Homework

• For Monday 3/14:
  – Get bigram and unigram frequencies from ANC website: http://americannationalcorpus.org/frequency.html
  – Implement two of the collocation-finding methods described in chapter 5
  – Find the top $n$ bigrams by each measure
  – Write it up – a paragraph or two describing what’s going on

• Read chapter 6

• Midterm week after next
Floating point arithmetic

- Digital computers can’t represent real numbers:

  ```
  bulba% python
  Python 2.2.1 (#1, Aug 30 2002, 12:15:30)
  [GCC 3.2 20020822 (Red Hat Linux Rawhide 3.2-4)] on linux2
  Type "help", "copyright", "credits" or "license" for more information.
  >>> 3.3
  3.2999999999999998
  >>>
  ```

- Financial calculations use integers

- Scientific calculations use approximations, which vary in their accuracy

- Standard for floating point calculations: IEEE 754
Floating point arithmetic

- Floating point numbers are stored as a *mantissa* and an *exponent*.

- IEEE floating point formats:

<table>
<thead>
<tr>
<th>precision</th>
<th>min</th>
<th>max</th>
<th>eps</th>
<th>digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>single</td>
<td>$1.2 \times 10^{-38}$</td>
<td>$3.4 \times 10^{38}$</td>
<td>$1.2 \times 10^{-7}$</td>
<td>7</td>
</tr>
<tr>
<td>double</td>
<td>$2.2 \times 10^{-308}$</td>
<td>$1.8 \times 10^{308}$</td>
<td>$2.2 \times 10^{-16}$</td>
<td>16</td>
</tr>
</tbody>
</table>

- Just because you can represent $10^{300}$ doesn’t mean you get 300 significant digits!

- Default in python and perl is double precision

- Don’t use single precision (*float*) unless you have a good reason.
Floating point arithmetic

• It’s easy to lose precision:

\[ 0.2 - 0.0 = 0.20000000000000001 \]
\[ 100000.2 - 100000.0 = 0.1999999999708962 \]

• Things to watch out for:
  – subtractions of numbers that are nearly equal,
  – additions of numbers whose magnitudes are nearly equal, but whose signs are opposite
  – additions and subtractions of numbers that differ greatly in magnitude
  – multiplying by very small numbers or dividing by very large numbers
  – avoid unstable/ill-conditioned algorithms
Floating point arithmetic

• Many problems with probabilities can be avoided by using log probabilities instead
  – exponents become products
  – products become sums

• Exact comparisons between floating point numbers can be misleading

• The same operations performed in a different order or on different hardware may give different results
Independence

- All statistical methods we looked at make the “i.i.d.” assumption

- This is clearly false for language, though we can sometimes fake it (bag of words model)

- Noisy channel models require a language model which assigns a probability to a text:

\[ P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i) \]

- The bag of words assumption makes estimation of the model easy, but is also very unrealistic:

  Also assumption bag but easy estimation is makes model of of, the the unrealistic very words.
Independence

- Instead, we can apply the chain rule:

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

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

- The fully independent model is a multinomial distribution with one parameter per word (conservatively, 20,000 parameters)

- The fully dependent model is a multinomial distribution with one parameter per word per context (something like \(10^{220}\) parameters)

- By comparison, there are maybe \(10^{79}\) atoms in the universe.
Independence

• There are too many different histories for the fully dependent model to be practical

• As a compromise, we can form equivalence classes of histories.

• Suppose $\phi$ is many-to-one function mapping histories to equivalence classes. Then our model is:

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | \phi(w_1, \ldots, w_{i-1}))$$

• Now all we need to do is specify $\phi$, and come up with a way of estimating parameters, and we’re done
Markov chains

• A particularly simple way of forming equivalence classes treats language as a *Markov chain*

• A Markov chain is a sequence of random variables in which, given the present, the future is conditionally independent of the past:

\[ P(W_n = w_n | W_1 = w_1, \ldots, W_{n-1} = w_{n-1}) = P(W_n = w_n | W_{n-1} = w_{n-1}) \]

• Only a fixed length previous history is relevant: Markov assumption, limited horizon

• Plus language is stationary and ergodic → i.i.d.
Markov chains

- Taking language as a Markov chain leads to $n$-gram models:

$$ P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | \phi(w_1, \ldots, w_{i-1})) $$

$$ = \prod_{i=1}^{n} P(w_i | w_{i-2}, w_{i-1}) $$

- Bigrams = first order Markov model, trigram = second order, four-gram = third order, etc

- Generally speaking, higher order models do a better job of predicting probabilities, but have more parameters (and so require more training data)
Markov models

- **First order**
  To him swallowed confess hear both. Which. Of save on train for are ay device and rote life have.

