Statistical Natural Language Processing

- The explosion of on-line text allows us to gather statistical data about language use. The most common applications include part-of-speech tagging and parsing.

  This usually requires a pre-tagged or pre-parsed training corpus!

- Statistical data can be gathered for explicit knowledge (e.g., common expressions), search control knowledge (e.g., to guide a chart parser), or preferences (e.g., the likelihoods of alternatives).

- Statistical methods can also be used to gather domain-specific knowledge to reveal strong preferences for a domain.

  Ex: in transportation texts the word “train” is usually a noun, but in educational texts “train” is usually a verb.

The Basics of Probability Theory

- Intuitively, the probability of an event is the likelihood that it will occur.

- Probabilities are usually described in terms of a **random variable** that can range over a set of values.

  Ex: You want to measure the likelihood that a tossed coin would land tail-side up. We would use a random variable TOSS and write the probability as P(TOSS=h).

  There are two possibilities, P(TOSS=h) = 1/2 = 0.5

- A **conditional probability** is the likelihood of an event X given that a condition Y is true, expressed as P(X | Y).

Conditional Probabilities and Bayes’ Rule

Conditional probability is defined as: P(X | Y) = P(X & Y) / P(Y)

Example:
if P(Elvis seen & UFO seen) = .01
and P(UFO seen) = .02
and P(Elvis seen) = .03
then P(Elvis seen | UFO seen) = .01/.02 = .5

Using Bayes’ rule we can compute conditional probabilities:

$$P(X \mid Y) = \frac{P(Y|X) * P(X)}{P(Y)}$$

$$P(\text{UFO seen} \mid \text{Elvis seen}) = \frac{.5 * .02}{.03} = .33$$

Estimating Probabilities

- It is often impossible to know the true probabilities of events. There is no way to determine the true probabilities associated with words because we don’t have access to all text ever written!
- But we can estimate probabilities by looking at a large text collection.
- Most of the time, we use simple frequency counts and ratios to estimate probabilities. These are called *maximum likelihood estimates*.

For example, suppose the word “flies” appears 1000 times in a corpus, 700 times as a verb and 300 times as a noun.

P(verb | flies) = 700/1000 = .70
P(noun | flies) = 300/1000 = .30

N-grams

- An **N-gram** is a sequence of N items.
- The most commonly used N-grams are **unigrams** (1 item), **bigrams** (2 items) and **trigrams** (3 items).
- We can estimate the probability of an N-gram using a text corpus. If X is a specific N-gram and k is the number of distinct N-grams in the corpus (for the same value of N), then:

$$P(X_j) = \frac{freq(X_j)}{\sum_{i=1}^{k} freq(X_i)}$$

For example, $P(''dog'') = \frac{freq(''dog'')}{\sum_{i=1}^{k} freq(word_i)}$

N-gram Language Models

- An *N-gram language model* captures a probability distribution over sequences of N-grams.
- Language models using *lexical N-grams* can estimate the likelihood of a sentence: P($Word_1$ ... $Word_k$)
  Language models using *part-of-speech N-grams* can estimate the likelihood of a sequence of part-of-speech tags: P(Tag_1 ... Tag_k).
- Probabilities for adjacent N-grams are multiplied. For example:

$$P(w_1...w_k) = \prod_{i=1}^{k} P(w_i) \quad \text{(for unigrams)}$$

$$P(w_1...w_k) = \prod_{i=1}^{k} P(w_i \mid w_{i-1}) \quad \text{(for bigrams)}$$

$$P(w_1...w_k) = \prod_{i=1}^{k} P(w_i \mid w_{i-2}, w_{i-1}) \quad \text{(for trigrams)}$$

Computing Probabilities in Log Space

- Individual probabilities can be small, and the product of many probabilities can be a very small number!
- It is usually best to compute N-gram probabilities in **log space**.

  Key observation: $log_2(X * Y) = log_2(X) + log_2(Y)$

  Each individual probability is stored as a **logprob**: $log_2(p)$.
  The product of those probabilities is the sum of the logprobs.
  At the very end, the actual probability value can (and should) be restored by taking the anti-log: $2^{log(X)} = X$.
- Another helpful fact: $log_b(X) = \frac{log_c(X)}{log_c(b)}$

  For example: $log_2(X) = \frac{log_{10}(X)}{log_{10}(2)}$

Mixing N-grams

- Other ways to deal with sparse data are *back-off models* and *deleted interpolation models*.
- A *back-off model* uses the largest N-gram that has reliable statistics. For example, if the trigram estimate is reliable, it will be used. Else, the bigram estimate will be used. And if that is not reliable enough either, a unigram estimate can be used as a last resort.
- A *deleted interpolation model* allows N-grams of multiple lengths to contribute to a probability estimate. For example, trigrams, bigrams, and unigrams can be used to compute the probability of a tag sequence, using weights to determine how much each type of N-gram contributes. For example:

$$P(T_i \mid T_1...T_{i-1}) = \lambda_1 P(T_i) + \lambda_2 P(T_i \mid T_{i-1}) + \lambda_3 P(T_i \mid T_{i-2}T_{i-1})$$

Sparse Data

- Estimating probabilities works well when you have a lot of data. However, many cases occur infrequently. When many cases are not well-represented in the training data, you have a **sparse data** problem.

  Ex: the Brown corpus is a well-known corpus that contains about 1,000,000 words. But it contains only 49,000 unique words and over 40,000 of those words occur $\leq$ 5 times!

- A particularly difficult problem is when something never appears in the training data. For example, a corpus might not contain *any* instances of the word "flies" as a noun.

Laplace Smoothing (aka: Add-One Smoothing)

Idea: Add 1 to all frequency counts to pretend you've seen everything at least once and then renormalize the probability space.

Let's assume a lexical unigram model.
Let $N = \sum_{i}^{|V|} freq(w_i)$
and $V = vocabulary\ size\ (\#\ of\ unique\ terms)$

Without smoothing, $P(w_x) = \frac{freq(w_x)}{N}$

First, we compute new frequency values for each term:

$$new\_freq(w_x) = (freq(w_x) + 1) * \frac{N}{N+V}$$

and then new probability estimates: $new\_P(w_x) = \frac{new\_freq(w_x)}{N}$
This is equivalent to: $new\_P(w_x) = \frac{freq(w_x)+1}{N+V}$

Add-One Smoothing for larger N-grams

Add-one smoothing for larger N-grams works exactly the same way.

For bigrams:

Without smoothing, $P(w_n \mid w_{n-1}) = \frac{freq(w_{n-1},w_n)}{freq(w_{n-1})}$

With smoothing, $new\_P(w_n \mid w_{n-1}) = \frac{freq(w_{n-1},w_n)+1}{freq(w_{n-1})+V}$

For trigrams:

Without smoothing, $P(w_n \mid w_{n-2}, w_{n-1}) = \frac{freq(w_{n-2},w_{n-1},w_n)}{freq(w_{n-2},w_{n-1})}$

With smoothing, $new\_P(w_n \mid w_{n-2}, w_{n-1}) = \frac{freq(w_{n-2},w_{n-1},w_n)+1}{freq(w_{n-2},w_{n-1})+V}$