Next Word Prediction

From a NY Times story...

– Stocks plunged this …. 
– Stocks plunged this morning, despite a cut in interest rates 
– Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall ...
– Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began trading for the first time since last Tuesday's terrorist attacks.

Human Word Prediction

• Clearly, at least some of us have the ability to predict future words in an utterance.
• How?

– Domain knowledge 
– Syntactic knowledge 
– Lexical knowledge
Claim

• A useful part of the knowledge needed to allow Word Prediction (guessing the next word) can be captured using simple statistical techniques.
• In particular, we'll rely on the notion of the probability of a sequence (e.g., sentence) and the likelihood of words co-occurring.

Word Prediction

• We can formalize this task using what are called \(N\)-gram models.
• \(N\)-grams are token sequences of length \(N\).
• Our earlier example contains the following 2-grams (aka bigrams)
  – (I notice), (notice three), (three guys), (guys standing), (standing on), (on the)
• Given knowledge of counts of \(N\)-grams such as these, we can guess likely next words in a sequence.

Why is this useful?

Example applications that employ language models:
• Speech recognition
• Handwriting recognition
• Spelling correction
• Machine translation systems
• Optical character recognizers

Real Word Spelling Errors

• They are leaving in about fifteen minutes to go to her horse.
• The study was conducted mainly by John Black.
• The design and construction of the system will take more than a year.
• Hopefully, all will continue smoothly in my absence.
• I need to notify the bank of….
• He is trying to figure out.

Handwriting Recognition

• Assume a note is given to a bank teller, which the teller reads as I have a gub.
• NLP to the rescue ….
  – gub is not a word
  – gun, gum, Gus, and gull are words, but gun has a higher probability in the context of a bank.

Counting

• Simple counting lies at the core of any probabilistic approach. So let's first take a look at what we’re counting.
  – He stepped out into the hall, was delighted to encounter a water brother.
  • 13 tokens, 15 if we include “,” and “.” as separate tokens.
  • Assuming we include the comma and period, how many bigrams are there?
Counting

• Not always that simple
  – I do uh main- mainly business data processing

• Spoken language poses various challenges.
  – Should we count “uh” and other fillers as tokens?
  – What about the repetition of “mainly”? Should such do-overs count twice or just once?
  – The answers depend on the application.
    • If we’re focusing on something like ASR to support indexing for search, then “uh” isn’t helpful (it’s not likely to occur as a query).
    • But filled pauses are very useful in dialog management, so we might want them there.

Counting: Types and Tokens

• How about
  – They picnicked by the pool, then lay back on the grass and looked at the stars.

  – 18 tokens (again counting punctuation)
  – But we might also note that “the” is used 3 times, so there are only 16 unique types (as opposed to tokens).
  – In going forward, we’ll have occasion to focus on counting both types and tokens of both words and N-grams.

Counting: Wordforms

• Should “cats” and “cat” count as the same when we’re counting?
• How about “geese” and “goose”?
• Some terminology:
  – Lemma: a set of lexical forms having the same stem, major part of speech, and rough word sense
  – Wordform: fully inflected surface form
• Again, we’ll have occasion to count both lemmas and wordforms

Counting: Corpora

• Corpora are (generally online) collections of text and speech
  – Linguistic Data Consortium (LDC)
  – Brown Corpus
  – Wall Street Journal and AP News corpora
  – ATIS, Broadcast News (speech)
  – TDT (test and speech)
  – Switchboard, Call Home (speech)
  – AQUAINT
• Brown et al (1992) large corpus of English text
  – 503 million wordform tokens
  – 293,181 wordform types
• Google
  – Crawl of 1,024,908,267,229 English tokens
  – 13,588,391 wordform types
• Numbers, misspellings, names, acronyms, etc.

