CS 4248
Natural Language Processing

Professor NG Hwee Tou
Department of Computer Science
School of Computing
National University of Singapore
nght@comp.nus.edu.sg

Chapter 4: N-Grams

- Word prediction
  - “Good morning, ladies and …”
- Probability of a word sequence
  - “all of a sudden I notice three guys standing on the sidewalk”
  - “all I of notice sidewalk three sudden guys on standing a the”
  - The second sentence has a much lower probability

Language Model

- Language Model (LM): statistical model of word sequences
- n-gram: Use the previous $n-1$ words to predict the next word

$$P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

$$P(w_1, w_2, \ldots, w_{n-1}, w_n)$$

N-Grams

- Applications
  - speech recognition
  - context-sensitive spelling error detection and correction
    - “He is trying to *fine* out.”
    - “The design *an* construction will take a year.”
  - machine translation
    - He briefed reporters on the main contents of the statement
    - He briefed to reporters on the chief contents of the statement

Simple (Unsmoothed) N-Grams

- Task: Estimating the probability of a word
- First attempt:
  - Suppose there is no corpus available
  - Use uniform distribution
  - Assume:
    - word types = V (e.g., 100,000)

$$P(w) = \frac{1}{V}$$

Simple (Unsmoothed) N-Grams

- Task: Estimating the probability of a word
- Second attempt:
  - Suppose there is a corpus
  - Assume:
    - word tokens = N
    - # times w appears in corpus = $C(w)$

$$P(w) = \frac{C(w)}{N}$$

Simple (Unsmoothed) N-Grams

- Task: Estimating the probability of a word
- Third attempt:
  - Suppose there is a corpus
  - Assume a word depends on its $n-1$ previous words

$$P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

Simple (Unsmoothed) N-Grams

$$P(w_1, w_2, \ldots, w_{N-1}, w_N)$$

$$= \quad P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1, w_2) \cdot \ldots \cdot P(w_N \mid w_1, \ldots, w_{N-1})$$

$$= \quad \prod_{k=1}^{N} P(w_k \mid w_1, \ldots, w_{k-1})$$

Simple (Unsmoothed) N-Grams

n-gram approximation:
w_k only depends on its previous $n-1$ words ($k \geq n$)

$$P(w_k \mid w_1, \ldots, w_{k-1}) \approx P(w_k \mid w_{k-(n-1)}, \ldots, w_{k-2}, w_{k-1})$$

bigram ($n=2$): $P(w_k \mid w_1, \ldots, w_{k-1}) \approx P(w_k \mid w_{k-1})$

trigram ($n=3$): $P(w_k \mid w_1, \ldots, w_{k-1}) \approx P(w_k \mid w_{k-2}, w_{k-1})$

Markov assumption

Bigram Approximation

$$P(w_1, w_2, \ldots, w_{N-1}, w_N) \approx \prod_{k=1}^{N} P(w_k \mid w_{k-1})$$

Example:

P(<s> i want english food </s>)
= P(i | <s>) P(want | i) P(english | want) P(food | english) P(</s> | food)

<s>: a special word meaning “start of sentence”
</s>: a special word meaning “end of sentence”

Note on Practical Problem

- Multiplying many probabilities results in a very small number and can cause numerical underflow
- Use logprob instead in the actual computation

$$\log(p_1 \cdot p_2) = \log p_1 + \log p_2$$

Estimating N-Gram Probability

Maximum Likelihood Estimate (MLE)

$$P(w_2|w_1) = \frac{C(w_1 w_2)}{\sum_w C(w_1 w)} = \frac{C(w_1 w_2)}{C(w_1)} \qquad \text{(bigram)}$$

$$P(w_n|w_1, w_2, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_{n-1}, w_n)}{C(w_1, \dots, w_{n-1})} \qquad \text{(n-gram)}$$

Estimating Bigram Probability

Example:

C(to eat) = 860

C(to) = 3256

$$P(\text{eat} \mid \text{to}) = \frac{C(\text{to eat})}{C(\text{to})} = \frac{860}{3256} = 0.26$$

Training and Test Sets

- Divide a corpus into:
  - Training set
  - Development set (held-out set)
  - Test set
- Typical breakdown: 80%, 10%, 10% respectively
- Train a model on the training set and evaluate on the test set
- Development set used to tune parameters
- Better predictive power if training set is closer to the test set (e.g., the same genre)

Unknown Words

- Closed vocabulary
  - Vocabulary is fixed
  - Test set only contains words from this vocabulary
  - No unknown words
- Open vocabulary
  - Test set contains unknown words (out of vocabulary (OOV) words)
  - OOV rate: percentage of OOV words in the test set

