- Trigram Language Models
 - · A trigram Language model consists of:
 - A finite set V
 - A parameter q(wlu,v) for each trigram u,v,w such that for w element V U {STOP}, and u,v element V U {*}
 - For any sentence $\underline{x_1 \dots x_n}$ where $\underline{x_i \in \mathcal{V}}$ for $\underline{i = 1 \dots (n-1)}$, and $\underline{x_n = \text{STOP}}$, the probability of the sentence under the trigram language model is

$$\underline{p(x_1 \dots x_n)} = \prod_{i=1}^n q(\underline{x_i} | \underline{x_{i-2}}, x_{i \neq 1})$$

where we define $x_0 = x_{-1} = *$.

Remaining estimation problem:

- Quite difficult to improve on and benefit of simplicity
- Trigram Estimation Problem

 $q(w_i \mid w_{i-2}, w_{i-1})$ For example: $q(\mathsf{laughs} \mid \mathsf{the}, \mathsf{dog})$ A natural estimate (the "maximum likelihood estimate"): $q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1})}$ $q(\mathsf{laughs} \mid \mathsf{the}, \mathsf{dog}) = \frac{\mathsf{Count}(\mathsf{the}, \mathsf{dog}, \mathsf{laughs})}{\mathsf{Count}(\mathsf{the}, \mathsf{dog})}$

- · Forms the starting point for the estimation methods
- Problem
 - Huge vocabulary size
 - If the trigram has been seen in the training data then the estimation will be zero
 - Leads to estimates being unrealistically low or undefined

- Evaluating a Language Model: Perplexity
 - ▶ We have some test data, m sentences

$$s_1, s_2, s_3, \ldots, s_m$$

We could look at the probability under our model $\prod_{i=1}^m p(s_i)$. Or more conveniently, the log probability





- The higher the log probability the better our model is at evaluating these test sentences
 - In fact the usual evaluation measure is perplexity



and M is the total number of words in the test data

- The average log probability word by word normalized to the length of the test examples
- Lower quantities of perplexity the better or model is to the fit of our test examples
- Some Intuition about Perplexity



$$q(\underline{w}|u,v) = \frac{1}{N}$$

for all $w \in \mathcal{Y} \cup \{STOP\}$, for all $u, v \in \mathcal{V} \cup \{*\}$.

► Easy to calculate the perplexity in this case:

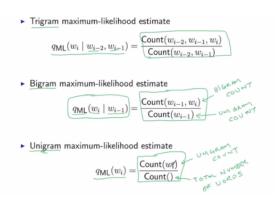
Perplexity is a measure of effective "branching factor"

- Language Model assigns the uniform distribution over all the words in the vocabulary
- Typical Values of Perplexity
 - \blacktriangleright Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}|=50,000$
 - ▶ A trigram model: $p(x_1 \dots x_n) = \prod_{i=1}^n q(x_i | \underline{x_{i-2}, x_{i \not = 1}})$. Perplexity = 74.
 - A bigram model: $p(x_1 \dots x_n) = \prod_{i=1}^n q(x_i|x_{i-1})$. Perplexity = 137
 - A unigram model: $p(x_1 \dots x_n) = \prod_{i=1}^n \boxed{g(x_i)}$. Perplexity = 955

- · Improvements from unigram model
- Estimation Techniques
- Sparse Data Problems

A natural estimate (the "maximum likelihood estimate"):
$$q(\underline{w_i} \mid \underline{w_{i-2}}, \underline{w_{i-1}}) = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1})} = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1})} = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1})} = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)} = \frac{\mathsf{Count}(w_{i-2}, w_i)}{\mathsf{Count}(w_{i-2}, w_i)} = \frac{\mathsf{Count}(w_{i-2}, w_i)}{\mathsf{$$

- · If the counts equal zero lead to many problems and the estimations are undefined
- The Bias-Variance Trade-OFF



- · Trigram estimate has the benefit that it conditions on a lot of context
 - It has a low bias
 - Reasonably probability of wi given the context
 - Has the problem that many of the counts will be equal to zero
 - Need a large dataset
- Unigram
 - Ignores the context
 - Counts converge very quickly
- Linear Interpolation

Take our estimate
$$q(w_i \mid w_{i-2}, w_{i-1})$$
 to be
$$\frac{q(w_i \mid w_{i-2}, w_{i-1})}{+\lambda_2 \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1})} + \lambda_2 \times q_{\text{ML}}(w_i \mid w_{i-1}, w_{i-1})}{+\lambda_3 \times q_{\text{ML}}(w_i \mid w_{i-1})} + \lambda_3 \times q_{\text{ML}}(w_i)}$$
 where $\lambda_1 + \lambda_2 + \lambda_3 = \underline{1}$, and $\lambda_i \geq 0$ for all i .

$$\lambda_1 = \lambda_2 = \lambda_3 = \underline{1}$$

$$\lambda_2 = \lambda_3 = \underline{1}$$

$$\lambda_3 = \underline{1}$$

$$\lambda_4 = \lambda_2 = \lambda_3 = \underline{1}$$

$$\lambda_4 = \lambda_3 = \underline{1}$$

$$\lambda_5 = \lambda_5 = \underline{1}$$

$$\lambda_5 =$$

- Lambda values control the weights of the estimates in the model
- Estimator is sensitive to the previous two words
 - It is robust in that it incorporates information from the more robust estimates from the bigram and unigram level
- Linear Interpolation (continued)
 - Our estimate correctly defines a distribution (define V' = V U {STOP})

