Language Models: Given the text so far, what word comes ________?

Introduction to Markov Models

REVISED SET

CIS 391
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
Counting Linguistic Entities II

What's A Sentence??

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.
Estimating the Probability of a String of Words

How can we estimate the probability of a string of words \( \mathbf{W} \)?

Easy!

Let \( \mathbf{W} = w_1 w_2 w_3 \ldots w_n \). Then, by the chain rule,

\[
P(\mathbf{W}) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1w_2) \times \ldots \times P(w_n|w_1 \ldots w_{n-1})
\]

We can estimate \( P(w_2|w_1) \) by the Maximum Likelihood Estimator

\[
\frac{\text{Count}(w_1w_2)}{\text{Count}(w_1)}
\]

and \( P(w_3|w_1w_2) \) by

\[
\frac{\text{Count}(w_1w_2w_3)}{\text{Count}(w_1w_2)}
\]

and so on....
Predicting the Next Word in a Sequence

RECOGNITION ERRORS in tasks ranging from optical character recognition to speech recognition can be reduced by looking at string probabilities.

RECOGNIZER POSSIBILITIES:

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

UNIGRAM FREQUENCIES (in $1.7 \times 10^6$ words):

<table>
<thead>
<tr>
<th>FORM</th>
<th>183</th>
<th>subsidy</th>
<th>15</th>
<th>FOR</th>
<th>18185</th>
</tr>
</thead>
<tbody>
<tr>
<td>farm</td>
<td>74</td>
<td>SUBSIDIES</td>
<td>55</td>
<td>far</td>
<td>570</td>
</tr>
</tbody>
</table>
Predicting the Next Word in a Sequence II

RECOGNIZER POSSIBILITIES:

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

BIGRAM FREQUENCIES:

<table>
<thead>
<tr>
<th>form subsidies</th>
<th>0</th>
<th>subsidies far</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FARM SUBSIDIES</td>
<td>4</td>
<td>SUBSIDIES FOR</td>
<td>6</td>
</tr>
<tr>
<td>form subsidy</td>
<td>0</td>
<td>subsidy far</td>
<td>0</td>
</tr>
<tr>
<td>farm subsidy</td>
<td>0</td>
<td>subsidy for</td>
<td>2</td>
</tr>
</tbody>
</table>

So how can we do this right?
Predicting the Next Word in a Sequence III

A good idea: Let’s predict that the next word in a sequence is *the most likely word* among the candidate possibilities.

Again, let \( W = w_1 w_2 w_3 \ldots w_n \).

So, if we’ve already seen \( w_2 \ldots w_{i-1} \), how do we estimate the probability of \( w_i \)?

Idea I: Use \( P(w_i) \).
BUT: See last slide…

A much better idea, formally:
Estimate the probability of \( w_i \) *given everything that has gone before!*
More formally, estimate \( P(w_i | w_1 w_2 \ldots w_{i-1}) \).
Estimating $P(w_i | w_1 w_2 \ldots w_{i-1})$, Take II

Again, we can estimate $P(w_i | w_1 w_2 \ldots w_{i-1})$ with the MLE

$$\frac{\text{Count}(w_1 w_2 \ldots w_i)}{\text{Count}(w_1 w_2 \ldots w_{i-1})}$$

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

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

NOT!!
Hmm.. The Web Changes Things
So, \( P(\text{"pot"}\mid\text{"heat up the oil in a large"}) = \frac{8}{49} = \)
But the web has grown!!!

"heat up the oil in a large pot"

1) Heat up the oil in a large pot over medium-high heat.  
2) Add the garlic and onions and sauté until tender (about 6-7 minutes). Add the flour ...

spicy | Real Men Eat Green - Where Manliness Meets Greenli...
www.realmeneatgreen.com/tag/spicy/ 
Jul 12, 2013 - 1) Heat up the oil in a large pot over medium-high heat.  
2) Add the garlic and onions and sauté until tender (about 6-7 minutes). Add the flour ...

soup | Real Men Eat Green - Where Manliness Meets Greenli...
www.realmeneatgreen.com/tag/soup/ 
Jul 12, 2013 - 1) Heat up the oil in a large pot over medium-high heat.  
2) Add the garlic and onions and sauté until tender (about 6-7 minutes). Add the flour ...

recipe | Real Men Eat Green - Where Manliness Meets Greenli...
www.realmeneatgreen.com/tag/recipe/ 
Jul 12, 2013 - 1) Heat up the oil in a large pot over medium-high heat.  
2) Add the garlic and onions and sauté until tender (about 6-7 minutes). Add the flour ...

Dinner | Life: From The Top
lifefromthetop.com/category/dinner/ 
Heat up the oil in a large pot (must fit 3 qts of stock plus veg & noodles), and sauté Onions and Garlic. Add Black Pepper when everything is heated. Sauté until ...
...
So….

