Introduction to Markov Models

Estimating the probability of phrases of words, sentences, etc.

But first:
A few preliminaries

What’s a Word? Tokenization Is Tricky…

How to tokenize N’T?

- It makes sense to tokenize didn’t as did n’t, hasn’t as has n’t.
- BUT can’t becomes ca n’t.

How to tokenize NEEDLE-LIKE, SEVEN-DAY, MID-OCTOBER, CRAY-3?

- It seems sensible to leave hyphenated items as single tokens.
- But:
  - New York-based
  - the New York-New Haven Railroad
  - an ad hoc solution

Q1: How to estimate the probability of a given sentence?

- A crucial step in speech recognition (and lots of other applications)
- First guess: products of unigrams

\[ \hat{P}(W) = \sum_{w \in W} P(w) \]

Proposed word lattice:

<table>
<thead>
<tr>
<th>form</th>
<th>subsidy</th>
<th>far</th>
</tr>
</thead>
</table>

Unigram counts (in 1.7 * 10⁶ words of AP text):

| form 183 | subsidy 15 | for 18185 |
| form 74  | subsidies 55 | for 570 |

Not quite right...

How to find Sentences??

The Obvious Heuristic For Sentence Boundaries:

/ [ ? ] [ - ] * /

But: I saw Mr. Jones visiting St. Peter’s basilica.

Patch: Delete the break after Mr. | Mrs. | Dr. | St. | Prof | … But:

- He left at 3 a.m. in the morning.
- He left at 3 a.m.
- In LISP 2.0 and 2. stand for the same number.

Patch: Don’t break if the next word isn’t capitalized.

But: We saw Peter on Jones St. Peter’s brother was with him.

Predicting a word sequence II

- Next guess: products of bigrams

\[ \hat{P}(W) = \sum_{w \in W} P(w_w) \]

Proposed word lattice:

| form subsidy for |
| farm subsidies far |

Bigram counts (in 1.7 * 10⁶ words of AP text):

| form subsidy 0 | subsidy for 2 |
| form subsidies 0 | subsidy for 0 |
| farm subsidy 0 | subsidies for 6 |
| farm subsidies 4 | subsidies for 0 |

Just right… But the counts are... Why?
How can we estimate $P$ correctly?

- **Problem:** Naïve Bayes model for bigrams violates independence assumptions.

**Let’s do this right...**

- Let $W = w_1 w_2 ... w_n$. Then, by the chain rule,
  \[
P(W) = P(w_1) P(w_2 | w_1) P(w_3 | w_1 w_2) ... P(w_n | w_1 ... w_{n-1})
  \]

- We can estimate $P(w_i | w_{i-1} ... w_1)$ by the Maximum Likelihood Estimator
  \[
  \hat{P}(w_i | w_{i-1} ... w_1) = \frac{\text{Count}(w_i, w_{i-1} ... w_1)}{\text{Count}(w_{i-1} ... w_1)}
  \]

  and so on...

and finally, Estimating $P(w_i | w_{i-2} ... w_1)$

Again, we can estimate $P(w_i | w_{i-2} ... w_1)$ with the MLE

\[
\hat{P}(w_i | w_{i-2} ... w_1) = \frac{\text{Count}(w_i, w_{i-2} ... w_1)}{\text{Count}(w_{i-2} ... w_1)}
\]

So to decide pot vs. pot in Heat up the oil in a large pot?, compute for pot

\[
\frac{\text{Count}(\text{“Heat up the oil in a large pot”})}{\text{Count}(\text{“Heat up the oil in a large”})}
\]

Hmm..The Web Changes Things (2008 or so)

Even the web in 2008 yields low counts!

Statistics and the Web II

So, $P(\text{“pot”} | \text{heat up the oil in a large } \_\_\_) = \frac{8}{49} \approx 0.16$

But the web has grown!!!

165/891 = 0.185
So….

- A larger corpus won’t help much unless it’s **HUGE** .... but the web is!!!

What if we only have 100 million words for our estimates??

**A BOTEC Estimate of What We Can Estimate**

What parameters can we estimate with 100 million words of training data??

