram $w_i$, where we don't know the identity of $w_i$ but we know it was observed $C(w_i)$ times in the training data.

# Katz Smoothing

- In GT smoothing, the discounted probability mass $p_{ML}(M_1)$ is uniformly spread among unseen n-grams.
- In Katz smoothing, the discounted probability mass is spread among unseen n-grams weighted by (n-1)-order model $p(w_i|w_{i-n+2}, ...w_{i-1})$

# Definition

## Katz/Good-Turing smoothing

$$p_{katz}(w_i|h) = \begin{cases} d_r(h, w_i)\dfrac{C(h, w_i)}{C(h)} & \text{if } r > 0 \\ \alpha_h p_{katz}(w_i|w_{i-n+2}, \ldots, w_{i-1}) & \text{if } r = 0 \end{cases}$$

- For Good-Turing discounting,

$$d_r(h, w_i) \approx \frac{r^*(h, w_i)}{r(h, w_i)}$$

- $\alpha_h$ is chosen so that the probability mass to be allocated by the $(n-1)$-gram model is equal to the probability mass discounted from the $r > 0$ n-grams.

# Computing $\alpha_h$

**Katz/Good-Turing smoothing**

$$p_{katz}(w_i|h) = \begin{cases} d_r(h, w_i)\dfrac{C(h, w_i)}{C(h)} & \text{if } r > 0 \\ \alpha_h p_{katz}(w_i|w_{i-n+2}, \ldots, w_{i-1}) & \text{if } r = 0 \end{cases}$$

- Let

$$p_{katz}(M_0|h) = 1 - \sum_{\{w_i : C(h,w_i) > 0\}} d_{r(h,w_i)} \frac{C(h, w_i)}{C(h)}$$

  be the probability mass allocated to the event of encountering any n-gram unseen in the training data given a history $h$.
- $\alpha_h$ must satisfy

$$\alpha_h \sum_{\{w_i : C(h,w_i) = 0\}} p_{katz}(w_i|w_{i-n+2}, \ldots, w_{i-1}) = p_{katz}(M_0|h)$$

# Motivation

- Consider a bigram LM where the phrase “SAN FRANCISCO” is frequent, and “FRANCISCO” is almost always preceded by the word “SAN”.
- The unigram probability of “FRANCISCO” will be high, and with $p_{katz}(w_i|h)$ it will have a high probability following some unseen history, say “APPLE FRANCISCO”.
- But this is probably wrong, because “FRANCISCO” should only follow the one history “SAN”.

Kneser-Ney smoothing addresses this situation.

# Definition

Let $N_{1+}(h, \bullet)$ be the number of unique n-grams seen in the training one or more times with history $h$.

**Kneser-Ney smoothing**

$$p_{KN}(w_i|h) = \frac{\max\{C(h, w_i) - D, 0\}}{\sum_{w_i} C(h, w_i)} + \frac{D}{\sum_{w_i} C(h, w_i)} N_{1+}(h, \bullet) p_{KN}(w_i | w_{i-n+2}, \ldots, w_{i-1})$$

$D < 1$ is the absolute discount subtracted from all n-grams seen in the training data.

# Derivation of $p_{KN}(w_i | w_{i-n+2}, \ldots, w_{i-1})$

The original objective for Kneser-Ney smoothing was for the smoothed distribution marginalized over the left-most word in the history to equal the marginalized ML distribution:

$$\sum_{w_{i-n+1}} p_{KN}(w_{i-n+1}, \ldots, w_i) = p_{ML}(w_{i-n+2}, \ldots, w_i)$$

Combining the above with $p_{kn}(w_i | h)$ form yields

$$p_{KN}(w_i | w_{i-n+2}, \ldots, w_{i-1}) = \frac{N_{1+}(\bullet, w_{i-n+2}, \ldots, w_i)}{\sum_{w_i} N_{1+}(\bullet, w_{i-n+2}, \ldots, w_i)}$$

which itself could be KN-smoothed.

A little non-obvious: see SRILM ngram-discount man page for details.

