

---

# Advanced Natural Language Processing

## Lecture 5

### Language Modeling (II)

Philipp Koehn

30 September 2011



## Back-Off

- In given corpus, we may never observe
  - Scottish beer drinkers
  - Scottish beer eaters
- Both have count 0
  - our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - beer drinkers
  - beer eaters

# Interpolation

- Higher and lower order n-gram models have different strengths and weaknesses
  - high-order n-grams are sensitive to more context, but have sparse counts
  - low-order n-grams consider only very limited context, but have robust counts
- Combine them

$$\begin{aligned} p_I(w_3|w_1, w_2) = & \lambda_1 p_1(w_3) \\ & \times \lambda_2 p_2(w_3|w_2) \\ & \times \lambda_3 p_3(w_3|w_1, w_2) \end{aligned}$$

## Recursive Interpolation

- We can trust some histories  $w_{i-n+1}, \dots, w_{i-1}$  more than others
- Condition interpolation weights on history:  $\lambda_{w_{i-n+1}, \dots, w_{i-1}}$
- Recursive definition of interpolation

$$p_n^I(w_i | w_{i-n+1}, \dots, w_{i-1}) = \lambda_{w_{i-n+1}, \dots, w_{i-1}} p_n(w_i | w_{i-n+1}, \dots, w_{i-1}) + \\ + (1 - \lambda_{w_{i-n+1}, \dots, w_{i-1}}) p_{n-1}^I(w_i | w_{i-n+2}, \dots, w_{i-1})$$

## Back-Off

- Trust the highest order language model that contains n-gram

$$p_n^{BO}(w_i | w_{i-n+1}, \dots, w_{i-1}) = \begin{cases} \alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1}) & \text{if } \text{count}_n(w_{i-n+1}, \dots, w_i) > 0 \\ d_n(w_{i-n+1}, \dots, w_{i-1}) p_{n-1}^{BO}(w_i | w_{i-n+2}, \dots, w_{i-1}) & \text{else} \end{cases}$$

- Requires
  - adjusted prediction model  $\alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1})$
  - discounting function  $d_n(w_1, \dots, w_{n-1})$

## Back-Off with Good-Turing Smoothing

- Previously, we computed n-gram probabilities based on relative frequency

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

- Good Turing smoothing adjusts counts  $c$  to expected counts  $c^*$

$$\text{count}^*(w_1, w_2) \leq \text{count}(w_1, w_2)$$

- We use these expected counts for the prediction model (but  $0^*$  remains  $0$ )

$$\alpha(w_2|w_1) = \frac{\text{count}^*(w_1, w_2)}{\text{count}(w_1)}$$

- This leaves probability mass for the discounting function

$$d_2(w_1) = 1 - \sum_{w_2} \alpha(w_2|w_1)$$

## Example

- Good Turing discounting is used for all positive counts

	count	$p$	GT count	$\alpha$
$p(\text{big} \text{a})$	3	$\frac{3}{7} = 0.43$	2.24	$\frac{2.24}{7} = 0.32$
$p(\text{house} \text{a})$	3	$\frac{3}{7} = 0.43$	2.24	$\frac{2.24}{7} = 0.32$
$p(\text{new} \text{a})$	1	$\frac{1}{7} = 0.14$	0.446	$\frac{0.446}{7} = 0.06$

- $1 - (0.32 + 0.32 + 0.06) = 0.30$  is left for back-off  $d_2(\text{a})$
- Note: actual values for  $d_2$  is slightly higher, since the predictions of the lower-order model to seen events at this level are not used.

## Diversity of Predicted Words

- Consider the bigram histories *spite* and *constant*
  - both occur 993 times in Europarl corpus
  - only 9 different words follow *spite*  
almost always followed by *of* (979 times), due to expression *in spite of*
  - 415 different words follow *constant*  
most frequent: *and* (42 times), *concern* (27 times), *pressure* (26 times),  
but huge tail of singletons: 268 different words
- More likely to see new bigram that starts with *constant* than *spite*
- Witten-Bell smoothing considers diversity of predicted words

