Chapter 5:
Backing-Off Language Models
5.1. Unknown words and unseen unigrams
Example „Crime and Punishment“

- Training corpus: 11 000 tokens
- Test corpus:
  - 3000 tokens
  - 108 of them not seen in training corpus

- Usually training corpus defines vocabulary
- Unseen words in test corpus are hence not part of the vocabulary

OOV-words: “out-of-vocabulary” words
Example „Crime and Punishment“

- Training corpus: 11 000 tokens
- Test corpus:
  - 3000 tokens
  - 108 of them not seen in training corpus

Definition:

\[
\text{OOV - rate} = \frac{\# \text{unseen words in test corpus}}{\# \text{of tokens in test corpus}}
\]
OOV-Rate on “Crime and Punishment”

- OOV-Rate = 108/3000 = 0.036 = 3.6%
How to defined a vocabulary

• Count the frequency of each word in a corpus
• Use the most frequent N words for the vocabulary
• Sometimes software supports only a maximum value of N=65536 for memory efficiency
• Example: how to construct a vocabulary
• Determine OOV-rate
• Change size of vocabulary
OOV-Rate vs. Vocabulary

$$\text{OOV - Rate} \propto \frac{1}{\text{Size of Vocabulary}}$$
Corresponding Problem with M-Gram

- Size of vocabulary: $N$
- Number of sequences of length $M$ with $N$ words:
  $$N^M$$
- Numerical example: $N=64000$ and $M=3$
  - $N^M = 3 \times 10^{14}$
  - No realistic possibility to observe all trigrams in a training corpus
5.2 Maximum Likelihood Estimate of Probabilities
From Perplexity to Likelihood

\[ PP = P(w_1...w_N)^{-1/N} \]

\[ = \exp\left( - \sum_{w,h} f(w, h) \log(P(w | h)) \right) \]

\[ - \log(PP) = \sum_{w,h} f(w, h) \log(P(w | h)) = N \sum_{w,h} N(w, h) \log(P(w | h)) \]

Define likelihood \[ F = \sum_{w,h} N(w, h) \log(P(w | h)) \]

Minimize perplexity \[ \Leftrightarrow \] Maximize likelihood
Maximum-Likelihood Estimation of Probabilities

Maximize

\[ F = \sum_{w,h} N(w,h) \log(P(w | h)) \]

Using the normalization constraint

\[ \sum_w P(w | h) = 1 \quad \text{for all } h \]

Modified likelihood using Lagrange multiplier

\[ F' = \sum_{w,h} N(w,h) \log(P(w | h)) + \sum_h \lambda(h) \left[ 1 - \sum_w P(w | h) \right] \]
Maximum-Likelihood Estimation of Probabilities (cont.)

\[ F' = \sum_{w,h} N(w, h) \log(P(w \mid h)) + \sum_h \lambda(h) \left[ 1 - \sum_w P(w \mid h) \right] \]

\[ \frac{\partial F'}{\partial P(w' \mid h')} = \frac{N(w', h')}{P(w' \mid h')} - \lambda(h') = 0 \quad \Rightarrow \quad P(w' \mid h') = \frac{N(w', h')}{\lambda(h')} \]

Constraint
\[ 1 = \sum_w P(w \mid h) = \sum_w \frac{N(w', h')}{\lambda(h')} = \frac{N(h')}{\lambda(h')} \quad \Rightarrow \quad \lambda(h') = N(h') \]

\[ P(w \mid h) = \frac{N(w, h)}{N(h)} \]

Maximum likelihood estimate results in relative frequencies
Issues with Maximum Likelihood Estimate

- Unseen events
  - vanishing probability estimate
  - this sequence can not be found in speech recognition
  - IR: is document does not cover all query terms, it can not be relevant

Need other modeling scheme
5.3 Absolute discounting and leaving-one-out Estimate of Probabilities
Back off language model

\[ P(w|h) = \begin{cases} 
\frac{N(w,h)}{N(h)} + & \text{für } N(w,h) > 0 \\
0 & \text{neglect} 
\end{cases} \]