Our estimate correctly defines a distribution (define
$$\mathcal{V}' = \mathcal{V} \cup \{\text{STOP}\}$$
):
$$\sum_{w \in \mathcal{V}'} q(w \mid u, v)$$

$$= \sum_{w \in \mathcal{V}'} [\lambda_1 \times q_{\mathsf{ML}}(w \mid u, v) + \lambda_2 \times q_{\mathsf{ML}}(w \mid v) + \lambda_3 \times q_{\mathsf{ML}}(w)]$$

$$= \lambda_1 \sum_{w} q_{\mathsf{ML}}(w \mid u, v) + \lambda_2 \sum_{w} q_{\mathsf{ML}}(w \mid v) + \lambda_3 \sum_{w} q_{\mathsf{ML}}(w)$$

$$= \lambda_1 + \lambda_2 + \lambda_3$$

$$= 1$$
 (Can show also that $q(w \mid u, v) \geq 0$ for all $w \in \mathcal{V}'$)

- How to estimate Lambda values?
 - Hold out part of training set as "validation" data
 - Define c'(w1,w2,w3) to be the number of times the trigram (w1,w2,w3) is seen in validation set
 - Trying to find the values of lambda that minimize the perplexity of the data and hence fit the data best as possible

- Allowing the lambdas to vary
 - ► Take a function II that partitions histories e.g.,

$$\Pi(\underline{w_{i=2}},\underline{w_{i\!\not=\!1}}) = \left\{ \begin{array}{ll} 1 & \text{If } \mathsf{Count}(w_{i-1},w_{i-2}) = 0 \\ 2 & \text{If } 1 \leq \mathsf{Count}(w_{i-1},w_{i-2}) \leq 2 \\ 3 & \text{If } 3 \leq \mathsf{Count}(w_{i-1},w_{i-2}) \leq 5 \\ 4 & \text{Otherwise} \end{array} \right.$$

▶ Introduce a dependence of the λ 's on the partition:

$$\begin{split} q(w_i \mid w_{i-2}, w_{i-1}) &= \quad \lambda_1^{\Pi(w_{i-2}, w_{i-1})} \times q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ &+ \lambda_2^{\Pi(w_{i-2}, w_{i-1})} \times q_{\mathsf{ML}}(w_i \mid w_{i-1}) \\ &+ \lambda_3^{\Pi(w_{i-2}, w_{i-1})} \times q_{\mathsf{ML}}(w_i) \end{split}$$
 where $\lambda_1^{\Pi(w_{i-2}, w_{i-1})} + \lambda_2^{\Pi(w_{i-2}, w_{i-1})} + \lambda_3^{\Pi(w_{i-2}, w_{i-1})} = 1$, and $\lambda_i^{\Pi(w_{i-2}, w_{i-1})} \geq 0$ for all i .

- · Partition is chosen by hand
- The lambdas vary depending upon which partition the bigram falls into
- Discounting Methods
- ► Say we've seen the following counts:

x	Count(x)	$q_{ML}(w_i \mid w_{i-1})$
the	48	
		45/10
the, dog	15	15/48
the, woman	11	11/48
the, man	10	10/48
the, park	5	5/48
the, job	2	2/48
the, telescope	1	1/48
the, manual	1	1/48
the, afternoon	1	1/48
the, country	1	1/48
the, street	1	1/48

- ► The maximum-likelihood estimates are high (particularly for low count items)
- The estimates are systematically high for probability of the X followed by "the"
- Now define "discounted" counts, Count'(x) = Count(x) .5
- New estimates

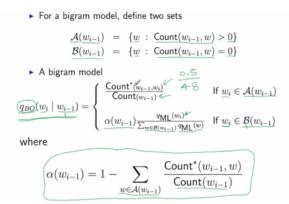
	x	Count(x)	$Count^*(x)$	Count(the)
	the	48		
١	the, dog	15	14.5	14.5/48
	the, woman	_11_	10.5	10.5/48
П	the, man	10	9.5	9.5/48
l	the, park	5	4.5	4.5/48
\	the, job	2	1.5	1.5/48
\	the, telescope	1	0.5	0.5/48
1	the, manual	1	0.5	0.5/48
1	the, afternoon	1	0.5	0.5/48
1	the, country	1	0.5	0.5/48
	the, street	61	0.5	0.5/48

- Essentially lowered the estimates through the discounting methods
- · Discounted sums to less 1
- We now have some "missing probability mass":

$$\boxed{\underline{\alpha(w_{i-1})} = \underline{1} - \sum_{\mathscr{B}} \frac{\mathsf{Count}^*(w_{i-1}, w)}{\mathsf{Count}(w_{i-1})}}$$

e.g., in our example, $\alpha(the)=10\times0.5/48=5/48$

- Katz Back-Off Models (Bigrams)



- Summary

- Three steps in deriving the language model probabilities
 - Expand p(w1....wn) using Chain rule
 - Make Markov Independence Assumptions
 - p(wilw1w2...wi-1) = p(wilwi-2wi-1)
 - second order markov assumptions
 - Smooth the estimates using low order counts
 - · Linear interpolation or discounting method
- Other methods used to improve language models:
 - Topic or long range features
 - Syntactic models