N-gram Models

CMSC 35100

Natural Language Processing

April 17, 2003

Roadmap

- *n*-gram models
  - Motivation
- Basic *n*-grams
  - Markov assumptions
- Coping with sparse data
  - Smoothing, Backoff
- Evaluating the model
  - Entropy and Perplexity

N-grams

- Perspective:
  - Some sequences (words/chars) are more likely than others
  - Given sequence, can guess most likely next
- Used in
  - Speech recognition
  - Spelling correction,
  - Augmentative communication
  - Other NL applications

Corpus Counts

- Estimate probabilities by counts in large collections of text/speech
- Issues:
  - Wordforms (surface) vs lemma (root)
  - Case? Punctuation? Disfluency?
  - Type (distinct words) vs Token (total)

Basic N-grams

- Most trivial: 1/#tokens: too simple!
- Standard unigram: frequency
  - # word occurrences/total corpus size
    - E.g. the=0.07; rabbit = 0.00001
  - Too simple: no context!
- Conditional probabilities of word sequences

$$P(w_1^n) = P_n(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)...P(w_n \mid w_1^n)$$
$$= \prod_{k=1} P(w_k \mid w_1^{k-1})$$

Markov Assumptions

- Exact computation requires too much data
- Approximate probability given all prior wds
  - Assume *finite* history
  - Bigram: Probability of word given 1 previous
    - First-order Markov
  - Trigram: Probability of word given 2 previous
- N-gram approximation

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

Bigram sequence $$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

Issues

- Relative frequency
  - Typically compute count of sequence
    - Divide by prefix

$$P(w_n \mid w_{n-1}) = \frac{C(w_n w_{n-1})}{C(w_{n-1})}$$

- Corpus sensitivity
  - Shakespeare vs Wall Street Journal
    - Very unnatural
- Ngrams
  - Unigram: little; bigrams: colloc; trigrams:phrase

Sparse Data Issues

- Zero-count n-grams
  - Problem: Not seen yet! Not necessarily impossible..
  - Solution: Estimate probabilities of unseen events
- Two strategies:
  - Smoothing
    - Divide estimated probability mass
  - Backoff
    - Guess higher order n-grams from lower

Smoothing out Zeroes

- Add-one smoothing
  - Simple: add 1 to all counts -> no zeroes!
  - Normalize by count and vocabulary size
- Unigrams:
  - Adjusted count: $c_i^* = (c_i + 1)\frac{N}{N+V}$
  - Adjusted probability $p_i^* = \frac{(c_i + 1)}{N+V}$
- Bigrams:
  - Adjusted probability $p^*(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$
- Problem: Too much weight on (former) zeroes

Estimating the Zeroes' Mass

- Witten-Bell discounting
  - Idea: Use count of things you've seen to estimate count of things you haven't
- Estimate probability mass of zeroes
  - Guess same as # of 1st time n-grams = types
- Total mass: $\sum_{i:c_i=0} p_i^* = \frac{N}{N+T}$
- Unigrams: $p^*_{i:c_i=0} = \frac{N}{Z(N+T)}; p^*_{i:c_i>0} = \frac{c_i}{N+T}$
- Bigrams: $Z(w_x) = \sum_{i:c(w_x w_i)=0} 1$

$$p^*(w_i \mid w_x) = \frac{T(w_x)}{Z(w_x)(N+T(w_x))}$$

$$p^*(w_i \mid w_x) = \frac{c(w_x w_i)}{c(w_x)+T(w_x)}$$

Good-Turing Smoothing

- Re-estimate based on n-grams with higher counts:
  - N1 = # bigrams with count 1: $N_c = \sum_{b:c(b)=c} 1$
  - E.g. estimate counts of bigrams that didn't occur based on those that occurred once
  - Assume that counts above some threshold are reliable
    - Basic,

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

$$c^* = c, c > k$$

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}$$

Backoff

- Idea: If no tri-grams, estimate with bigrams
- E.g. $\hat{P}(w_n \mid w_{n-2}w_{n-1}) = P(w_n \mid w_{n-2}w_{n-1}), ifC(w_{n-2}w_{n-1}w_n) > 0$
- $= \alpha_1 P(w_n \mid w_{n-1}), ifC(w_{n-2}w_{n-1}w_n) = 0 \,\&\, C(w_{n-1}w_n) > 0$
- $= \alpha_2 P(w_n), o.w.$
- Deleted interpolation:
  - Replace α's with λ's that are trained for word contexts

Evaluating n-gram models

- Entropy & Perplexity
  - Information theoretic measures
  - Measures information in grammar or fit to data
  - Conceptually, lower bound on # bits to encode
- Entropy: H(X): X is a random var, *p:* prob fn

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

  - E.g. 8 things: number as code => 3 bits/trans
  - Alt. short code if high prob; longer if lower
    - Can reduce
- Perplexity: 2^H
  - Weighted average of number of choices

Entropy of a Sequence

- Basic sequence

$$\frac{1}{n} H(W_1^n) = -\frac{1}{n} \sum_{W_1^n \in L} p(W_1^n) \log_2 p(W_1^n)$$

- Entropy of language: infinite lengths
  - Assume stationary & ergodic

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \sum_{W \in L} p(w_1, ..., w_n) \log p(w_1, ..., w_n)$$

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(w_1, ..., w_n)$$

Cross-Entropy

- Comparing models
  - Actual distribution unknown
  - Use simplified model to estimate
    - Closer match will have lower cross-entropy

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n} \sum_{W \in L} p(w_1,...,w_n) \log m(w_1,...,w_n)$$

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n} \log m(w_1,...,w_n)$$

Entropy of English

- Shannon’s experiment
  - Subjects guess strings of letters, count guesses
  - Entropy of guess seq = Entropy of letter seq
  - 1.3 bits; Restricted text
- Build stochastic model on text & compute
  - Brown computed trigram model on varied corpus
  - Compute (pre-char) entropy of model
  - 1.75 bits