- **Second order**
  Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

- **Third order**
  This shall forbid it should be branded, if renown made it empty.

- **Fourth order**
  They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!
Markov models

- **First order**
  Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives.

- **Second order**
  Last December through the way to preserve the Hudson corporation NBEC Taylor would seem to complete the major central planners 1.5% of the USE has already old MX corporation of living on information such as more frequently fishing to keep her.

- **Third order**
  They also point to $99.6 billion from 204063% of the rates of interest stores as Mexico and Brazil on market conditions.
Parameter estimation

• For a trigram model:

\[ P(w_n|w_{n-2}, w_{n-1}) = \frac{P(w_{n-2}, w_{n-1}, w_n)}{P(w_{n-2}, w_{n-1})} \]

So, we need to estimate the marginal bigram and trigram probabilities.

• If we want to estimate the parameter vector \( \theta \) from a training corpus \( c \), then find:

\[
\hat{\theta} = \arg\max_{\theta} P(\theta|c) \\
= \arg\max_{\theta} \frac{P(c|\theta) P(\theta)}{P(c)} \\
= \arg\max_{\theta} P(c|\theta) P(\theta)
\]
Parameter estimation

- If we assume $P(\theta)$ is uniform, then we get the maximum likelihood estimate:

$$\hat{\theta} = \arg\max_{\theta} P(c|\theta)$$

- For trigram probabilities, the MLE is:

$$P(w_1, w_2, w_3) = \frac{C(w_1, w_2, w_3)}{N}$$

- **Maximum** likelihood sounds good, but how well does it work?

- We need a way to measure the quality of a language model
To evaluate language models, we use it to play the Shannon game:

A Russian spacecraft filled in for the _____

At each point, there is some true probability distribution $P(w_i|w_1,\ldots,w_{i-1})$ and a probability $Q(w_i|w_1,\ldots,w_{i-1})$ predicted by our model.

We can use the KL divergence $D(P\|Q)$ as a measure of the quality of a language model $Q$.
Recall that the cross entropy between two distributions is related to their KL divergence:

\[
H_P(X, Q) = H_P(X) + D(P || Q)
\]

\[
= - \sum_x p(x) \log p(x) + \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x)
\]

\[
= - \sum_x p(x) \log q(x)
\]
Cross entropy

- We want to know the average per word cross entropy between the true distribution $P$ and our model $Q$ for language $L$:

$$H(L, Q) = -\lim_{n \to \infty} \frac{1}{n} \sum_{w_{1:n}} p(w_{1:n}) \log q(w_{1:n})$$

$$= -\lim_{n \to \infty} \frac{1}{n} E[\log q(w_{1:n})]$$

$$\approx -\lim_{n \to \infty} \frac{1}{n} \log q(w_{1:n})$$
Cross entropy

- Given a sample of text \( w_1, \ldots, w_n \), this estimate of the cross entropy of a language model (sometimes called the logprob) is:

\[
LP(Q) = -\frac{1}{n} \log q(w_{1:n})
\]

\[
= -\frac{1}{n} \log \prod_{i=1}^{n} q(w_n|w_1, \ldots, w_{n-1})
\]

\[
= -\frac{1}{n} \sum_{i}^{n} \log q(w_n|w_1, \ldots, w_{n-1})
\]

- In playing the Shannon game, this is the average number of guesses you will have to make given your imperfect model \( Q \).

- The perplexity \( PP \) is simply \( 2^{LP} \), (assuming log\(_2\)).
Parameter estimation

• Back to the MLE – what’s the probability assigned to a word in a context in which it did not occur in the training data?

\[
P(w_n | w_{n-2}, w_{n-1}) = \frac{P(w_{n-2}, w_{n-1}, w_n)}{P(w_{n-2}, w_{n-1})} = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})} = 0
\]

• The likelihood of a corpus which contains this will then be 0, and the cross entropy (and perplexity) is $\infty$!

• The MLE uses only information from the training text – how can we do better?
Parameter estimation

- A simple trick is to increase the size of the history equivalence classes

- Replace words occurring once in the training data (*hapax legomena*) with a designated symbol OOV or UNK

- Replace any unknown words in the test sequence (i.e., words which occur zero or one time in the training data) with the same symbol

- We never get zero probabilities this way, but we end up assigning all OOV’s the same probability

- Curse of Zipf