Introduction to Markov Models

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

But first:
A few preliminaries

What counts as a word?  A tricky question....

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 W?
• A crucial step in speech recognition (and lots of other applications)
• First guess: products of unigrams
  \[ \hat{P}(W) = \prod_{w \in W} P(w) \]

Given word lattice:

<table>
<thead>
<tr>
<th>form</th>
<th>subsidy</th>
<th>far</th>
</tr>
</thead>
<tbody>
<tr>
<td>farm</td>
<td>subsidies</td>
<td>far</td>
</tr>
</tbody>
</table>

Unigram counts (in 1.7 * 10^6 words of AP text):

| form: 183 | subsidy: 15 | far: 18185 |
| farm: 74   | subsidies: 55 | far: 570 |

Most likely word string given \( \hat{P}(W) \) isn’t 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. | Sr. | Prof | … But:

• He left at 3 a.m. in the morning.
• He left at 3 a.m.
• In LISP 2.0 and 2.0, 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
  \[ \hat{P}(W) = \prod_{i=1}^{n} P(w_i|w_{i-1}) \]

Given word lattice:

<table>
<thead>
<tr>
<th>form</th>
<th>subsidy</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>farm</td>
<td>subsidies</td>
<td>for</td>
</tr>
</tbody>
</table>

Bigram counts (in 1.7 * 10^6 words of AP text):

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

Better (if not quite right) … (But the counts are vast! Why?)
How can we estimate $P(W)$ correctly?

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

Let’s do this right……

- Let $W=w_1w_2w_3…w_n$. Then, by the chain rule,
  $$P(W) = P(w_1) * P(w_2 | w_1) * P(w_3 | w_1w_2) * … * P(w_n | w_1w_2…w_{n-1})$$

- We can estimate $P(w_i | w_{1:i-1})$ by the Maximum Likelihood Estimator
  $$\frac{\text{Count}(w_i | w_{1:i-1})}{\text{Count}(w_{1:i-1})}$$
  and $P(w_i | w_{1:i})$ by
  $$\frac{\text{Count}(w_i | w_{1:i})}{\text{Count}(w_i)}$$
  and so on……

and finally, Estimating $P(w_n | w_1w_2…w_{n-1})$

Again, we can estimate $P(w_n | w_1w_2…w_{n-1})$ with the MLE

$$\frac{\text{Count}(w_n | w_1w_2…w_{n-1})}{\text{Count}(w_1w_2…w_{n-1})}$$

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

$$\frac{\text{Count}(\text{Heat up the oil in a large pot})}{\text{Count}(\text{Heat up the oil in a large})}$$

UNLESS OUR CORPUS IS REALLY HUGE BOTH COUNTS WILL BE 0, yielding 0/0

The Web IS Huge …… (2008 version)

Even the web in 2008 yields low counts!

Statistics and the Web II

So, $P(\text{"pot" | Heat up the oil in a large___}) = \frac{8}{49}$ ≈ 0.16

The Web is HUGE!! (2016 version)

….
But 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>word unigrams</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.

Review: How can we estimate $P(W)$ correctly?

- Problem: Naive Bayes model for bigrams violates independence assumptions.

Let’s do this right….

- Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$. Then, by the chain rule,

\[
P(W) = P(\mathbf{w}_1) * P(\mathbf{w}_2 | \mathbf{w}_1) * P(\mathbf{w}_3 | \mathbf{w}_1 \mathbf{w}_2) * \ldots * P(\mathbf{w}_n | \mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_{n-1})
\]

- We can estimate $P(\mathbf{w}_i | \mathbf{w}_{i-1})$ by the Maximum Likelihood Estimator

\[
\frac{\text{Count}(\mathbf{w}_i | \mathbf{w}_{i-1})}{\text{Count}(\mathbf{w}_{i-1})}
\]

and $P(\mathbf{w}_i | \mathbf{w}_{i-2})$ by

\[
\frac{\text{Count}(\mathbf{w}_i | \mathbf{w}_{i-2})}{\text{Count}(\mathbf{w}_{i-2})}
\]

and so on…

The Markov Assumption: Only the Immediate Past Matters

The (First Order) Markov Assumption:

$P(\mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_n) = P(\mathbf{w}_1) * P(\mathbf{w}_2 | \mathbf{w}_1) * P(\mathbf{w}_3 | \mathbf{w}_1 \mathbf{w}_2) * \ldots * P(\mathbf{w}_n | \mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_{n-1})$

Under this assumption, instead of

\[
P(W) = P(\mathbf{w}_1) * P(\mathbf{w}_2 | \mathbf{w}_1) * P(\mathbf{w}_3 | \mathbf{w}_1 \mathbf{w}_2) * \ldots * P(\mathbf{w}_n | \mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_{n-1})
\]

we estimate the probability of a string $W$ by

\[
P(W) = P(\mathbf{w}_1) * P(\mathbf{w}_2 | \mathbf{w}_1) * P(\mathbf{w}_3 | \mathbf{w}_2) * \ldots * P(\mathbf{w}_n | \mathbf{w}_{n-1})
\]

The Markov Assumption: Estimation

We estimate the probability of each $\mathbf{w}_i$, given previous context by

$P(\mathbf{w}_i | \mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_{i-1}) = P(\mathbf{w}_i | \mathbf{w}_{i-1})$

which can be estimated by

\[
\frac{\text{Count}(\mathbf{w}_i | \mathbf{w}_{i-1})}{\text{Count}(\mathbf{w}_{i-1})}
\]

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

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

given the Markov assumption!