Back off weight \( \alpha(h) \)
Back off distribution \( \beta(w|h) \) (normalized!)
Discouting parameter \( d \)
Backing-off language model (without animation)

\[
P(w | h) = \begin{cases} 
\frac{N(w, h) - d}{N(h)} + \alpha(h) \beta(w | h) & \text{für } N(w, h) > 0 \\
\alpha(h) \beta(w | h) & \text{für } N(w, h) = 0 
\end{cases}
\]

Backing-off weight \( \alpha(h) \)
Backoff distribution \( \beta(w | h) \) (normalized!)
Discounting parameter \( d \)
Calculating the backing-off weight $\alpha(h)$

Use normalization

\[
1 = \sum_w P(w \mid h) = \sum_{w, N(w, h) > 0} \frac{N(w, h) - d}{N(h)} + \sum_w \alpha(h) \beta(w \mid h)
\]

\[
\sum_{w, N(w, h) > 0} d \quad = 1 - \frac{\sum_{w, N(w, h) > 0} d}{N(h)} + \alpha(h)
\]

Solve for $\alpha(h)$

\[
\alpha(h) = \frac{\sum_{w, N(w, h) > 0} d}{N(h)} = \frac{d \sum_{w, N(w, h) > 0} 1}{N(h)} = \frac{d R(h)}{N(h)} \quad \text{with} \quad R(h) = \sum_{w, N(w, h) > 0} 1
\]
Calculating the backing-off weight $\alpha(h)$

Back off weight

$$\alpha(h) = \frac{d R(h)}{N(h)}$$

with

$$R(h) = \sum_{w:N(w,h)>0} 1$$
Storing the counts
(Example: Bigram v,w)

N, R

v_1

N(v_1), R(v_1)  N(v_2), R(v_2)  ...  N(v_R), R(v_R)

w_1

N(v_1 w_1)  N(v_1 w_2)  ...  N(v_1 w_{R(v_1)})

w_2

N(v_2 w_1)  N(v_2 w_2)  ...  N(v_2 w_{R(v_1)})

N(v_1, R(v_1))  N(v_2, R(v_2))  ...  N(v_R, R(v_R))

w_R(v_1)

N(v_1, w_{R(v_1)})

w_R(v_R)

N(v_R, w_{R(v_R)})
Examples

- Example of backing-off code
- Example of language model
• Example:
  • Numerically show the influence of the discounting parameter
Influence of Discounting Parameter
• How to get a good estimate for the discounting parameter without tuning it explicitly?
Closed form optimization solution for discounting parameter

- Idea of cross validation: split training data

Use for parameter optimization
Variants of cross validation

• Suppose you want to train a unigram language model:
  • What is the smallest possible cross validation corpus?
  • How do you avoid fluctuations?
Leaving-one-out

- Each unigram of the corpus can be the cross validation corpus
- Calculate average of all possible cross validation corpora
Change of counts for leaving one out

- \( w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ w_6 \ w_7 \ w_8 \ w_9 \ \ldots \ w_N \)

\[ N(w_6) \iff N(w_6) - 1 \]

\[ N \iff N - 1 \]
Leaving-one-out likelihood

\[ F_{LOO} = \sum_{n=1}^{N} \log P_{LOO}(w_n) = \sum_{w \in W} N(w) \log P_{LOO}(w) \]

\[ = \sum_{w \in W : N(w) > 1} N(w) \log \left( \frac{N(w) - 1 - d}{N - 1} + \frac{Rd}{N - 1} \beta \right) \]

\[ + \sum_{w \in W : N(w) = 1} \log \left( \frac{(R-1)d}{N - 1} \beta \right) \]

Calculate \( \frac{\partial F_{LOO}}{\partial d} = 0 \)
Maximizing Leaving-one-out Likelihood

Calculate \[ \frac{\partial F_{LOO}}{\partial d} = 0 \]

