Unit 1: Sequence Models

Language Models
### Python Review: Styles

- **do not write ...**
  
  - `for key in d.keys():`
  - `if d.has_key(key):`
  - `i = 0
    for x in a:
      ...
      i += 1`
  - `a[0:len(a) - i]`
  - `for line in sys.stdin.readlines():`
  - `for x in a:
    print x,`
  - `print
  - s = ""
    for i in range(lev):
      s += " "`
  - `print s`

- **when you can write ...**
  
  - `for key in d:`
  - `if key in d:`
  - `for i, x in enumerate(a):`
  - `a[:-i]`
  - `for line in sys.stdin:`
  - `print " ".join(map(str, a))`
  - `print " " * lev`
Noisy-Channel Model

WFSA  t...t  WFST  w...w
Noisy-Channel Model

\[ p(t...t) \]

**BEST PATH IS**

\[ \arg\max_{t...t} p(t...t | w...w) \]

Each try sequence scored by

\[ P(w...w | t...t) \]

compose

Each try sequence scored by

\[ P(t...t) \cdot P(w...w | t...t) \]
Applications of Noisy-Channel

| Application                          | Input          | Output         | $p(i)$                              | $p(o|i)$                     |
|--------------------------------------|----------------|----------------|-------------------------------------|------------------------------|
| Machine Translation                  | $L_1$ word sequences | $L_2$ word sequences | $p(L_1)$ in a language model        | translation model            |
| Optical Character Recognition (OCR)  | actual text    | text with mistakes | prob of language text               | model of OCR errors          |
| Part Of Speech (POS) tagging         | POS tag sequences | English words  | prob of POS sequences               |                             |
| Speech recognition                   | word sequences | speech signal   | prob of word sequences              | acoustic model               |

- spelling correction
- correct text
- text with mistakes
- prob. of language text
- noisy spelling

Wednesday, September 21, 2011
Noisy Channel Examples

Th qck brwn fx jmps vr th lzy dg.
Ths sntnc hs ll twnty sx lttrs n th lphbt.

I cnduo't bvleiee taht I culod aulaclty uesdtannrd waht I was rdnaieg. Unisg the icndeblire pweor of the hmuan mnid, aocdcrnig to rseecrah at Cmabrigde Uinervtisy, it dseno't mttaer in waht oderr the ltters in a wrod are, the olny irpoamtnt tihng is taht the frsit and lsat ltteer be in the rhgit pclae.

Therestcanbeatotalmessandyoucanstillreaditwi thoutaproblem. Thisisbecause thehumanminddo esnotreadeveryletterbyitself, butthewordasawh ole.
Noisy Channel Examples
Language Model for Generation

- search suggestions

![Search Suggestions Image]
Language Models

- **problem:** what is \( P(w) = P(w_1 \, w_2 \ldots \, w_n) \)?
- **ranking:** \( P(\text{an apple}) > P(\text{a apple}) = 0, \ P(\text{he often swim}) = 0 \)
- **prediction:** what’s the next word? use \( P(w_n \mid w_1 \ldots w_{n-1}) \)

- Obama gave a speech about ______.

- \( P(w_1 \, w_2 \ldots \, w_n) = P(w_1) \, P(w_2 \mid w_1) \ldots \, P(w_n \mid w_1 \ldots w_{n-1}) \)
- \( \approx P(w_1) \, P(w_2 \mid w_1) \, P(w_3 \mid w_1 \, w_2) \ldots \, P(w_n \mid w_{n-2} \, w_{n-1}) \) \hspace{4cm} \text{trigram}
- \( \approx P(w_1) \, P(w_2 \mid w_1) \, P(w_3 \mid w_2) \ldots \, P(w_n \mid w_{n-1}) \) \hspace{4cm} \text{bigram}
- \( \approx P(w_1) \, P(w_2) \, P(w_3) \ldots \, P(w_n) \) \hspace{4cm} \text{unigram}
- \( \approx P(w) \, P(w) \, P(w) \ldots \, P(w) \) \hspace{4cm} \text{0-gram}

sequence prob, not just joint prob.
Estimating $n$-gram Models

- Maximum likelihood: $p_{ML}(x) = \frac{c(x)}{N}; \quad p_{ML}(xy) = \frac{c(xy)}{c(x)}$
- Problem: unknown words/sequences (unobserved events)
- Sparse data problem
- Solution: smoothing
Smoothing

