Computational Linguistics 1

CMSC/LING 723, LBSC 744

Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 6: 20 September 2011

Homework Agenda

- HW0 – graded
  - http://grades.cs.umd.edu
  - Comments from the TA
- HW1 – due today!
  - Observations
- HW2 – assigned Thursday, due next Thursday 9/29
- Questions, comments, concerns?
- Language Models
- Part-of-speech Tagging

Agenda

- Language Models
  - Higher n-gram models
  - Smoothing
    - Combining estimators
    - Backoff
    - OOVs
  - Evaluating LMs
- Part-of-speech Tagging

Higher n-gram LM Generators

- Generated by a unigram LM:
  - because regime more likely where clothing for racial 's politicians % . who
  - 're <unk> with in , human economic some into unit Clark <unk> for 's to . They that securities East % compared <unk> As The to to in Ivan its 7.20 at measures 17 seven prediction on 43-foot in a . the and Lipton Most % precarious in
- Generated by a bigram LM:
  - But he has eaten .
  - When it first time it is to issue
  - In Direct disaster closed yesterday 's $ 2,000 orders in a percentage of fighting quality output.
- Generated by a trigram LM:
  - Imperial troublesome Oakland .) .
  - So what 's capital stock market .
  - The company noted that the state can be discussed researchers have run last long impasse between 1986 and end of last year .
- State-of-the-art

Higher n-gram LM Generators

- Generated by a (smoothed) trigram LM:
  - Imperial troublesome Oakland .) .
  - So what 's capital stock market .
  - The company noted that the state can be discussed researchers have run last long impasse between 1986 and end of last year .
- Generated by an unsmoothed trigram LM:
  - He adds that spending on the <unk> are beginning to produce a staunchly conservative younger generation .
  - In Japan , which would have to be proved right – he tried to rally support in the junk bond market .
- So why smooth?
  - LMs as acceptors

Agenda

- Language Models
  - Smoothing
    - Combining estimators
    - Backoff
    - OOVs
  - Evaluating LMs
- Part-of-speech Tagging

Combining Estimators

- Three major combination techniques:
  - Simple Linear Interpolation of MLEs
  - Katz Backoff
  - Kneser-Ney Smoothing

Linear MLE Interpolation

- Mix higher n-gram models with lower n-gram models
  - To offset sparsity
-

$$P(w_k|w_{k-2}w_{k-1}) = \lambda_1 P(w_k|w_{k-2}w_{k-1}) + \lambda_2 P(w_k|w_{k-1}) + \lambda_3 P(w_k)$$

$$0 <= \lambda_i <= 1 \qquad \sum_i \lambda_i = 1$$

Backoff Models

- Consult higher n-gram models first, then if counts are 0, back off to a lower-order model (instead of consulting all models at the same time)
- Continue "backing off" until you reach a model that has non-zero counts
- Need to incorporate discounting as a part of the algorithm
- Because if we back off to a lower-order model without taking something from the higher-order models, we are adding extra mass!

Katz Backoff

Given a trigram "*x y z*"

Why P_{GT} *instead of* P_{MLE}? To reserve probability for lower-order models.

$$P_{katz}(z|x,y) = \begin{cases} P_{GT}(z|x,y), & \text{if } C(x,y,z) > 0 \\ \alpha(x,y)P_{katz}(z|y), & \text{otherwise} \end{cases}$$

$$P_{katz}(z|y) = \begin{cases} P_{GT}(z|y), & \text{if } C(y,z) > 0 \\ \alpha(y)P_{GT}(z), & \text{otherwise} \end{cases}$$

Why α's? So lower-order models' mass sums to what we stole by discounting.

Absolute & Kneser-Ney Smoothing

- Observation:
  - Average Good-Turing discount for $r \geq 3$ is largely constant over r
  - So, why not simply subtract a fixed discount D (≤1) from non-zero counts?
- Absolute Discounting: discounted bigram model, back off to MLE unigram model
- Kneser-Ney: Interpolate discounted model with a special "continuation" unigram model

Kneser-Ney Smoothing

- Intuition
  - Lower order model important only when higher order model is sparse
  - Should be optimized to perform in such situations
- Example
  - C(Los Angeles) = C(Angeles) = M; M is very large
  - "Angeles" always and only occurs after "Los"
  - Unigram MLE for "Angeles" will be high and a normal backoff algorithm will likely pick it in any context
  - It shouldn't, because "Angeles" occurs with only a single context in the entire training data