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

- **WHY??**
  Power law (Zipf) distribution of language
Zipf’s Law

- Zipf (1949) characterized the relation between word frequency and rank as:

\[ f \cdot r = C \text{ (for constant } C) \]

\[ r = \frac{C}{f} \]

\[ \log(r) = \log(C) - \log(f) \]

- 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 \((f)\).
Word frequency & rank in Brown Corpus vs Zipf

From: Interactive mathematics http://www.intmath.com
Zipf’s law for the Brown corpus
The Sparse Data Problem

WHAT PARAMETERS CAN WE ESTIMATE WITH A 100 MILLION WORDS OF TRAINING DATA?

Assuming (for now) uniform distribution of 5000 words and 40 part of speech tags..

<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>
The Markov Assumption: Only the Immediate Past Matters

The (First Order) Markov Assumption:

\[ P(w_i|w_1 \ldots w_{i-1}) = P(w_i|w_{i-1}) \]

Under this assumption, instead of

\[ P(W) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1w_2) \times \ldots \times P(w_n|w_1 \ldots w_{n-1}) \]

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

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

We estimate the probability of $w_i$ given previous context by

$$P(w_i | w_1w_2...w_{i-1}) = P(w_i | w_{i-1})$$

which can be estimated by

$$\frac{\text{Count}(w_{i-1}w_i)}{\text{Count}(w_{i-1})}$$

So we’re back to bigrams!!!
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
  \[
  a(i, j) = \frac{\text{Count}(w_iw_j)}{\text{Count}(w_i)}
  \]

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

![Diagram of a Markov model with transition probabilities](image.png)
Visualizing an \( n \)-gram based language model: the Shannon/Miller/Selfridge method

- To generate a series of \( k \) unigrams:
  - Fix some ordering of the vocabulary \( v_1, v_2, v_3, \ldots, v_N \).
  - For each word \( w_i \), \( 1 \leq i \leq k \)
    - Choose a random value \( r_i \) between 0 and 1
    - \( w_i = \) the first \( v_j \) such that \( \sum_{n=1}^{j} p(v_n) \geq r_i \)
Visualizing an $n$-gram based language model: the Shannon/Miller/Selfridge method

- To generate a series of $k$ bigrams:
  - Fix some ordering of the vocabulary $v_1 v_2 v_3 \ldots v_N$.
  - Use unigram method to generate an initial word $w_1$.
  - For each word $w_i$, $2 \leq i \leq k$
    - Choose a random value $r_i$ between 0 and 1
    - $w_i = \text{the first } v_j \text{ such that } \sum_{n=1}^{j} p(v_n \mid w_{prev}) > r_i$
    - $w_{prev} = w_i$
The Shannon/Miller/Selfridge method trained on Shakespeare

**Unigram**
To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have Every enter now severally so, let Hill he late speaks; or! a more to leg less first you enter Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

**Bigram**
What means, sir. I confess she? then all sorts, he is trim, captain. Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

**Trigram**
Sweet prince, Falstaff shall die. Harry of Monmouth’s grave. This shall forbid it should be branded, if renown made it empty. Indeed the duke; and had a very good friend. Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.

**Quadrigram**
King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; Will you not tell me who I am? It cannot be but so. Indeed the short and the long. Marry, ’tis a noble Lepidus.

(This and next two slides from Jurafsky)
Wall Street Journal just isn’t Shakespeare

**Unigram**

Months the my and issue of year foreign new exchange’s september were recession ex-
change new 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
keep her

**Trigram**

They also point to ninety nine point six billion dollars from two hundred four oh six three
percent of the rates of interest stores 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,

\[ P(W) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_{i+1}|w_i) \times \ldots \times P(w_n|w_{n-1}) \]

But what if we’ve never before seen \( w_iw_{i+1} \) in string \( W \)?

Then our estimate of \( P(w_{i+1}|w_i) \) is

\[
\frac{\text{Count}(w_iw_{i+1})}{\text{Count}(w_i)} = \frac{0}{\text{Count}(w_i)} = 0
\]

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

This black art is why NLP is taught in the engineering school – Jason Eisner
Smoothing

- To fix this, we can *smooth* the data.
- Assume we know how many types *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.
Smoothing

….is like Robin Hood:

- it steals from the rich
- and gives to the poor
Solution: Add-One Smoothing (again)

- **Pro:** Very simple technique

- **Cons:**
  - Probability of frequent $n$-grams is underestimated
  - Probability of rare (or unseen) $n$-grams is overestimated
  - Therefore, too much probability mass is shifted towards unseen $n$-grams
  - All unseen $n$-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 $n-1$ grams and the like...
  
  *(Hint for next homework 😊)*