# Witten-Bell Smoothing

- Recursive interpolation method
- Number of possible extensions of a history  $w_1, \dots, w_{n-1}$  in training data

$$N_{1+}(w_1, \dots, w_{n-1}, \bullet) = |\{w_n : c(w_1, \dots, w_{n-1}, w_n) > 0\}|$$

- Lambda parameters

$$1 - \lambda_{w_1, \dots, w_{n-1}} = \frac{N_{1+}(w_1, \dots, w_{n-1}, \bullet)}{N_{1+}(w_1, \dots, w_{n-1}, \bullet) + \sum_{w_n} c(w_1, \dots, w_{n-1}, w_n)}$$

## Witten-Bell Smoothing: Examples

Let us apply this to our two examples:

$$\begin{aligned} 1 - \lambda_{spite} &= \frac{N_{1+}(spite, \bullet)}{N_{1+}(spite, \bullet) + \sum_{w_n} c(spite, w_n)} \\ &= \frac{9}{9 + 993} = 0.00898 \end{aligned}$$

$$\begin{aligned} 1 - \lambda_{constant} &= \frac{N_{1+}(constant, \bullet)}{N_{1+}(constant, \bullet) + \sum_{w_n} c(constant, w_n)} \\ &= \frac{415}{415 + 993} = 0.29474 \end{aligned}$$

## Diversity of Histories

- Consider the word [York](#)
  - fairly frequent word in Europarl corpus, occurs 477 times
  - as frequent as [foods](#), [indicates](#) and [providers](#)
  - in unigram language model: a respectable probability
- However, it almost always directly follows [New](#) (473 times)
- Recall: unigram model only used, if the bigram model inconclusive
  - [York](#) unlikely second word in unseen bigram
  - in back-off unigram model, [York](#) should have low probability

## Kneser-Ney Smoothing

- Kneser-Ney smoothing takes diversity of histories into account
- Count of histories for a word

$$N_{1+}(\bullet w) = |\{w_i : c(w_i, w) > 0\}|$$

- Recall: maximum likelihood estimation of unigram language model

$$p_{ML}(w) = \frac{c(w)}{\sum_i c(w_i)}$$

- In Kneser-Ney smoothing, replace raw counts with count of histories

$$p_{KN}(w) = \frac{N_{1+}(\bullet w)}{\sum_{w_i} N_{1+}(w_i w)}$$

# Modified Kneser-Ney Smoothing

- Based on interpolation

$$p_n^{BO}(w_i | w_{i-n+1}, \dots, w_{i-1}) = \begin{cases} \alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1}) & \text{if } \text{count}_n(w_{i-n+1}, \dots, w_i) > 0 \\ d_n(w_{i-n+1}, \dots, w_{i-1}) p_{n-1}^{BO}(w_i | w_{i-n+2}, \dots, w_{i-1}) & \text{else} \end{cases}$$

- Requires
  - adjusted prediction model  $\alpha_n(w_i | w_{i-n+1}, \dots, w_{i-1})$
  - discounting function  $d_n(w_1, \dots, w_{n-1})$

## Formula for $\alpha$ for Highest Order N-Gram Model

- Absolute discounting: subtract a fixed  $D$  from all non-zero counts

$$\alpha(w_n|w_1, \dots, w_{n-1}) = \frac{c(w_1, \dots, w_n) - D}{\sum_w c(w_1, \dots, w_{n-1}, w)}$$

- Refinement: three different discount values

$$D(c) = \begin{cases} D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c \geq 3 \end{cases}$$

## Discount Parameters

- Optimal discounting parameters  $D_1, D_2, D_{3+}$  can be computed quite easily

$$Y = \frac{N_1}{N_1 + 2N_2}$$

$$D_1 = 1 - 2Y \frac{N_2}{N_1}$$

$$D_2 = 2 - 3Y \frac{N_3}{N_2}$$

$$D_{3+} = 3 - 4Y \frac{N_4}{N_3}$$

- Values  $N_c$  are the counts of n-grams with exactly count  $c$

## Formula for $d$ for Highest Order N-Gram Model

- Probability mass set aside from seen events

$$d(w_1, \dots, w_{n-1}) = \frac{\sum_{i \in \{1, 2, 3+\}} D_i N_i(w_1, \dots, w_{n-1} \bullet)}{\sum_{w_n} c(w_1, \dots, w_n)}$$