Summary of Terminology

• Sentence: unit of written language
• Utterance: unit of spoken language
• Word Form: the inflected form that appears in the corpus
• Lemma: lexical forms having the same stem, part of speech, and word sense
• Types: number of distinct words in a corpus (vocabulary size)
• Tokens: total number of words

Language Modeling

• We can model the word prediction task as the ability to assess the conditional probability of a word given the previous words in the sequence
  – $P(w_n|w_1, w_2, \ldots, w_{n-1})$
• We’ll call a statistical model that can assess this a Language Model
Language Modeling

• How might we go about calculating such a conditional probability?

\[ P(\text{the} \mid \text{its water is so transparent that}) = \frac{\text{Count(its water is so transparent that the)}}{\text{Count(its water is so transparent that)}} \]

Language Modeling

• Unfortunately, for most sequences and for most text collections we won’t get good estimates from this method.
  – What we’re likely to get is 0. Or worse 0/0.
• Clearly, we’ll have to be a little more clever.
  – Let’s use the chain rule of probability
  – And a particularly useful independence assumption.

Chain Rule

conditional probability: \[ P(A \mid B) = \frac{P(A \cap B)}{P(B)} \]

So: \[ P(A \cap B) = P(B \mid A)P(A) \]

“the dog”: \[ P(\text{The} \land \text{dog}) = P(\text{dog} \mid \text{the})P(\text{the}) \]

“the dog bites”:
\[ P(\text{The} \land \text{dog} \land \text{bites}) = P(\text{The})P(\text{dog} \mid \text{the})P(\text{bites} \mid \text{The} \land \text{dog}) \]

Chain Rule

The probability of a word sequence is the probability of a conjunctive event.

\[
P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})
\]

Unfortunately, that’s really not helpful in general. Why?

Markov Assumption

\[ P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_1^{n-k}) \]

• \( P(w_n) \) can be approximated using only N-1 previous words of context
• This lets us collect statistics in practice
• Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past
• Order of a Markov model: length of prior context

Markov Assumption

• Probabilities come from a training corpus, which is used to design the model.
  – overly narrow corpus: probabilities don’t generalize
  – overly general corpus: probabilities don’t reflect task or domain
• A separate test corpus is used to evaluate the model, typically using standard metrics
  – held out test set
  – cross validation
  – evaluation differences should be statistically significant

Training and Testing
Simple N-Grams

- An N-gram model uses the previous N-1 words to predict the next one:
  - $P(w_n | w_{n-1})$
  - We'll pretty much always be dealing with $P(\text{word} | \text{some prefix})$
- unigrams: $P(\text{dog})$
- bigrams: $P(\text{dog} | \text{big})$
- trigrams: $P(\text{dog} | \text{the big})$
- quadrigrams: $P(\text{dog} | \text{the big dopey})$

Using N-Grams

- Recall that
  - $P(w_n | w_{1..n-1}) \approx P(w_n | w_{n-N+1..n-1})$
- For a bigram grammar
  - $P(\text{sentence})$ can be approximated by multiplying all the bigram probabilities in the sequence
    - $P(\text{I want to each Chinese food}) = P(\text{I} | <\text{start}>)$
    - $P(\text{want} | \text{I}) P(\text{to} | \text{want}) P(\text{eat} | \text{to}) P(\text{Chinese} | \text{eat}) P(\text{food} | \text{Chinese})$

Berkeley Restaurant Project Sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