(n-1)-order model allocates a bigger portion of the discount to words having more left histories: “APPLE FRANCISCO” is unlikely.

# Some comments

## Kneser-Ney smoothing

$$p_{KN}(w_i|h) = \frac{\max\{C(h, w_i) - D, 0\}}{\sum_{w_i} C(h, w_i)}$$

$$+ \frac{D}{\sum_{w_i} C(h, w_i)} N_{1+}(h, \bullet) p_{KN}(w_i|w_{i-n+2}, \dots, w_{i-1})$$

$$p_{KN}(w_i|w_{i-n+2}, \dots, w_{i-1}) = \frac{N_{1+}(\bullet, w_{i-n+2}, \dots, w_i)}{\sum_{w_i} N_{1+}(\bullet, w_{i-n+2}, \dots, w_i)}$$

- (n-1)-order model allocates a bigger portion of the discount to words having more left histories: “APPLE FRANCISCO” is unlikely.
- (n-1)-order is not estimating the true distribution $p(w_i|w_{i-n+2}, \dots, w_{i-1})$!

$$p_{KN}(w_i|w_{i-n+2}, \dots, w_{i-1}) \neq p(w_i|w_{i-n+2}, \dots, w_{i-1})$$

## How well do they work?

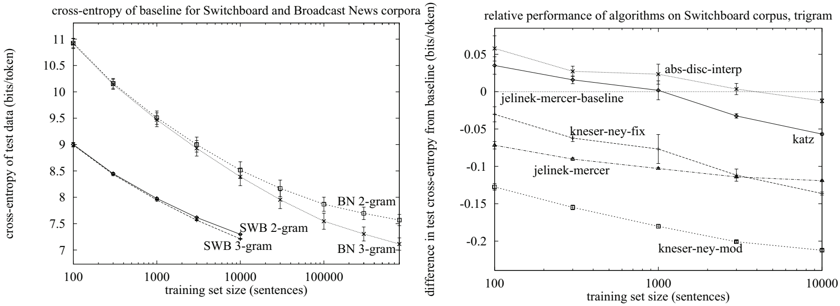


Figure: Baseline LM performance. From previous slide: “On planned speech (Broadcast News corpus, DARPA 1996 and 1997 competitions), the relative WER improvement is 12%-20% for each bit decrease of cross-entropy.”

# Outline

## Another problem

Language Models can be large - too many parameters for an ASR recognizer to handle efficiently

**Solution: Pruning**

Remove parameters from an LM by removing explicitly represented n-grams, so they can be approximated by lower-order n-grams

The goal is to remove the n-grams in such a way that minimizes the damage (in terms of cross-entropy) to the LM

# Low count cut off pruning

Drop n-grams that are seen less than $k$ times.

- Simple
- Only coarse control of the model size
- For a given model size, lower cross-entropies can be achieved with other pruning methods.

# Entropy-based Pruning [Stolcke 2000]

Idea: Prune the least damaging n-gram, one at a time, until the model is the desired size.
Least Damaging: The n-gram, whose removal minimize the KL-divergence between the original LM $p(w_i|h)$ and the pruned model $p'(w_i|h)$.

$$D_{KL}(p(w_i|h)||p'(w_i|h)) = \sum_{w_i,h} p(w_i, h) \left(\log p(w_i|h) - \log p'(w_i|h)\right)$$

# Entropy-based Pruning advantages

- Advantages
  - Can prune an arbitrary number of n-grams.
  - Raises the entropy less than removing low-count n-grams.
  - Can efficiently update the n-gram probabilities and back-offs and only needs the information in the LM being pruned, so there is no need to keep around the original n-gram counts.
- Results
  - In [Stolcke 2000], authors show that entropy pruning can reduce the size of the LM by a factor of four without increasing the WER of their recognizer, and raising the LM cross-entropy only slightly.
  - Entropy-pruning an n-gram model down to the size of an $(n-1)$-gram model yields a lower cross-entropy model than just using an unpruned $(n-1)$-gram model.