Unknown Words

- Add a pseudo-word <UNK>
- Choose a vocabulary (word list) that is fixed in advance
- Convert any OOV word in the training set to <UNK>
- Estimate the probability for <UNK> (treat <UNK> like a regular word in the training set)

Evaluating N-Grams: Perplexity

- Extrinsic evaluation
  - Measure the quality of a language model by embedding it in an application (e.g., speech recognition) and measure the total performance of the application
  - Expensive, time-consuming approach

Evaluating N-Grams: Perplexity

- Intrinsic evaluation
  - Measure the quality of a language model independent of any application
  - Cheaper, quicker alternative
  - Perplexity: intrinsic evaluation metric
  - Improvement in perplexity often correlates with improvement in speech recognition performance

Perplexity

- A better language model better predicts the test data and assigns a higher probability to the test data

$$PP = m(w_1, \ldots, w_n)^{-\frac{1}{n}} = \frac{1}{\sqrt[n]{m(w_1, \ldots, w_n)}}$$

- A better language model has a lower perplexity

Perplexity

- For bigram language model:

$$PP = \frac{1}{\sqrt[n]{m(w_1, \ldots, w_n)}} = \frac{1}{\sqrt[n]{\prod_{k=1}^{n} P(w_k \mid w_{k-1})}}$$

Perplexity

- Consider a string of digits (0 – 9) occurring with equal probability:

$$PP = \frac{1}{\sqrt[n]{m(w_1, \ldots, w_n)}} = \frac{1}{\sqrt[n]{\left(\frac{1}{10}\right)^n}} = 10$$

- Weighted average number of choices a random variable has to make (weighted average branching factor of a language)

Perplexity

- Training data: 38 million words of WSJ (Wall Street Journal)
- Test data: 1.5 million words of WSJ
- Unigram perplexity = 962
- Bigram perplexity = 170
- Trigram perplexity = 109

Smoothing

- Any particular training corpus is finite
- Some perfectly acceptable n-grams are bound to be missing from the training corpus
- Sparse data problem
- Deal with zero probability

Smoothing

- Smoothing
  - Reevaluating zero probability n-grams and assigning them non-zero probability
- Also called Discounting
  - Lowering non-zero n-gram counts in order to assign some probability mass to the zero n-grams

Add-One Smoothing for Bigram

$$C(w_0) = C(w_0 w_1) + \cdots + C(w_0 w_V)$$

$$C(w_0) + V = \{C(w_0 w_1) + 1\} + \cdots + \{C(w_0 w_V) + 1\}$$

$$1 = \frac{C(w_0) + V}{C(w_0) + V} = \frac{C(w_0 w_1) + 1}{C(w_0) + V} + \cdots + \frac{C(w_0 w_V) + 1}{C(w_0) + V}$$

$$1 = \frac{C(w_0 w_1) + 1}{C(w_0) + V} + \cdots + \frac{C(w_0 w_V) + 1}{C(w_0) + V}$$

$$P(w \mid w_0) = \frac{C(w_0 w) + 1}{C(w_0) + V}$$

Smoothed Count

- Smoothed bigram count (adjusted bigram count)

$$= C^*(w_0 w)$$

$$P(w \mid w_0) = \frac{C(w_0 w) + 1}{C(w_0) + V} = \frac{C^*(w_0 w)}{C(w_0)}$$

$$C^*(w_0 w) = \{C(w_0 w) + 1\} \times \frac{C(w_0)}{C(w_0) + V}$$

Discount

- Discount d

$$d = \frac{C^*(w_0 w)}{C(w_0 w)}$$

$$d = \frac{C(w_0 w) + 1}{C(w_0 w)} \times \frac{C(w_0)}{C(w_0) + V}$$

Witten-Bell Smoothing for Bigram

$$C(w_0) = C(w_0 w_1) + \cdots + C(w_0 w_T)$$

$$T \equiv T(w_0) \text{ is the number of distinct word types following } w_0$$

$$C(w_0 w_1) > 0, \ldots, C(w_0 w_T) > 0$$

$$T(w_0) \text{ is the number of seen bigram types following } w_0$$

$$Z(w_0) \text{ is the number of unseen bigram types following } w_0$$

$$T(w_0) + Z(w_0) = V$$

$$C(w_0) + T(w_0) = C(w_0 w_1) + \cdots + C(w_0 w_T) + T(w_0)$$

$$1 = \frac{C(w_0) + T(w_0)}{C(w_0) + T(w_0)} = \frac{C(w_0 w_1)}{C(w_0) + T(w_0)} + \cdots + \frac{C(w_0 w_T)}{C(w_0) + T(w_0)} + \frac{T(w_0)}{C(w_0) + T(w_0)}$$

$$P(w \mid w_0) = \frac{C(w_0 w)}{C(w_0) + T(w_0)} \quad \text{if } C(w_0 w) > 0$$

$$P(w \mid w_0) = \frac{T(w_0)}{Z(w_0)(C(w_0) + T(w_0))} \quad \text{if } C(w_0 w) = 0$$