Assuming (for now) uniform distribution over only 5000 words

<table>
<thead>
<tr>
<th>Event</th>
<th>Count</th>
<th>Estimate Quality?</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>5000</td>
<td>Excellent</td>
</tr>
<tr>
<td>word bigrams</td>
<td>25 million</td>
<td>OK</td>
</tr>
<tr>
<td>word trigrams</td>
<td>12.5 billion</td>
<td>Terrible</td>
</tr>
</tbody>
</table>

So even with 10^8 words of data, for even trigrams we encounter the **sparse data problem.**

**The Markov Assumption: Only the Immediate Past Matters**

The (First Order) Markov Assumption:

\[ P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2\ldots w_1) \]

Under this assumption, instead of

\[ P(W) = P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1}) \]

we estimate the probability of a string \( W \) by

\[ P(W) = P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1}) \]

**The Markov Assumption: Estimation**

We estimate the probability of each \( w_i \) given previous context by

\[ P(w_i|w_1w_2\ldots w_{i-1}) = P(w_i|w_{i-1}) \]

which can be estimated by

\[ \frac{\text{Count}(w_{i-1}nw_i)}{\text{Count}(w_{i-1})} \]

So we’re back to counting only unigrams and bigrams!!

AND we have a **correct practical estimation method for** \( P(W) \)

**Visualizing an n-gram based language model:**

**the Shannon/Miller/Selfridge method**

- To generate a sequence of \( n \) words given unigram estimates:
  - Fix some ordering of the vocabulary \( v_1v_2v_3\ldots v_k \).
  - For each word \( w_i, 1 \leq i \leq n \)
    - Choose a random value \( r_i \) between 0 and 1
    - Let \( \sum_{j=1}^{w_i} P(v_{j}) \geq r_i \)
    - \( w_{i+1} = \) the first \( v_j \) such that
Visualizing an n-gram based language model: the Shannon/Miller/Selfridge method

- To generate a sequence of \( n \) words given a 1st order Markov model (i.e. conditioned on one previous word):
  - Fix some ordering of the vocabulary \( v_1, v_2, v_3, \ldots, v_k \).
  - Use unigram method to generate an initial word \( w_1 \).
  - For each remaining \( w_i \), \( 2 \leq i \leq n \):
    - Choose a random value \( r_i \) between 0 and 1.
    - \( w_i = \text{the first } v_j \text{ such that } \sum_{m} P(v_{i-1} | v_{m}) \leq r_i \).

Wall Street Journal just isn't Shakespeare

- Unigrams:
  - Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives.
  - Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already sold M. N. corporation of trading on information such as more frequently trading to keep her.
  - They also point to ninety nine point six billion dollars from two hundred four six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

The Shannon/Miller/Selfridge method trained on Shakespeare

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of \( V^2 = 844 \text{ million possible bigrams.} \)
- So 99.96% of the possible bigrams were never seen (have zero entries in the table).
- Quadrigrams worse: What's coming out looks like Shakespeare because it is Shakespeare.

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of \( V^2 = 844 \text{ million possible bigrams.} \)
- So 99.96% of the possible bigrams were never seen (have zero entries in the table).
- Quadrigrams worse: What's coming out looks like Shakespeare because it is Shakespeare.

The Sparse Data Problem

- English word frequencies well described by Zipf's Law:
  - Zipf (1949) characterized the relation between word frequency and rank as:
    \[
    f \cdot r = C \quad \text{(for constant } C) \]
    \[
    r = \frac{C}{f} \]
    \[
    \log(r) = \log(C) - \log(f) \]
  - Purely Zipfian data plots as a straight line on a log-log scale.

*Rank (r): The numerical position of a word in a list sorted by decreasing frequency (f).
Smoothing

At least one unknown word likely \textit{per sentence} given Zipf!!

To fix 0’s caused by this, we can \textit{smooth} the data.

- Assume we know how many types \textit{never} occur in the data.
- Steal probability mass from types that occur at least once.
- Distribute this probability mass over the types that never occur.

Solution: Add-One Smoothing (again)

- \textbf{Pro:} Very simple technique
- \textbf{Cons:}
  - Probability of frequent \textit{n}-grams is underestimated
  - Probability of rare (or unseen) \textit{n}-grams is overestimated
  - Therefore, too much probability mass is shifted towards unseen \textit{n}-grams
  - All unseen \textit{n}-grams are smoothed in the same way
- Using a smaller added-count improves things but only some
- More advanced techniques (Kneser Ney, Witten-Bell) use properties of component \textit{n}-1 grams and the like...
  \textit{(Hint for this homework \textdegree )}