• have to give some probability mass to unseen events
  • (by discounting from seen events)
• Q1: how to divide this wedge up?
• Q2: how to squeeze it into the pie?

new wedge (one tiny slice for each character sequence of length <= 20 that was never observed in training data)
ML, MAP, and smoothing

• simple question: what’s P(H) if you see H H H H H?
• always maximize **posterior**: what’s the best m given d?
• with uniform **prior**, same as **likelihood** (explains data)

\[
\text{argmax}_m p(m|d) = \text{argmax}_m p(m) \cdot p(d|m) \quad \text{bayes, and } p(d) = 1
\]

\[
= \text{argmax}_m p(d|m) \quad \text{when } p(m) \text{ uniform}
\]

Suppose \( d = H H T H \)

\( m_1 \) coin is unbiased

\[ p(d|m) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = 0.0625 \]

\( m_2 \) coin is biased so that \( P(H) = \frac{3}{4} \)

\[ p(d|m) = \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} = 0.105 \]

\( m_3 \) coin is biased so that \( P(H) = \frac{9}{10} \)

\[ p(d|m) = \frac{9}{10} \cdot \frac{9}{10} \cdot \frac{1}{10} \cdot \frac{9}{10} = 0.073 \]
ML, MAP, and smoothing

- what if we have arbitrary prior
  - like $p(\theta) = \theta (1-\theta)$
- maximum a posteriori estimation (MAP)
- MAP approaches MLE with infinite
- MAP = MLE + smoothing
  - this prior is just “extra two tosses, unbiased”
  - you can inject other priors, like “extra 4 tosses, 3 Hs”
Smoothing: Add One (Laplace)

- MAP: add a “pseudocount” of 1 to every word in Vocab

- \[ P_{\text{lap}}(x) = (c(x) + 1) / (N + V) \]
  - \( V \) is Vocabulary size

- \[ P_{\text{lap}}(\text{unk}) = 1 / (N+V) \]
  - same probability for all unks

- how much prob. mass for unks in the above diagram?

- e.g., \( N=10^6 \) words, \( V=26^{20} \), \( V_{\text{obs}} = 10^5 \), \( V_{\text{unk}} = 26^{20} - 10^5 \)
Smoothing: Add Less than One

- add one gives too much weight on unseen words!
- solution: add less than one (Lidstone) to each word in \( V \)
  \[
P_{\text{lid}}(x) = \frac{c(x) + \lambda}{N + \lambda V}
\]
  \( 0 < \lambda < 1 \) is a parameter
  - \( P_{\text{lid}}(\text{unk}) = \frac{\lambda}{N + \lambda V} \) still same for unks, but smaller
- Q: how to tune this \( \lambda \)? on held-out data!
Smoothing: Witten-Bell

• key idea: use one-count things to guess for zero-counts
  • recurring idea for unknown events, also for Good-Turing

• prob. mass for unseen: $T / (N + T)$ \hspace{1cm} T: # of seen types
  • 2 kinds of events: one for each token, one for each type
  • = MLE of seeing a new type ($T$ among $N+T$ are new
  • divide this mass evenly among $V-T$ unknown words

• $p_{wb}(x) = T / (V-T)(N+T)$ \hspace{1cm} unknown word
  \hspace{1cm} = c(x) / (N+T)$ \hspace{1cm} known word

• bigram case more involved; see J&M Chapter for details
Smoothing: Good-Turing

- again, one-count words in training ~ unseen in test
- let $N_c = \# \text{ of words with frequency } r \text{ in training}$
- $P_{GT}(x) = \frac{c'(x)}{N}$ where $c'(x) = \frac{(c(x)+1)N_{c(x)+1}}{N_c(x)}$
- total adjusted mass is $\sum_c c' N_c = \sum_c (c+1) \frac{N_{c+1}}{N}$
- remaining mass: $\frac{N_1}{N}$: split evenly among unks

**Example:**

<table>
<thead>
<tr>
<th>$r$</th>
<th>$N_r$</th>
<th>$N_{r+1}$</th>
<th>$r^* \text{ or } r_{\text{avg}}$</th>
<th>$r^*/N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>100</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>40</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Smoothing: Good-Turing

- from Church and Gale (1991).
- bigram LMs. unigram vocab size = $4 \times 10^5$.
- $Tr$ is the frequencies in the held-out data (see $f_{empirical}$).