Witten-Bell Smoothing for Bigram

- Consider diversity of predicted words
- Example (from Statistical Machine Translation, Philipp Koehn)
  - spite and constant both occur 993 times in Europarl corpus (C(spite) = C(constant) = 993)
  - 9 different words follow spite (T(spite) = 9)
    - of (979 times)
  - 415 different words follow constant (T(constant) = 415)
    - and (42 times)
    - concern (27 times)
    - pressure (26 times)
    - Long tail of 268 different words (1 time each)

Witten-Bell Smoothing for Bigram

- $\frac{T(spite)}{C(spite)+T(spite)} = \frac{9}{993+9} = 0.009$
- $\frac{T(constant)}{C(constant)+T(constant)} = \frac{415}{993+415} = 0.295$
- More likely to see some new bigram that starts with constant

Interpolation

- Higher order n-grams (e.g., trigrams) and lower-order n-grams (e.g., bigrams) have different strengths and weaknesses
  - Higher-order n-grams utilize more context, but have sparser counts
  - Lower-order n-grams consider limited context, but have more robust counts
- Combine them

Interpolation

Trigrams:

$$\hat{P}(w_0 \mid w_{-2} w_{-1}) =$$
$$\lambda_1 P(w_0 \mid w_{-2} w_{-1}) + \lambda_2 P(w_0 \mid w_{-1}) + \lambda_3 P(w_0)$$

Bigrams:

$$\hat{P}(w_0 \mid w_{-1}) = \lambda_1 P(w_0 \mid w_{-1}) + \lambda_2 P(w_0)$$

$$\sum_i \lambda_i = 1$$

Interpolation

- To set λ_i:
  - Divide a corpus into training set and development (held-out) set
  - Gather counts from training set
  - Select λ_is that maximize the probability of the development set

Backoff

- If no examples of a particular trigram $w_{-2}w_{-1}w_0$ to compute $P(w_0|w_{-2}w_{-1})$, estimate by using bigram probability $P(w_0|w_{-1})$
- If no examples of a particular bigram $w_{-1}w_0$ to compute $P(w_0|w_{-1})$, estimate by using unigram probability $P(w_0)$
- Example:
  – Have not seen “Scottish beer drinkers”, “Scottish beer eaters”
  – Backoff to “beer drinkers”, “beer eaters”

Backoff for Bigram

$$\hat{P}(w_0 \mid w_{-1}) = \begin{cases} \widetilde{P}(w_0 \mid w_{-1}) & \text{if } C(w_{-1}w_0) > 0 \\ \alpha(w_{-1}) \cdot \widetilde{P}(w_0) & \text{if } C(w_{-1}w_0) = 0 \end{cases}$$

$\widetilde{P}(w_0 \mid w_{-1}), \widetilde{P}(w_0)$: discounted probability

Backoff for Bigram

$$\sum_{w_0} \hat{P}(w_0 \mid w_{-1}) = 1$$

$$\sum_{w_0 : C(w_{-1} w_0) > 0} \widetilde{P}(w_0 \mid w_{-1}) + \sum_{w_0 : C(w_{-1} w_0) = 0} \alpha(w_{-1}) \cdot \widetilde{P}(w_0) = 1$$

$$\alpha(w_{-1}) = \frac{1 - \sum_{w_0 : C(w_{-1} w_0) > 0} \widetilde{P}(w_0 \mid w_{-1})}{\sum_{w_0 : C(w_{-1} w_0) = 0} \widetilde{P}(w_0)}$$

$$\alpha(w_{-1}) = \frac{1 - \sum_{w_0 : C(w_{-1} w_0) > 0} \widetilde{P}(w_0 \mid w_{-1})}{1 - \sum_{w_0 : C(w_{-1} w_0) > 0} \widetilde{P}(w_0)}$$

Kneser-Ney Smoothing for Bigram

- Two innovations:
  - Absolute discounting for seen bigrams

$$\frac{C(w_{-1}w_0) - D}{C(w_{-1})} \qquad 0 \leq D \leq 1$$

  - Diversity of histories: Backoff distribution based on number of unique words preceding w_0 (words that have appeared in more contexts are more likely to appear in some new context as well)

$$\frac{|\{w_{-1} : C(w_{-1}w_0) > 0\}|}{\sum_w |\{w_{-1} : C(w_{-1}w) > 0\}|}$$

Kneser-Ney Smoothing for Bigram

- Count of histories for a word w (number of unique words preceding w):

$$|\{w_{-1}: C(w_{-1}w) > 0\}|$$

- Replace raw count of w with count of histories of w

Kneser-Ney Smoothing for Bigram

- Example (from SMT, Philipp)
  - Consider the word York
  - Occurs 477 times in Europarl (fairly frequent)
  - As frequent as foods, indicates, and providers
  - However, York almost always directly follows New (473 times)
  - Backoff to unigram from bigram
  - York unlikely to be the second word in unseen bigram, i.e., York should have low probability in backoff