- $N_i$  for  $i \in \{1, 2, 3+\}$  are computed based on the count of extensions of a history  $w_1, \dots, w_{n-1}$  with count 1, 2, and 3 or more, respectively.
- Similar to Witten-Bell smoothing

## Formula for $\alpha$ for Lower Order N-Gram Models

- Recall: base on count of histories  $N_{1+}(\bullet w)$  in which word may appear, not raw counts.

$$\alpha(w_n|w_1, \dots, w_{n-1}) = \frac{N_{1+}(\bullet w_1, \dots, w_n) - D}{\sum_w N_{1+}(\bullet w_1, \dots, w_{n-1}, w)}$$

- Again, three different values for  $D$  ( $D_1$ ,  $D_2$ ,  $D_{3+}$ ), based on the count of the history  $w_1, \dots, w_{n-1}$

## Formula for $d$ for Lower Order N-Gram Models

- Probability mass set aside available for the  $d$  function

$$d(w_1, \dots, w_{n-1}) = \frac{\sum_{i \in \{1, 2, 3+\}} D_i N_i(w_1, \dots, w_{n-1} \bullet)}{\sum_{w_n} c(w_1, \dots, w_n)}$$

## Interpolated Back-Off

- Back-off models use only highest order n-gram
  - if sparse, not very reliable.
  - two different n-grams with same history occur once → same probability
  - one may be an outlier, the other under-represented in training
- To remedy this, always consider the lower-order back-off models
- Adapting the  $\alpha$  function into interpolated  $\alpha_I$  function by adding back-off

$$\alpha_I(w_n|w_1, \dots, w_{n-1}) = \alpha(w_n|w_1, \dots, w_{n-1}) \\ + d(w_1, \dots, w_{n-1}) p_I(w_n|w_2, \dots, w_{n-1})$$

- Note that  $d$  function needs to be adapted as well

# Evaluation

Evaluation of smoothing methods:

Perplexity for language models trained on the Europarl corpus

<b>Smoothing method</b>	<b>bigram</b>	<b>trigram</b>	<b>4-gram</b>
Good-Turing	96.2	62.9	59.9
Witten-Bell	97.1	63.8	60.4
Modified Kneser-Ney	95.4	61.6	58.6
Interpolated Modified Kneser-Ney	94.5	59.3	54.0

# Managing the Size of the Model

- Millions to billions of words are easy to get  
(trillions of English words available on the web)
  
- But: huge language models do not fit into RAM

## Number of Unique N-Grams

Number of unique n-grams in Europarl corpus  
29,501,088 tokens (words and punctuation)

Order	Unique n-grams	Singletons
unigram	86,700	33,447 (38.6%)
bigram	1,948,935	1,132,844 (58.1%)
trigram	8,092,798	6,022,286 (74.4%)
4-gram	15,303,847	13,081,621 (85.5%)
5-gram	19,882,175	18,324,577 (92.2%)

→ remove singletons of higher order n-grams

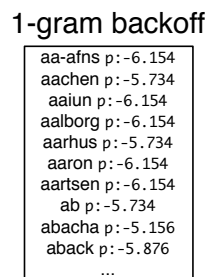
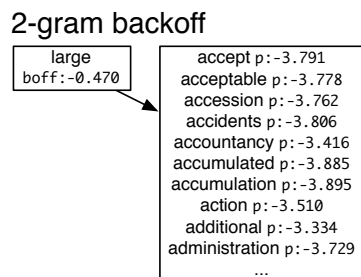
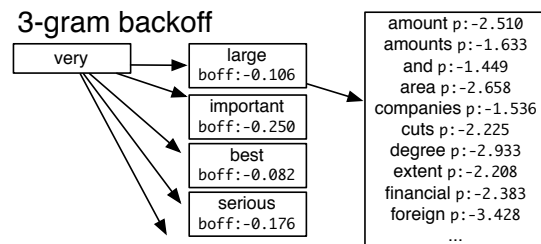
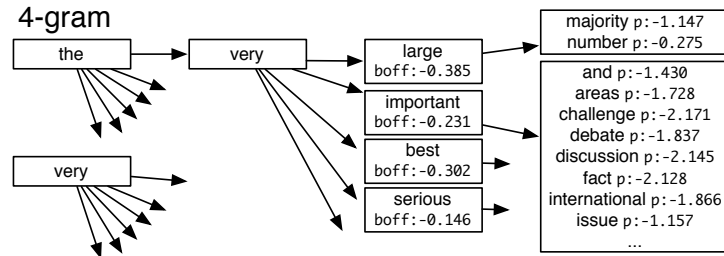
## Estimation on Disk