A Bigram Grammar Fragment from BERP

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eat on</td>
<td>.16</td>
<td>Eat Thai</td>
<td>.03</td>
</tr>
<tr>
<td>Eat some</td>
<td>.06</td>
<td>Eat breakfast</td>
<td>.03</td>
</tr>
<tr>
<td>Eat lunch</td>
<td>.06</td>
<td>Eat in</td>
<td>.02</td>
</tr>
<tr>
<td>Eat dinner</td>
<td>.05</td>
<td>Eat Chinese</td>
<td>.02</td>
</tr>
<tr>
<td>Eat at</td>
<td>.04</td>
<td>Eat Mexican</td>
<td>.02</td>
</tr>
<tr>
<td>Eat a</td>
<td>.04</td>
<td>Eat tomorrow</td>
<td>.01</td>
</tr>
<tr>
<td>Eat Indian</td>
<td>.04</td>
<td>Eat dessert</td>
<td>.007</td>
</tr>
<tr>
<td>Eat today</td>
<td>.03</td>
<td>Eat British</td>
<td>.001</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;start&gt; I</td>
<td>.25</td>
<td>Want some</td>
<td>.04</td>
</tr>
<tr>
<td>&lt;start&gt; I'd</td>
<td>.06</td>
<td>Want Thai</td>
<td>.01</td>
</tr>
<tr>
<td>&lt;start&gt; Tell</td>
<td>.04</td>
<td>To eat</td>
<td>.26</td>
</tr>
<tr>
<td>&lt;start&gt; I'm</td>
<td>.02</td>
<td>To have</td>
<td>.14</td>
</tr>
<tr>
<td>I want</td>
<td>.32</td>
<td>To spend</td>
<td>.09</td>
</tr>
<tr>
<td>I would</td>
<td>.29</td>
<td>To be</td>
<td>.02</td>
</tr>
<tr>
<td>I don't</td>
<td>.08</td>
<td>British food</td>
<td>.60</td>
</tr>
<tr>
<td>I have</td>
<td>.04</td>
<td>British restaurant</td>
<td>.15</td>
</tr>
<tr>
<td>Want to</td>
<td>.65</td>
<td>British cuisine</td>
<td>.01</td>
</tr>
<tr>
<td>Want a</td>
<td>.05</td>
<td>British lunch</td>
<td>.01</td>
</tr>
</tbody>
</table>

- $P(\text{I want to eat British food}) = P(\text{I} | <\text{start}>)$
  - $P(\text{want} | \text{I}) P(\text{to} | \text{want}) P(\text{eat} | \text{to}) P(\text{British} | \text{eat}) P(\text{food} | \text{British}) = .25* .32* .65* .26* .001* .60 = .000080$
- vs. $P(\text{I want to eat Chinese food}) = .00015$
- Probabilities seem to capture "syntactic" facts, "world knowledge"
  - eat is often followed by a NP
  - British food is not too popular
How do we get the N-gram probabilities?

Use Maximum Likelihood Estimation

What is Maximum Likelihood Estimation?

Some review

Toss a coin: head or tail.
Suppose we know: $P(\text{head}) = 0.5$
Let’s toss the coin 9 times, what is the probability of having 4 heads and 5 tails?

Some review

Toss a coin: head or tail.
Suppose we know: $P(\text{head}) = 0.5$
Let’s toss the coin $n$ times, the probability of having $h$ heads:

$$
\binom{n}{h} p^h (1-p)^{n-h}
$$

If $p$ (i.e., the model parameter) is known, we can get the probability of a certain observation (i.e., $P(O|p)$).

Maximum Likelihood Estimation

What if we only see the observation, we don’t know $p$, then we want to find:

$$
\arg\max_p P(p | O)
$$

Because:

$$
P(p | O) \propto P(O | p)
$$

(why?)

Therefore we need to find:

$$
\arg\max_p P(O | p)
$$

Find the model parameter that makes the observation data most likely

Simple example of MLE

If we toss a coin 100 times and observe 56 heads and 44 tails, what is $P(\text{head})$? i.e., $p$?

arg max_p P(p | O)

$$
P(p = 0.5 | O) \propto P(O | p = 0.5) = \frac{100!}{56!44!} 0.5^{56} 0.5^{44} = 0.0389
$$

$$
P(p = 0.52 | O) \propto P(O | p = 0.52) = \frac{100!}{56!44!} 0.52^{56} 0.48^{44} = 0.058
$$
### Simple example of MLE

**Maximum Likelihood Estimation (MLE)**

- **P(p | O)**
  - 0.0378
  - 0.62
  - 0.57
  - 0.0738
  - 0.58

### MLE for N-gram

N-gram models can be trained by counting and normalization.

**Bigram:**

\[ P(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \]

**N-gram:**

\[ P(w_n | w_{n-N+1}) = \frac{C(w_{n-N+1}w_n)}{C(w_{n-N+1})} \]

### BERP Bigram Counts

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>Want</th>
<th>To</th>
<th>Eat</th>
<th>Chinese</th>
<th>Food</th>
<th>Lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Want</td>
<td>3</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>To</td>
<td>3</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Eat</td>
<td>0</td>
<td>2</td>
<td>19</td>
<td>2</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>Food</td>
<td>19</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### BERP Bigram Probabilities

- Normalization: divide each row’s counts by appropriate unigram counts.

