Ling/CSE 472: Introduction to Computational Linguistics

4/23/12
N-grams continued + look-back & review

Overview

- Review: simple n-grams
- Smoothing
- Interpolation
- Backoff
- Practical Issues: Toolkits & Data Formats
- Class-based N-grams
- LM adaptation
- Longer-distance information
- Review/lookback

N-gram basics

- N-gram models are a way of modeling the probability of a string of words, or the probability of the N+1st word being w given words 1-N.
- Usually in comparison to something ... for example?
- Ideally:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1})$$

- Approximated as (bigram version):

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1})$$

N-grams and linguistic knowledge

- Is an n-gram model a grammar?
- What kinds of information about a language does it capture?
- What kinds of information about a language does it miss?

Calculating simple (unsmoothed) n-grams

- Bigram probability for a word *y* given a previous word *x*:
- Out of all the times you saw *x*, in what percentage was it followed by *y*?

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

- What’s wrong with this?
- How can it be improved?

Solutions

- Smoothing: redistribute probability mass from seen to unseen n-grams
- Backoff: Use lower-order n-grams when higher-order ones aren’t available
- Interpolation: Use lower-order and higher-order ones together, with weights

Smoothing

- Add-one smoothing: Before normalizing the counts, add one to every possible n-gram (given vocabulary + <UNK>)
  - What’s wrong with this?
- Simple Good-Turing Discounting: Use the count of things observed only once (*hapax legomena*) to estimate the count of the unseen
  - *missing mass* = P(things with freq 0 in training) = hapaxes/all items

Simple Linear Interpolation

- Combine different order N-grams by linear interpolation

$$\begin{aligned}\hat{P}(w_n \mid w_{n-2}w_{n-1}) \quad = \quad & \lambda_1 P(w_n \mid w_{n-2}w_{n-1}) \\ & \lambda_2 P(w_n \mid w_{n-1}) \\ & \lambda_3 P(w_n)\end{aligned}$$

- Lambdas must sum to 1. Why?

- How are lambdas set?

Backoff

- Intuition: Use information from lower-order n-grams only if higher-order ones aren’t there

- Because probabilities must sum to 1 over whole model, use discounting to get revised probabilities for each n-gram

Practical Issues

- N-gram probabilities get problematic for computation (underflow) because they are so small

- Solution: convert to log probabilities, changing multiplication to addition and working with numbers that aren’t so small

- Toolkits: This has all been implemented already, so you don’t need to reimplement.

    - SRILM: http://www.speech.sri.com/projects/srilm/

    - Already on patas: /NLP_TOOLS/ml_tools/lm/srilm/latest

Class-based N-grams

- Looking at sequences of word classes, rather than word types
- What kind of classes?
    - Manually defined for application domain
    - Automatically learned via clustering
    - POS (doesn’t help)
- Generally mixed with word-based N-grams

Longer distance information

- Goodman 2006: With a corpus 284 million words, 5-grams improve on 4-grams, but above that longer n-grams don't help

- Skip N-grams: N-grams calculated over sequences of words that aren't necessarily contiguous.

  - Why might this help?

- Factored Language Models (Bilmes & Kirchhoff 2003): Represent words as bundles of stems + morphological features, then learn N-grams across that lattice

  - Why might this help?

Overview

- Review: simple n-grams
- Smoothing
- Interpolation
- Backoff
- Practical Issues: Toolkits & Data Formats
- Class-based N-grams
- LM adaptation
- Longer-distance information
- Review/lookback

Midterm: Wednesday 4/25

- In-class
- Open book, open notes, closed computers, closed internet
- Covers everything so far (except today)
- Write directly on the test paper (no bluebook required)

Midterm: Study guide

- Regular expressions & regular languages
- FSTs: graph notation
- FSTs: transition tables
- FSTs & regular relations
- FSTs & phonological rewrite rules
- Structure of evaluations in compling
- Precision & Recall
- TTS components
- N-grams (but not backoff, smoothing etc)

Synthesis

- What is computational linguistics?
- How does it differ from other subfields of CS/Linguistics?
- How is it similar to other subfields of CS/Linguistics?

Formal languages

- A formal language is a set of strings
- Things you can do with a formal language:
  - Recognize it
  - Parse it
  - Generate it

Formal languages

- Where have we encountered formal languages so far?

- What have we studied that does not involve formal languages?

Knowledge bases

- Knowledge bases are encodings of (linguistic) information
- What kinds have we seen so far in this class?
    - What are they used for?
    - What do they encode?

Reflection

- In what ways has the material covered so far matched your expectations for how compling/NLP works?

- In what ways has it been surprising?

- What presuppositions that you brought to the class have been challenged?

Overview

- Review: simple n-grams
- Smoothing
- Interpolation
- Backoff
- Practical Issues: Toolkits & Data Formats
- Class-based N-grams
- LM adaptation
- Longer-distance information
- Review/lookback