- Skip calculating the derivative
- Result

\[ 0 = \sum_{r=2}^{\infty} n_r \frac{R \beta - 1}{r - 1 - d + R \beta d} + \frac{n_1}{d} \]

\[ \Rightarrow \text{no closed form solution} \]
Iterative solution

Split off terms in \( n_1 \) and \( n_2 \)

Solve for \( d \) and use as an iterative equation

\[
d_{i+1} = \frac{n_1}{(1 - R\beta) \left( n_1 + 2n_2 + \sum_{r=3}^{\infty} \frac{n_r r(1-d_i(1-R\beta))}{r - 1 - d_i(1-R\beta)} \right)}
\]
Approximate solution

Just use first iteration and initialize with $d_0=0$

$$d_1 = \frac{n_1}{(1 - R\beta) \left( n_1 + 2n_2 + \sum_{r=3}^{\infty} \frac{n_r r}{r-1} \right)}$$
Approximate solution

Usually $R\beta < 1$
Ignore the sum (only a small change to discounting parameter)

$$d \approx \frac{n_1}{n_1 + 2n_2}$$

Works very well for all practical applications
Demonstrate Approximate Solution

• -> Test formula using switchboard bigram
Remember Zipf’s Law

Zipf (simplified):

\[ N(r) = \frac{N_0}{r^{\gamma}} \]
Estimate $n_1$ and $n_2$

- Length of the first step in the Zipf-Distribution

\[ N(r) = \frac{N_0}{r^\gamma} \quad \Rightarrow \quad r = \left( \frac{N_0}{N} \right)^{\frac{1}{\gamma}} \]

\[ \Rightarrow \quad n_1 = \left( \frac{N_0}{\frac{1}{2}} \right)^{\frac{1}{\gamma}} - \left( \frac{N_0}{\frac{3}{2}} \right)^{\frac{1}{\gamma}} \]

and \[ n_2 = \left( \frac{N_0}{\frac{3}{2}} \right)^{\frac{1}{\gamma}} - \left( \frac{N_0}{\frac{5}{2}} \right)^{\frac{1}{\gamma}} \]
Estimate Discounting Parameter

\[
d \approx \frac{n_1}{n_1 + 2n_2} = \frac{1}{1 + 2 \frac{n_2}{n_1}}
\]

\[
e = \frac{1}{1 + 2 \frac{(2/3)^{1/\gamma} - (2/5)^{1/\gamma}}{(2^{1/\gamma} - (2/3)^{1/\gamma})}}
\]

For \( \gamma = 1 \) : \( d \approx \frac{5}{7} \approx 0.71 \)

Estimate only gives degradation as compared to optimal numerical solution
Slope in Zipf vs. Discounting

The larger the slope, the smaller the discounting parameter
Zipf for Unigram and Bigram

Longer range models have larger discounting parameters
5.4 Other smoothing methods
Floor discounting ("add epsilon")

Add a little bit to each count

\[ P(w \mid h) = \frac{1}{Z} (N(w, h) + \varepsilon) \]

1/Z: Normalisation

\[ 1 = \sum_{w=1}^V P(w \mid h) = \frac{1}{Z} (N(h) + \varepsilon V) \quad \Rightarrow \quad Z = (N(h) + \varepsilon V) \]

\[ P(w \mid h) = \frac{N(w, h) + \varepsilon}{N(h) + \varepsilon V} \]
Linear Discounting

\[ P(w \mid h) = (1 - \varepsilon) \frac{N(w, h)}{N(h)} + \varepsilon \beta(w \mid h) \]

\( \varepsilon \): determines amount of smoothing
\( \beta(w \mid h) \): "backing-off distribution"

Variant: \( P(w \mid h) = (1 - \varepsilon(h)) \frac{N(w, h)}{N(h)} + \varepsilon(h) \beta(w \mid h) \)

Popular value: \( \varepsilon(h) = \frac{n_1(h)}{N(h)} \) with \( n_1(h) = \sum_{w : N(w, h) = 1} 1 \)