### Kinds of Knowledge

- As crude as they are, N-gram probabilities capture a range of interesting facts about language.

  - \( P(\text{english} | \text{want}) = .0011 \)
  - \( P(\text{chinese} | \text{want}) = .0065 \)
  - \( P(\text{to} | \text{want}) = .66 \)
  - \( P(\text{to} | \text{eat}) = .28 \)
  - \( P(\text{food} | \text{to}) = 0 \)
  - \( P(\text{want} | \text{spend}) = 0 \)
  - \( P(1 | <s>) = .25 \)
• What about
  – $P(I | I) = 0.0023$
  – $P(I | \text{want}) = 0.0025$
  – $P(I | \text{food}) = 0.013$

Shannon’s Method
• Assigning probabilities to sentences is all well and good, but it’s not terribly illuminating. A more interesting task is to turn the model around and use it to generate random sentences that are *like* the sentences from which the model was derived.
• Generally attributed to Claude Shannon.

Approximating Shakespeare
• As we increase the value of $N$, the accuracy of the $n$-gram model increases
• Generating sentences with random unigrams...
  – Every enter now severally so, let
  – Hill he late speaks; or! a more to leg less first you enter
• With bigrams...
  – 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.

• Quadrigrams
  – What! I will go seek the traitor Gloucester.
  – Will you not tell me who I am?
N-Gram Training Sensitivity

- If we repeated the Shakespeare experiment but trained on a Wall Street Journal corpus, there would be little overlap in the output.
- This has major implications for corpus selection or design.

<table>
<thead>
<tr>
<th>N-Gram Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono: the say and know of your foreign exchange's september some recession ex-cadept mail it's all a surprise to six executives.</td>
</tr>
<tr>
<td>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. corporates of living or information such as more frequently falling to keep her.</td>
</tr>
<tr>
<td>Trigram: They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rate of interest rates in Mexico and turmoil on market conditions.</td>
</tr>
</tbody>
</table>

Evaluation

- How do we know if our models are any good?
- And in particular, how do we know if one model is better than another.
- Well Shannon’s game gives us an intuition.
- The generated texts from the higher order models sure look better. That is, they sound more like the text the model was obtained from.
- But what does that mean? How do we quantify that?

Unknown Words

- But once we start looking at test data, we’ll run into words that we haven’t seen before (pretty much regardless of how much training data you have).
- With an Open Vocabulary task
  - Create an unknown word token <UNK>
  - Training of <UNK> probabilities
- From a dictionary or
- A subset of terms from the training set
- At test normalization phase, any training word not in L changed to <UNK>
- At test time
- Use UNK counts for any word not in training

Evaluating a Language Model

- Given two language models, how to measure which one is better?
- What could be an intuitive but expensive way?
  - Extrinsic evaluation: speech recognition
  - Expensive
  - Intrinsic evaluation: Perplexity

Perplexity

- Perplexity is the probability of the test set (assigned by the language model), normalized by the number of words: \( PP(W) = \frac{1}{L} \prod_{i=1}^{L} P(w_i | w_{i-1}) \)
- For bigrams: \( PP(W) = \frac{1}{L} \prod_{i=1}^{L} P(w_i | w_{i-1}) \)

Minimizing perplexity is the same as maximizing probability
- The best language model is one that best predicts an unseen test set.
Evaluating a Language Model

- Given test data (T), which has n sentences: S1, S2, … Sn
- We could look at the probability under our model \( P(T) = \prod_{i=1}^{n} P(S_i) \) or more conveniently, the log probability
  \[ \log \prod_{i=1}^{n} P(S_i) = \sum_{i=1}^{n} \log P(S_i) \]
- The usual measure is **perplexity**
  \[ \text{Perplexity} = 2^{-\frac{x}{W}} \text{ where } x = \frac{1}{W} \sum_{i=1}^{W} \log P(S_i) \]
  where \( W \) is the total number of words in the test data

---

Lower perplexity means a better model

- Training 38 million words, test 1.5 million words, WSJ

<table>
<thead>
<tr>
<th>N-gram Order</th>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perplexity</td>
<td>962</td>
<td>170</td>
<td>109</td>
</tr>
</tbody>
</table>