Data Intensive Linguistics — Lecture 4
Language Modeling (II): Smoothing and Back-Off

Philipp Koehn
19 January 2006

Evaluation of language models

- We want to evaluate the quality of language models
- A good language model gives a high probability to real English
- We measure this with cross entropy and perplexity

Entropy over sequences will depend highly on how long these sequences are. To have a more meaningful measure, we want to measure entropy per word, also called the entropy rate:
\[
\frac{1}{n} H(w_1, \ldots, w_n) = \frac{1}{n} \sum_{w_i \in L} p(W^n_i) \log p(W^n_i)
\]

To measure true entropy of a language \( L \), we need to consider sequences of infinite length
\[
H(L) = \lim_{n \to \infty} \frac{1}{n} H(w_1, \ldots, w_n)
= \lim_{n \to \infty} \frac{1}{n} \sum_{W^{n}_i \in L} p(W^n_i) \log p(W^n_i)
\]

Cross-entropy

- In practice, we do not have the real probability distribution \( p \) for the language \( L \), only a model \( m \) for it.
- We define cross-entropy (replacing \( p \) with \( m \)) as
\[
H(p, m) = \lim_{n \to \infty} -\frac{1}{n} \log m(W^n_i)
\]
- True entropy of a language is an upper bound from cross-entropy:
\[
H(p) \leq H(p, m)
\]
- Cross-entropy is useful measure how well the model fits the true distribution.

Language Modeling Example

- Training set:
  
  there is a big house
  i buy a house
  they buy the new house

- Model:

  \[
  \begin{align*}
  p(\text{big|a}) &= 0.5 & p(a \mid \text{here}) &= 1 & p(\text{big}|\text{the}) &= 1 \\
  p(\text{house|a}) &= 0.5 & p(buy|a) &= 1 & p(a \mid \text{buy}) &= 0.5 \\
  p(\text{new|the}) &= 1 & p(\text{house|big}) &= 1 & p(\text{the|buy}) &= 0.5 \\
  p(a) &= 1 & p(\text{house|new}) &= 1 & p(\text{the|buy} < s) &= 0.33
  \end{align*}
  \]

- Test sentence \( S \): they buy a big house

  \[
  p(S) = 0.333 \times \frac{1}{\log a} \times 0.5 \times \frac{1}{\log a} \times 0.5 \times \frac{1}{\log a} = 0.0033
  \]

Entropy rate of a language

- We want to us entropy and perplexity to measure how well a model explains the test data

- Recall entropy:
\[
H(p) = -\sum_x p(x) \log p(x)
\]

- Entropy over sequences \( w_1, \ldots, w_n \) from a language \( L \):
\[
H(w_1, \ldots, w_n) = -\sum_{W^n_i \in L} p(W^n_i) \log p(W^n_i)
\]

- This can be simplified (Shannon-McMillan-Breiman theorem) to:
\[
H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(W^n_i)
\]

- Intuitive explanation: If the sequence is infinite, we do not need to sum over all possible sequences, since the infinite sequence contains all sequences

Using cross-entropy

- In practice, we do not have an infinite sequence, but a limited test set. However, if the test set is large enough, its measured cross-entropy approximates the true cross-entropy.

  - Example: \( p(S) = 0.333 \times \frac{1}{\log a} \times 0.5 \times \frac{1}{\log a} \times 0.5 \times \frac{1}{\log a} = 0.0033 \)

  \[
  \begin{align*}
  H(p, m) &= -\frac{1}{5} \log p(S) \\
  &= -\frac{1}{5} \log 0.333 + \log \frac{1}{a} + \log 0.5 + \log \frac{1}{a} + \log \frac{1}{a} \\
  &= \frac{1}{5} \left(-1.086 + \frac{1}{\log a} + \frac{1}{\log a} + \frac{1}{\log a} + \frac{1}{\log a} \right) = 0.7173
  \end{align*}
  \]
Perplexity

- **Perplexity** is defined as
  \[ PP = 2^{H(p|m)} = 2^{H(m) - \sum_{i=1}^{n} \log m(w_i | p, m)} \]
- In our example, \( H(m, p) = 0.7173 \Rightarrow PP = 1.0441 \)
- Intuitively, perplexity is the average number of choices at each point (weighted by the model)
- Perplexity is the most common measure to evaluate language models

---

Add-one smoothing: results

Church and Gale (1991a) experiment: 22 million words training, 22 million words testing, from same domain (AP newswire), counts of bigrams:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Actual frequency in test</th>
<th>Expected frequency in test (add one)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000027</td>
<td>0.000032</td>
</tr>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.000274</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>0.000411</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>0.000648</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>0.000865</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>0.001022</td>
</tr>
</tbody>
</table>

We overestimate 0-count bigrams (0.000027 > 0.000027), but since there are so many, they use up so much probability mass that hardly any is left.