See example
Good-Turing Discounting

Adjusted counts:

\[ N^*(wh) = \begin{cases} 
[N(w, h) + 1] \frac{n_{N^*(wh) + 1}}{n_{N^*(wh)}} & \text{für } N(w, h) \leq 5 \\
N(wh) & \text{sonst}
\end{cases} \]

Model:

\[ P(w \mid h) = \frac{N^*(wh)}{N(h)} + \lambda(h) \beta(w \mid h) \]

„Backing-off weight from normalization:

\[ \lambda(h) = 1 - \frac{N^*(h)}{N(h)} \]
5.5 Kneser-Ney Smoothing
Kneser-Ney Smoothing

- Try to fine a dedicated backing-off distribution
- Two different variants
  - “Marginal constraint” backing off
  - Singleton backing-off
Idea

\[ P(w \mid h) = \begin{cases} \frac{N(w, h) - d}{N(h)} + \alpha(h) \beta(w \mid \hat{h}) & \text{für } N(w, h) > 0 \\ \alpha(h) \beta(w \mid \hat{h}) & \end{cases} \]

With \( \hat{h} \) being the history shortened by one word

\[ p(w \mid \hat{h}) = \sum_{h'} P(w \mid h) P(h' \mid \hat{h}) \quad (**) \]

with \( h' \) being the word removed from the history

\[ \implies \text{look for backing off distribution such that } (**) \text{ is satisfied} \]
Solution

- Direct calculation
- Solution

\[ \beta(w | \hat{h}) = \frac{N_+(w \hat{h})}{\sum_w N_+(w \hat{h})} \]

with

\[ N_+(w \hat{h}) = \sum_{h' : N(w'h') > 0} 1 \]

Marginal constraint backing-off
Alternative Determine Backing-Off Distribution using Leaving-One-Out

\[
F = F_0 + \sum_{h' h w : N(w h h') = 1} \log(\alpha(h) \beta(w | \hat{h})) \\
+ \sum_{\hat{h}} \lambda(\hat{h}) \left( 1 - \sum_{w} \beta(w | \hat{h}) \right)
\]

- \(F_0\) likelihood for events seen more than once
- \(\lambda(\hat{h})\) Lagrange multiplier
Solution for Leaving-One-Out Approach to Backing-Off Distribution

- Direct calculation of first derivative
- Solution

\[
\beta(w | \hat{h}) = \frac{N_1(w\hat{h})}{\sum_w N_1(w\hat{h})}
\]

with

\[
N_1(w\hat{h}) = \sum_{h':N(w\hat{h}h')=1} 1
\]
## Comparison Marginal-Constraint/Singelton Backing-Off

<table>
<thead>
<tr>
<th>Backing-off Modell</th>
<th>Bigram</th>
<th>Trigram</th>
<th>4-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rel. Frequencies</td>
<td>160.1</td>
<td>146.4</td>
<td>147.8</td>
</tr>
<tr>
<td>Marginal Constraint</td>
<td>154.8</td>
<td>138.3</td>
<td>137.9</td>
</tr>
<tr>
<td>Singelton BO</td>
<td>156.2</td>
<td>142.6</td>
<td>141.9</td>
</tr>
</tbody>
</table>
Getting smaller language models:
Pruning of trees
Count-Trees

Simple criterion: Remove all infrequent nodes
Improved Criterion: Change in Likelihood

\[
\Delta F_{\text{Leaf}} = N(w, h) \log P(w \mid h) - N(wh) \log \left( \alpha(h) \beta(w \mid \hat{h}) \right)
\]

\[
= N(w, h) \log \left( \frac{P(w \mid h)}{\alpha(h) \beta(w \mid \hat{h})} \right)
\]

Create measure for sub-tree(s) from measures from leaves
Effect of language model pruning
Summary

• OOV-words
• Maximum likelihood estimator
• Backing-off language models
• Discounting parameter
• Kneser-Ney Smoothing
• Pruning of trees