Kneser-Ney Smoothing

- Kneser-Ney: Interpolate discounted model with a special "continuation" unigram model
  - Based on appearance of unigrams in different contexts
  - Excellent performance, state of the art

$$P_{KN}(w_k|w_{k-1}) = \frac{C(w_{k-1}w_k) - D}{C(w_{k-1})} + \beta(w_k)P_{CONT}(w_k)$$

$$P_{CONT}(w_i) = \frac{N(\bullet\ w_i)}{\sum_{w'} N(\bullet\ w')}$$

$N(\bullet\ w_i)$ = number of different contexts w_i has appeared in

- Why interpolation, not backoff?

Modeling OOVs

- Take vocabulary list, truncate at some reasonable number of words
  - Or frequency of words: i.e., remove words that occur fewer than 5 times
- During training:
  - Consider any words that don't occur in this list as unknown or out of vocabulary (OOV) words
  - Replace all OOVs with the special word <UNK>
  - Treat <UNK> as any other word to count and estimate probabilities
- During testing:
  - Replace unknown words with <UNK> and use LM
  - Test set characterized by OOV rate (percentage of OOVs)

Better Modeling of OOVs?

- Orthography
  - -ing words vs -ion words
  - stemming
- Surrounding context
  - Previous word, previous two words
  - Next word, next two words
  - Sentence position

Agenda

- Language Models
  - Smoothing
    - Combining estimators
    - Backoff
    - OOVs
  - Evaluating LMs: Perplexity
- Part-of-speech Tagging

Evaluating LMs

- Why evaluate LMs?
  - For profit!
- Intrinsic vs extrinsic evaluation
- Extrinsic
  - If I use LM_1 in my MT pipeline, do I do better than if I use LM_2?
- Intrinsic: Perplexity
  - Evaluate against a test sentence
  - "How surprised are you on average by what comes next in the sentence?"
  - Lower is better. (Less surprised/better predictor.)

Computing Perplexity

- Given testset W with words $w_1, \ldots, w_N$
- Treat entire test set as one word sequence
- Perplexity is defined as the probability of the entire test set normalized by the number of words

$$PP(T) = P(w_1, \ldots, w_N)^{-1/N}$$

- Using the probability chain rule and (say) a bigram LM, we can write this as

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- A lot easer to do with log probs!

Practical Evaluation

- Typical range of perplexities on English text is 50-1000
- Closed vocabulary testing yields much lower perplexities
- Testing across genres yields higher perplexities
- Can only compare perplexities if the LMs use the same vocabulary

Order	Unigram	Bigram	Trigram
PP	962	170	109

Training: N=38 million, V~20000, open vocabulary, Katz backoff where applicable
Test: 1.5 million words, same genre as training

Typical "State of the Art" LMs

- Training
  - N = 10 billion words, V = 300k words
  - 4-gram model with Kneser-Ney smoothing
- Testing
  - 25 million words, OOV rate 3.8%
  - Perplexity ~50
- For MT systems at UMD
  - 5-gram model with Kneser-Ney smoothing
  - Computationally, required more memory than we had!

Agenda: LM Summary

- Language Models
  - Assign probabilities to sequences of tokens
- N-gram language models
  - Consider only limited histories
- Data sparsity
  - Smoothing to the rescue!
  - Variations on a theme: different techniques for redistributing probability mass
  - Important: make sure you still have a valid probability distribution!
- Evaluating LMs

Agenda

- Language Models
  - Smoothing
    - Combining estimators
    - Backoff
    - OOVs
  - Evaluating LMs: Perplexity
- Part-of-speech Tagging

Part-of-speech (POS) Tagging

- "Classes" of words
- 8 parts of speech: noun, verb, pronoun, preposition, adverb, conjunction, participle, article
  - Verbs are actions
  - Adjectives are properties
  - Nouns are things
- Mad Libs??

Better Modeling of OOVs?

- Orthography
  - -ing words vs -ion words
  - stemming
- Surrounding context
  - Previous word, previous two words
  - Next word, next two words
  - Sentence position
- What happens if we add POS-tag information?

How do we define POS?

- (Next time!!)