---

Deleted estimation

- Counts in training \( C_s(w_1, ..., w_n) \)
- Counts how often an ngram seen in training is seen in held-out training \( C_h(w_1, ..., w_n) \)
- Number of ngrams with training count \( r: N_r \)
- Total times ngrams of training count \( r \) seen in held-out data: \( T_r \)
- Hold-out estimator:
  \[ p_h(w_1, ..., w_n) = \frac{T_r}{N_rN} \text{ where count}(w_1, ..., w_n) = r \]

---

Deleted estimation: results

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Actual frequency in test</th>
<th>Expected frequency in test (Good Tusing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000027</td>
<td>0.000137</td>
</tr>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.395</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.24</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.23</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.22</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>4.22</td>
</tr>
</tbody>
</table>

- Still overestimates unseen bigrams (why?)

---

Recap from last lecture

- If we estimate probabilities solely from counts, we give probability 0 to unseen events (bigrams, trigrams, etc.)
- One attempt to address this was with add-one smoothing.

---

Using held-out data

- We know from the test data, how much probability mass should be assigned to certain counts.
- We can not use the test data for estimation, because that would be cheating.
- Divide up the training data: one half for count collection, one half for collecting frequencies in unseen text.
- Both halves can be switched and results combined to not lose out on training data.

---

Using both halves

- Both halves can be switched and results combined to not lose out on training data
  \[ p_h(w_1, ..., w_n) = \frac{T_r}{N(N_r + N_h)} \text{ where count}(w_1, ..., w_n) = r \]

---

Good-Turing discounting

- Method based on the assumption of binomial distribution of frequencies.
- Translate real counts \( r \) for words with adjusted counts \( r^* \):
  \[ r^* = (r + 1) \frac{E(N_{r+1})}{E(N_r)} \]
  \( N_r \) is the count of counts: number of words with frequency \( r \).
- The probability mass reserved for unseen events is \( E(N_r)/N \).
- For large \( r \) (where \( N_{r+1} \) is often 0), so various other methods can be applied (don’t adjust counts, curve fitting to linear regression). See Manning & Schütze for details.
Good-Turing discounting: results

- Almost perfect:

<table>
<thead>
<tr>
<th>Frequency in training</th>
<th>Actual frequency in test</th>
<th>Expected frequency in test (Good Turing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000027</td>
<td>0.000027</td>
</tr>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.23</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>4.22</td>
</tr>
</tbody>
</table>

Combing estimators

- We would like to use higher-order n-gram language models.
- … but there are many n-grams with count 0.
- Linear interpolation \( p_h \) of estimators \( p_n \) of different order \( n \):
  \[
p_h(w_n | w_{n-2}, w_{n-1}) = \lambda_1 p_1(w_n) + \lambda_2 p_2(w_n | w_{n-1}) + \lambda_3 p_3(w_n | w_{n-2}, w_{n-1})
\]
  \( \lambda_1 + \lambda_2 + \lambda_3 = 1 \)

General linear interpolation

- We can generalize interpolation and backoff:
  \[
p_h(w_n | w_{n-2}, w_{n-1}) = \lambda_1 p_1(w_n | w_{n-2}, w_{n-1}) p_1(w_n) + \lambda_2 p_2(w_n | w_{n-1}) p_2(w_n | w_{n-1}) + \lambda_3 p_3(w_n | w_{n-2}, w_{n-1}) p_3(w_n | w_{n-2}, w_{n-1})
\]
- How do we set the \( \lambda \)s?

Is smoothing enough?

- If two events (bigrams, trigrams) are both seen with the same frequency, they are given the same probability:

<table>
<thead>
<tr>
<th>n-gram</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>scottish beer is</td>
<td>0</td>
</tr>
<tr>
<td>scottish beer green</td>
<td>0</td>
</tr>
<tr>
<td>beer in</td>
<td>45</td>
</tr>
<tr>
<td>beer green</td>
<td>0</td>
</tr>
</tbody>
</table>
- If there is not sufficient evidence, we may want to back off to lower-order n-grams

Katz’s backing-off

- Another approach is to back off to lower order n-gram language models

\[
p_h(w_n | w_{n-2}, w_{n-1}) = \begin{cases} 
1 - d(w_{n-2}, w_{n-1}) & \text{if count}(w_{n-2}, w_{n-1}) > 0 \\
\alpha(w_{n-2}, w_{n-1}) p_h(w_n | w_{n-1}) & \text{otherwise}
\end{cases}
\]
- The weight \( \alpha(w_{n-2}, w_{n-1}) \) given to the backoff path has to be chosen appropriately. Because this gives probability mass to unseen events, the maximum likelihood estimate has to be discounted (by \( d(w_{n-2}, w_{n-1}) \))

Consideration for weights \( \lambda(w_{n-2}, w_{n-1}) \)

- Based on \( \text{count}(w_{n-2}, w_{n-1}) \): the more frequent the history, the higher \( \lambda \).
- Organize histories in bins with similar counts, and optimize the resulting few \( \lambda(bins(w_{n-2}, w_{n-1})) \) by optimizing perplexity on a limited development set.
- Also consider entropy of predictions:
  - both great deal and of that occur 178 times in a selection of novels by Jane Austen
  - of that is followed by 115 different words
  - great deal is followed by 36 different words, 36% of the time of follows

Other methods 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 \( p(w_n | w_{n-c}) \)
- Factored language models: back-off to word stems, part-of-speech tags
- Syntactic language models: using parse trees
- Language models trained on 200 billion words using 2 TB disk space

PK  DL  16 January 2006