- Language models too large to *build*
- What needs to be stored in RAM?
  - maximum likelihood estimation

$$p(w_n | w_1, \dots, w_{n-1}) = \frac{\text{count}(w_1, \dots, w_n)}{\text{count}(w_1, \dots, w_{n-1})}$$

- can be done separately for each history  $w_1, \dots, w_{n-1}$
- Keep data on disk
  - extract all n-grams into files on-disk
  - sort by history on disk
  - only keep n-grams with shared history in RAM
- Smoothing techniques may require additional statistics

# Efficient Data Structures



- Need to store probabilities for
  - the very large majority
  - the very language number

- Both share history the very large

→ no need to store history twice

→ Trie

## Fewer Bits to Store Probabilities

- Index for words
  - two bytes allow a vocabulary of  $2^{16} = 65,536$  words, typically more needed
  - Huffman coding to use fewer bits for frequent words.
- Probabilities
  - typically stored in log format as floats (4 or 8 bytes)
  - quantization of probabilities to use even less memory, maybe just 4-8 bits

## Reducing Vocabulary Size

- For instance: each number is treated as a separate token
- Replace them with a number token NUM
  - but: we want our language model to prefer

$$p_{\text{LM}}(\text{I pay } 950.00 \text{ in May } 2007) > p_{\text{LM}}(\text{I pay } 2007 \text{ in May } 950.00)$$

- not possible with number token

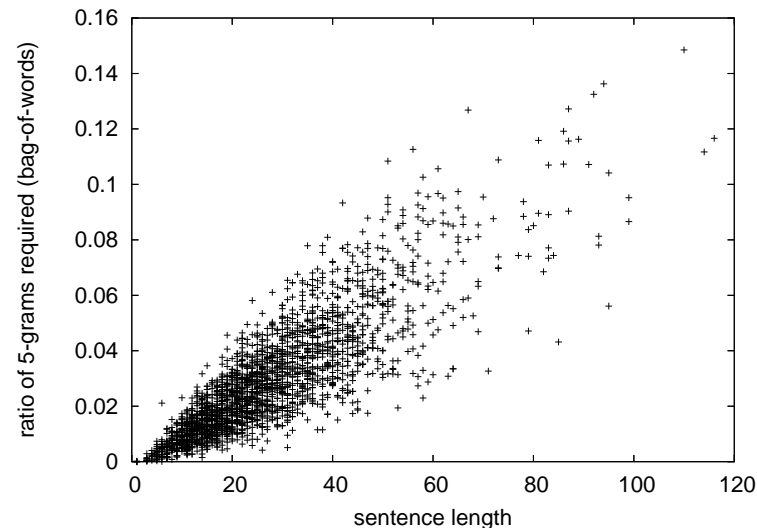
$$p_{\text{LM}}(\text{I pay NUM in May NUM}) = p_{\text{LM}}(\text{I pay NUM in May NUM})$$

- Replace each digit (with unique symbol, e.g., @ or 5), retain some distinctions

$$p_{\text{LM}}(\text{I pay } 555.55 \text{ in May } 5555) > p_{\text{LM}}(\text{I pay } 5555 \text{ in May } 555.55)$$

## Filtering Irrelevant N-Grams

- We use language model in decoding
  - we only produce English words in translation options
  - filter language model down to n-grams containing only those words
- Ratio of 5-grams needed to all 5-grams (by sentence length):



## Other Topics in Language Modeling

- Language modeling is still an active field of research
- There are many back-off and interpolation methods
- Skip n-gram models: back-off to  $p(w_n|w_{n-2})$
- Factored language models: back-off to word stems, part-of-speech tags
- Syntactic language models: using parse trees