Markov Models

A bigram model can be viewed as a Markov model, a probabilistic FSA with

- $S$, a set of states, one for each word $w_i$ in the vocabulary.
- $A$, a transition matrix where $a(i,j)$ is the probability of going from state $w_i$ to state $w_j$.
- The probability $a(i,j)$ can be estimated by

\[
a(i,j) = \frac{\text{Count}(w_i w_j)}{\text{Count}(w_i)}
\]

- $\Pi$, a vector of initial state probabilities, where $\pi(i)$ is the probability of the first word being $w_i$.

Markov Models
Cumulative distribution Functions (CDFs)

- The CDF of a random variable \( X \) is denoted by \( F(x) \) and is defined by \( F(x) = \Pr(X \leq x) \).
- If \( X \) is a discrete random variable, then
  \[
  F(x) = \sum_{x_i \leq x} \Pr(X = x_i).
  \]

Corpus: “the mouse ran up the clock. The spider ran up the waterspout.”
- \( \Pr(\text{the}) = \frac{4}{12} = \frac{1}{3} \)
- \( \Pr(\text{ran}) = \Pr(\text{up}) = \frac{1}{6} \)
- \( \Pr(\text{mouse}) = \Pr(\text{clock}) = \Pr(\text{spider}) = \Pr(\text{waterspout}) = \frac{1}{12} \)

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 \( v_1, v_2, v_3, \ldots, v_k \).
- For each word \( w_i \), \( 1 \leq i \leq n \)
  - Choose a random value \( r_i \) between 0 and 1
  - \( w_i \) is the first \( v_j \) such that \( \sum_{m=1}^{j-1} \Pr(v_m) \geq r_i \).

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
  - \( w_i \) is the first \( v_j \) such that \( \sum_{m=1}^{j} \Pr(v_m | w_{i-1}) \geq r_i \).

The Shannon/Miller/Selfridge method trained on Shakespeare

Unigram
To him swallowed confess lean both. Which. Of save extract for an ay device and wise life have
Every seven now seventy six, his
Hill be late speaks yet a mane to lay less first you enter.
Are where clark and right have scissor excellency took of. Sleep leave we near, the like

Bigram
What means, sir I confess she? then all sorts, he is trist, captious
Why dost stand forth thy enemies, forsooth, he is this jupière hit the King Henry. Live king. Follow
What we, hath got to the that lost and sert to cold and younger backward, nor the first gentlemen?

Trigram
Sweet prince. Fabulous shall the. Harry of Montesquieu's grieve
This shall forth it should be haed of enuma made it empty.
Indeed the like, and had a very good fram.
Fly, and will not relate these news of price. Therefore the solemn of purging, as they say. 'Yo-ho-

Quadrigram
Kepp Henry What! I will go seek the traitor Gloucester. Extant some of the watch. A great breakfast set'th he
Will you not tell me when I am
It cannot be hat on.
Indeb the short and the long. Many. An a noble Lapidus.

Wall Street Journal just isn’t Shakespeare

Unigram
Months the say and issue of year foreign new exchange’s septembar were recession ex-

change now endorsed a acquire to six executives

Bigram
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 old M. X. corporation of living on information such as more frequently fishing to loc-

Trigram
They also point to ninety nine point six billion dollars from two hundred four oh six three
percent of the rates of interest rates as Mexico and Brazil on market conditions
**Shakespeare as corpus**

- \( N = 884,647 \) tokens, \( V = 29,066 \)
- Shakespeare produced 300,000 bigram types out of \( V^2 = 844 \) 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**

**Again**

Under the Markov Assumption,

\[
P(W) = P(w_1) \cdot P(w_2|w_1) \cdot \ldots \cdot P(w_{n-1}|w_{n-2}) \cdot P(w_n|w_{n-1})
\]

But what if we’ve never before seen \( w_{n-1}w_{n-2} \) in string \( W \)?

Then our estimate of \( P(w_{n-1}|w_{n-2}) \) is

\[
\frac{\text{Count}(w_{n-1}|w_{n-2})}{\text{Count}(w_{n-2})} \approx 0
\]

So our estimate of \( P(W) = 0 \!.

- So we smooth....
- How likely is a 0 count? Much more likely than I let on!!!