Kneser-Ney Smoothing for Bigram

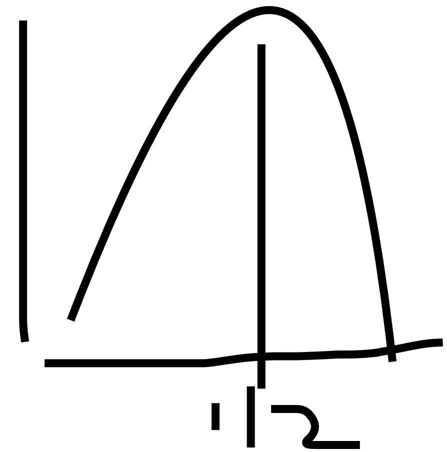
$$P_{KN}(w_0 \mid w_{-1}) = \begin{cases} \dfrac{C(w_{-1}w_0) - D}{C(w_{-1})} & \text{if } C(w_{-1}w_0) > 0 \\ \alpha(w_{-1}) \cdot \dfrac{|\{w'_{-1} : C(w'_{-1}w_0) > 0\}|}{\sum_w |\{w'_{-1} : C(w'_{-1}w) > 0\}|} & \text{if } C(w_{-1}w_0) = 0 \end{cases}$$

Kneser-Ney smoothing has been found to be the most effective smoothing method for n-gram language modeling

Entropy

- Measure of uncertainty
- Entropy H(X) of a random variable X:

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



- Measured in bits
- Number of bits to encode information in the optimal coding scheme

Example 1

$$P(\text{horse1}) = P(\text{horse2}) = \cdots = P(\text{horse8}) = \frac{1}{8}$$

$$H = -\sum_{x} p(x)\log_2 p(x) = -8(\frac{1}{8}\log_2 \frac{1}{8}) = 3 \text{ bits}$$

Coding scheme :

horse1 : 001

horse2 : 010

$\cdots$

horse7 : 111

horse8 : 000

Example 2

x	horse1	horse2	horse3	horse4	horse5	horse6	horse7	horse8
p(x)	1/2	1/4	1/8	1/16	1/64	1/64	1/64	1/64

$$\begin{aligned} H &= -\sum_x p(x)\log_2 p(x) \\ &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - 4\left(\frac{1}{64}\log_2\frac{1}{64}\right) \\ &= 2 \text{ bits} \end{aligned}$$

sum(p *(-logp))
-log(p) = no. of bits

Coding scheme :

1*1/2 + 2*1/4 + 3*1/8 + 4*1/16 + 6*1/64*4 = 2

x	horse1	horse2	horse3	horse4	horse5	horse6	horse7	horse8
code	0	10	110	1110	111100	111101	111110	111111

Short encodings for more probable horses

Entropy of a Sequence

- Entropy of a random variable ranging over all finite sequences of words of length n in a language L:

$$H(w_1, w_2, \ldots, w_n) = -\sum_{W_1^n \in L} p(W_1^n) \log_2 p(W_1^n)$$

- Entropy rate (per-word entropy):

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

Entropy of a Language

$$
\begin{aligned}
H(L) &= \lim_{n\to\infty} \frac{1}{n} H(w_1, \ldots, w_n) \\
&= \lim_{n\to\infty} -\frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log_2 p(w_1, \ldots, w_n)
\end{aligned}
$$

By Shannon-McMillan-Breiman theorem:

$$
H(L) = \lim_{n\to\infty} -\frac{1}{n} \log_2 p(w_1, \ldots, w_n)
$$

Cross Entropy

- Used for comparing two language models
- p: Actual probability distribution that generated some data
- m: A model of p (approximation to p)
- Cross entropy of m on p:

$$H(p,m) = \lim_{n \to \infty} -\frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log_2 m(w_1, \ldots, w_n)$$

Cross Entropy

- By Shannon-McMillan-Breiman theorem:

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n}\log_2 m(w_1,\ldots,w_n)$$

- Property of cross entropy:

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

- Difference between H(p,m) and H(p) is a measure of how accurate model m is
- The more accurate a model, the lower its cross-entropy

Perplexity

- Perplexity (PP) = 2^H

$$PP = 2^{-\frac{1}{n}\log_2 m(w_1,\ldots,w_n)} = m(w_1,\ldots,w_n)^{-\frac{1}{n}} = \frac{1}{\sqrt[n]{m(w_1,\ldots,w_n)}}$$