N-gram Models

Roadmap

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

Information & Communication

- Shannon (1948)
- Perspective:
  - Message *selected* from possible messages
    - Number (or function of #) of messages measure of information produced by selecting that message
    - Logarithmic measure
      - Base 2: # of bits

Probabilistic Language Generation

- *Coin-flipping models*
  - A sentence is generated by a randomized algorithm
    - The generator can be in one of several "states"
    - Flip coins to choose the next state.
    - Flip other coins to decide which letter or word to output

Shannon's Generated Language

- 1. *Zero-order approximation:*
  - XFOML RXKXRJFFUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD
- 2. *First-order approximation:*
  - OCRO HLI RGWR NWIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH RBL
- 3. Second-order approximation:
  - ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIND ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Shannon’s Word Models

- 1. *First-order approximation:*
  - REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE
- 2. *Second-order approximation:*
  - THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

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
  - Language Identification
  - Information Retrieval

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

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

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

Backoff

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

Toward an Information Measure

- Knowledge: event probabilities available
- Desirable characteristics: H(p1,p2,…,pn)
  - Continuous in pi
  - If pi equally likely, monotonic increasing in n
    - If equally likely, more choice w/more elements
  - If broken into successive choices, weighted sum
- Entropy: H(X): X is a random var, *p:* prob fn

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

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

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

- Perplexity: 2^H
  - Weighted average of number of choices

Computing Entropy

- Picking horses (Cover and Thomas)
- Send message: identify horse - 1 of 8
  - If all horses equally likely, p(i) = 1/8

$$H(X) = -\sum_{i=1}^{8} 1/8 \log 1/8 = -\log 1/8 = 3 bits$$

  - Some horses more likely:
    - 1: ½; 2: ¼; 3: 1/8; 4: 1/16; 5,6,7,8: 1/64

$$H(X) = -\sum_{i=1}^{8} p(i) \log p(i) = 2 bits$$

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

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

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

Cross-Entropy

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

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

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

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

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

$$H(p) \le H(p,m)$$

Perplexity Model Comparison

- Compare models with different history
- Train models
  - 38 million words – Wall Street Journal
- Compute perplexity on held-out test set
  - 1.5 million words (~20K unique, smoothed)
- N-gram Order | Perplexity
  - Unigram | 962
  - Bigram | 170
  - Trigram | 109

Does the model improve?

- Compute probability of data under model
  - Compute perplexity
- Relative measure
  - Decrease toward optimum?
  - Lower than competing model?

Iter	0	1	2	3	4	5	6	9	10
P(data)	9^-19	1^-16	2^-16	3^-16	4^-16	4^-16	4^-16	5^-16	5^-16
Perplex	3.393	2.95	2.88	2.85	2.84	2.83	2.83	2.8272	2.8271

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 (per-char) entropy of model
  - 1.75 bits

Using N-grams

- Language Identification
  - Take text samples
    - English, French, Spanish, German
  - Build character tri-gram models
  - Test Sample: Compute maximum likelihood
    - Best match is chosen language
- Authorship attribution