- Trigram Language Models
  - A trigram Language model consists of:
    - A finite set V
    - A parameter q(w|u,v) for each trigram u,v,w such that for w element V U {STOP}, and u,v element V U {*}

▸ For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = \text{STOP}$, the probability of the sentence under the trigram language model is

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

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

For the sentence

the dog barks STOP

we would have

$$\begin{aligned} p(\text{the dog barks STOP}) = \; & q(\text{the}|*, *) \\ & \times q(\text{dog}|*, \text{the}) \\ & \times q(\text{barks}|\text{the, dog}) \\ & \times q(\text{STOP}|\text{dog, barks}) \end{aligned}$$

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

Remaining estimation problem:

$$q(w_i \mid w_{i-2}, w_{i-1})$$

For example:

$$q(\text{laughs} \mid \text{the, dog})$$

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

  - 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*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

$q(\text{the}|*,*)$
$\times\, q(\text{dog}|*,\text{the})$
$\times \ldots$

  • The higher the log probability the better our model is at evaluating these test sentences

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

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

- Say we have a vocabulary $\mathcal{V}$, and $N = |\mathcal{V}| + 1$ and model that predicts

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

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

- Easy to calculate the perplexity in this case:

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \log \frac{1}{N}$$

$$\Rightarrow \quad \text{Perplexity} = N$$

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

- Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$
- A trigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$. Perplexity $= 74$
- A bigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-1})$. Perplexity $= 137$
- A unigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i)$. Perplexity $= 955$

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

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

TRIGRAM COUNT

BIGRAM COUNT

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count}(\text{the, dog, laughs})}{\text{Count}(\text{the, dog})}$$

Say our vocabulary size is $N = |\mathcal{V}|$, then there are N^3 parameters in the model.

e.g., $N = 20,000 \Rightarrow 20,000^3 = \boxed{8 \times 10^{12}}$ parameters

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

- Trigram maximum-likelihood estimate

$$q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) = \boxed{\frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}}$$

- Bigram maximum-likelihood estimate

$$\boxed{q_{\text{ML}}(w_i \mid w_{i-1})} = \boxed{\frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}}$$

BIGRAM COUNT

UNIGRAM COUNT

- Unigram maximum-likelihood estimate

$$q_{\text{ML}}(w_i) = \boxed{\frac{\text{Count}(w_i)}{\text{Count}()}}$$

UNIGRAM COUNT

TOTAL NUMBER OF WORDS

- 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

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \; & \lambda_1 \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\text{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3 \times q_{\text{ML}}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i.

$$\boxed{\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}}$$

$$\begin{aligned} & q(\text{laughs} \mid \text{the, dog}) \\ & = \frac{1}{3} \times q_{\text{ML}}(\text{laughs} \mid \text{the, dog}) \\ & + \frac{1}{3} \times q_{\text{ML}}(\text{laughs} \mid \text{dog}) \\ & + \frac{1}{3} \times q_{\text{ML}}(\text{laughs}) \end{aligned}$$

- 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 $\mathcal{V}' = \mathcal{V} \cup \{\text{STOP}\}$):

$$\sum_{w \in \mathcal{V}'} q(w \mid u, v)$$
$$= \sum_{w \in \mathcal{V}'} [\lambda_1 \times q_{\text{ML}}(w \mid u, v) + \lambda_2 \times q_{\text{ML}}(w \mid v) + \lambda_3 \times q_{\text{ML}}(w)]$$
$$= \lambda_1 \underbrace{\boxed{\sum_w q_{\text{ML}}(w \mid u, v)}}_{=1} + \lambda_2 \underbrace{\boxed{\sum_w q_{\text{ML}}(w \mid v)}}_{=1} + \lambda_3 \underbrace{\boxed{\sum_w q_{\text{ML}}(w)}}_{=1}$$
$$= \boxed{\lambda_1 + \lambda_2 + \lambda_3}$$
$$= 1$$

(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

- Choose $\lambda_1, \lambda_2, \lambda_3$ to maximize:

$$\boxed{L(\lambda_1, \lambda_2, \lambda_3)} = \boxed{\sum_{w_1, w_2, w_3} c'(w_1, w_2, w_3) \boxed{\log q(w_3 \mid w_1, w_2)}}$$

such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i, and where

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \; & \lambda_1 \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\text{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3 \times q_{\text{ML}}(w_i) \end{aligned}$$

- Allowing the lambdas to vary

- Take a function Π that partitions histories e.g.,

$$\Pi(w_{i-2}, w_{i-1}) = \begin{cases} 1 & \text{If Count}(w_{i-1}, w_{i-2}) = 0 \\ 2 & \text{If } 1 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 2 \\ 3 & \text{If } 3 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 5 \\ 4 & \text{Otherwise} \end{cases}$$

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

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \; & \lambda_1^{\Pi(w_{i-2}, w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2^{\Pi(w_{i-2}, w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3^{\Pi(w_{i-2}, w_{i-1})} \times q_{\text{ML}}(w_i) \end{aligned}$$

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)	$q_{\text{ML}}(w_i \mid w_{i-1})$
the	48	
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)	$\frac{\text{Count}^*(x)}{\text{Count(the)}}$
the	48		
the, dog	15	14.5	14.5/48
the, woman	11	10.5	10.5/48
the, man	10	9.5	9.5/48
the, park	5	4.5	4.5/48
the, job	2	1.5	1.5/48
the, telescope	1	0.5	0.5/48
the, manual	1	0.5	0.5/48
the, afternoon	1	0.5	0.5/48
the, country	1	0.5	0.5/48
the, street	1	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":

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

e.g., in our example, $\alpha(the) = 10 \times 0.5/48 = 5/48$

- Katz Back-Off Models (Bigrams)

  - For a bigram model, define two sets

$$\mathcal{A}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

  - A bigram model

$$q_{BO}(w_i \mid w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} \quad \frac{0.5}{48} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q_{ML}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

- Summary
  - Three steps in deriving the language model probabilities
    - Expand p(w1….wn) using Chain rule
    - Make Markov Independence Assumptions
      - p(wi|w1w2…wi-1) = p(wi|wi-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