<table>
<thead>
<tr>
<th>$r$</th>
<th>$f_{MLE}$</th>
<th>$f_{empirical}$</th>
<th>$f_{Lap}$</th>
<th>$f_{del}$</th>
<th>$f_{GT}$</th>
<th>$N_r$</th>
<th>$Tr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000027</td>
<td>0.000027</td>
<td>0.000137</td>
<td>0.000037</td>
<td>0.000027</td>
<td>74</td>
<td>2019187</td>
</tr>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.000274</td>
<td>0.396</td>
<td>0.446</td>
<td>74</td>
<td>671</td>
<td>100000</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>0.000411</td>
<td>1.24</td>
<td>1.26</td>
<td>449</td>
<td>721</td>
<td>564153</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>0.000548</td>
<td>2.23</td>
<td>2.24</td>
<td>188</td>
<td>933</td>
<td>424015</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>0.000685</td>
<td>3.22</td>
<td>3.24</td>
<td>105</td>
<td>668</td>
<td>341099</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>0.000822</td>
<td>4.22</td>
<td>4.22</td>
<td>68</td>
<td>379</td>
<td>287776</td>
</tr>
<tr>
<td>6</td>
<td>5.23</td>
<td>0.000959</td>
<td>5.20</td>
<td>5.19</td>
<td>48</td>
<td>190</td>
<td>251951</td>
</tr>
<tr>
<td>7</td>
<td>6.21</td>
<td>0.00109</td>
<td>6.21</td>
<td>6.21</td>
<td>35</td>
<td>709</td>
<td>221693</td>
</tr>
<tr>
<td>8</td>
<td>7.21</td>
<td>0.00123</td>
<td>7.18</td>
<td>7.24</td>
<td>27</td>
<td>710</td>
<td>199779</td>
</tr>
<tr>
<td>9</td>
<td>8.26</td>
<td>0.00137</td>
<td>8.18</td>
<td>8.25</td>
<td>22</td>
<td>280</td>
<td>183971</td>
</tr>
</tbody>
</table>
Smoothing: Good-Turing

- Good-Turing is much better than add (less than) one
- problem 1: $N_{c_{\text{max}+1}} = 0$, so $c'_{\text{max}} = 0$
  - solution: only adjust counts for those less than $k$ (e.g., 5)
- problem 2: what if $N_c = 0$ for some middle $c$?
  - solution: smooth $N_c$ itself

\[ N_r \]

This curve ($N_r = a r^b e$) gives better $N_r$

\[ \text{RENORMALIZE!!!} \]
Smoothing: Backoff

\[ \hat{p}(w_i | w_{i-2} w_{i-1}) = \begin{cases} 
\hat{p}(w_i | w_{i-2} w_{i-1}), & \text{if } C(w_{i-2} w_{i-1} w_i) > 0 \\
\alpha_1 p(w_i | w_{i-1}), & \text{if } C(w_{i-2} w_{i-1} w_i) = 0 \\
\alpha_2 p(w_i), & \text{and } C(w_{i-1} w_i) > 0 \\
& \text{otherwise.}
\end{cases} \]
Smoothing: Interpolation

\[ \hat{p}(w_i \mid w_{i-2} w_{i-1}) = \lambda_1 p(w_i \mid w_{i-2} w_{i-1}) + \lambda_2 p(w_i \mid w_{i-1}) + \lambda_3 p(w_i) \]

subject to the constraint that \( \sum_j \lambda_j = 1 \)
Entropy and Perplexity

- classical entropy: $H(X) = - \sum p(x) \log p(x)$
- $H(L) = \lim \frac{1}{n} H(w_1...w_n)$
  $$= \lim \frac{1}{n} \sum_{w \in L} p(w_1...w_n) \log p(w_1...w_n)$$
- Shannon-McMillan-Breiman theorem:
  $$H(L) = \lim -\frac{1}{n} \log p(w_1...w_n)$$
- perplexity is $2^{H(L)}$