# Efficient computation of $D_{KL}(p(w_i|h)||p'(w_i|h))$

Removing an n-gram $h, w_i$ from $p(w_i|h)$ changes it only through estimates involving history $h$, and no other histories. Therefore we can write

$$
\begin{aligned}
D_{KL}(p(w_i|h)||p'(w_i|h)) &= \sum_{w_i} p(w_i, h) \left(\log p(w_i|h) - \log p'(w_i|h)\right) \\
&= p(h) \sum_{w_i} p(w_i|h) \left(\log p(w_i|h) - \log p'(w_i|h)\right)
\end{aligned}
$$

- $p(h)$ is computed using only the existing model
  - important for understanding interaction between pruning and smoothing

# Outline

## Entropy-based pruning and Kneser-Ney smoothing

Remember pruning criterion:

$$D_{KL}(p(w_i|h)||p'(w_i|h)) = p(h)\sum_{w_i} p(w_i|h)\left(\log p(w_i|h) - \log p'(w_i|h)\right)$$

$p(h)$ is calculated from the smoothed model *model*:

$$\begin{aligned} p(h) &= p(w_{i-n+1}, ..., w_{i-1}) \\ &= p_{model}(w_{i-n+1}) \prod_{j=1}^{n-2} p_{model}(w_{i-j}|w_{i-n+1}, ..., w_{i-j-1}) \end{aligned}$$

- Makes sense for Katz/Good-Turing smoothing.
- for Kneser-Ney smoothing the lower order models are not an estimate for the true n-gram distribution.
  - $p(h)$ calculated from a Kneser-Ney smoothed LM will be a poor estimate of the true distribution.
  - $D_{KL}(p(w_i|h)||p'(w_i|h))$ will be inaccurate.

## Correcting $p(h)$ is not enough.

- Estimating $p(h)$ correctly (say from maximum likelihood or Katz/Good-Turing smoothed models) helps, but still worse than good-turing smoothing + entropy pruning [Chelba, Brants, Neveitt, Xu, 2010].
- Simply removing n-grams from higher-order Kneser-Ney smoothed models introduces problems.
  - (n-1)-order models are not designed to model n-grams which occur in the upper-level models.
- Aggressively pruning the vocabulary hurts KN-smoothed LMs for the same reasons.
  - Words with low token counts are removed $\Rightarrow$ their n-grams are also pruned from the n-order model.
  - (n-1)-models are forced to model (n-1)-grams that were excluded from their training.

## Example

3-gram LM with 10,000 word vocabulary, trained on 80% of the Fisher Corpus.

Table: Effect of pruning on the cross-entropy (bits) of smoothed models.

| | GT-smoothing | KN-smoothing |
|---|---|---|
| no pruning | 6.722 | 6.686 |
| pruning | 6.809 | 6.819 |

see `http://mickey.ifp.uiuc.edu/wiki/Fisher_Language_Model` for experiments showing these trends

## Conclusions

- Knesser-Ney smoothing creates monolithic language models
  - Knesser-Ney smoothing outerperforms Good-Turing smoothing if nothing else is done to it
  - Lower order n-grams cannot be used independently of the highest order n-grams
  - Lower order n-grams are a bad estimate of the true distribution $p(w|h)$
  - vocabulary pruning and entropy-based pruning ruins a Knesser-Ney smoothed model
- Good-Turing smoothing of n-order LMs contains good (n-1)-order LMs within it
  - Lower order n-grams can be used independently of the highest order n-grams
  - Lower order n-grams are a good estimate of the true distribution $p(w|h)$
  - vocabulary pruning and entropy-based pruning works OK with a Good-Turing smoothed model

# Outline

# References


- Chen and Goodman, 1998 "An Empirical Study of Smoothing Techniques for Language Modeling"
- Stolcke, 2000 "Entropy-based pruning of backoff language models"
- http://www.speech.sri.com/projects/srilm/manpages/ngram-discount.7.html
- Chelba, Brants, Neveitt, Xu, 2010 "Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing"
- Klakow and Peters 2002 "Testing the correlation of word error rate and perplexity"
- http://mickey.ifp.uiuc.edu/wiki/Fisher_Language_Model