Artificial Intelligence Programming

Statistical NLP

Cindi Thompson

Department of Computer Science

University of San Francisco

Outline

- n-grams
  - Applications of n-grams
- review - Context-free grammars
- Probabilistic CFGs
- Information Extraction

Advantages of classical NLP

- Classical NLP approaches use a parser to generate a parse tree.
- This can then be used to transform knowledge into a form that can be reasoned with.
  - Identifies sentence structure
  - Easier to do semantic interpretation
  - Can handle anaphora, synonyms, etc.

Disadvantages of classical NLP

- Doesn't take frequency into account
- No way to choose between different parses for a sentence
- Can't deal with incorrect grammar
- Requires a lexicon.
- Maybe we can incorporate both statistical information and structure.

How do you represent a document?

- Many NLP techniques treat a document as a *bag of words*.
- Order is discarded; we just count how often each word appears.
- No semantics involved
- Intuition: Frequently-appearing words give an indication of subject matter.
- Advantage: No need to parse, computationally tractable for large collections.
- Disadvantage: Contextual information and meaning is lost.

Bag of Words model

- There might be some “cleanup” of the document - will cover that later
- Once a document has been cleaned, the simplest model just counts how many times each word occurs in the document.
- This is typically represented as a dictionary.
- You built this in assignment 1.

Bag-of-words = "BOW"

n-grams

- The simplest way to add structure to the BOW model is to count the occurrence not only of single tokens, but of *sequences* of tokens.
  - So far, we've considered words as tokens.
- A token is sometimes called a *gram*
- an n-gram model considers the probability that a sequence of n tokens occurs in a row.
  - More precisely, it is the probability $P(token_i|token_{i-1}, token_{i-2}, ..., token_{i-n})$

Building a corpus

Where do we get the conditional probabilities?

- To measure how frequently words occur in general, we must construct a corpus.
  - This is a large collection of documents
- Must be careful to ensure that we select documents of the appropriate style
- Different types of documents have different word frequencies
  - New York Times vs Livejournal
- The statistical distribution of words in a corpus is called a *language model*.

n-grams

- BOW uses 1-grams, or unigrams.
- We could also choose to count *bigrams*, or 2-grams.
- The sentence “Every good boy deserves fudge” contains the bigrams “every good”, “good boy”, “boy deserves”, “deserves fudge”
- We could continue this approach to 3-grams, or 4-grams, or 5-grams.
- Longer n-grams give us more accurate information about content, since they include phrases rather than single words.
- What’s the downside here?

Sampling theory

- We need to be able to estimate the probability of each n-gram occurring.
- We can do this by collecting a corpus and counting the distribution of words in the corpus.

Estimating bi-gram probabilities

- $P(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)}{count(w_{i-1})}$
- $P(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$

Example

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

P(I | <s>) = 2/3 = 0.67 P(Sam | <s>) = 1/3 = 0.33
P(am | I) = 2/3 = 0.67 P(</s> | Sam) = 1/2 = 0.5
P(Sam | am) = 1/2 = .5 P(do | I) = 1/3 =.33

Sampling Problems

- If the corpus is too small, these counts may not be reflective of an n-gram's true frequency.
- Many n-grams will not appear at all in our corpus.
- For example, if we have a lexicon of 20,000 words, there are:
  - $20,000^2$ = 400 million distinct bigrams
  - $20,000^3$ = 8 trillion distinct trigrams
  - $20,000^4 = 1.6 \times 10^{17}$ distinct 4-grams

Zero probabilities!

Input

- . . . denied the allegations
- . . . denied the reports
- . . . denied the claims
- . . . denied the request

P("offer" | "denied the") = 0

- But its not so unusual to see this:
  - . . . denied the offer
  - . . . denied the loan

Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V}$

Easy, but not very theoretically satisfying - a blunt instrument!

Linear Interpolation Smoothing

- We can also use estimates of shorter-length n-grams to help out.
  - Assumption: the sequence w_1, w_2, w_3 and the sequence w_1, w_2 are related.
- More precisely, we want to know $P(w_3|w_2, w_1)$. We count all 1-grams, 2-grams, and 3-grams.
- We estimate $P(w_3|w_2, w_1)$ as $\alpha_1 P(w_3|w_2, w_1) + \alpha_2 P(w_3|w_2) + \alpha_3 P(w_3)$
- So where do we get $\alpha_1, \alpha_2, \alpha_3$?
  - They might be fixed, based on past experience.
  - Or, we could learn them.

Application: language detection

- n-grams have also been successfully used to detect the language a document is in.
- Approach: consider *letters* as tokens, rather than words.
- Gather a corpus in a variety of different languages (Wikipedia works well here.)
- Process the documents, and count all two-grams.
- Estimate probabilities for Language L with $\frac{count}{\#of2-grams}$
  Call this P_L
- Assumption: different languages have characteristic two-grams.

Application: language detection

- To classify a document by language:
  - Find all two-grams in the document. Call this set T.
  - For each language L, the *likelihood* that the document is of language L is:
    $P_L(t_1) \times P_L(t_2) \times ... \times P_L(t_n)$
  - The language with the highest likelihood is the most probable language.
    - (this is a form of Bayesian inference - we'll spend more time on this later in the semester.)

Going further

- n-grams and segmentation provide some interesting ideas:
  - We can combine structure with statistical knowledge.
  - Probabilities can be used to help guide search
  - Probabilities can help a parser choose between different outcomes.
- But, no structure used apart from colocation.
- Maybe we can apply these ideas to grammars.

Reminder: CFGs

- Recall context-free grammars from the last lecture
- Single non-terminal on the left, anything on the right.
  - S -> NP VP
  - VP -> Verb | Verb PP
  - Verb -> 'run' | 'sleep'
- We can construct sentences that have more than one legal parse.
  - "Squad helps dog bite victim"
- CFGs don't give us any information about which parse to select.

Probabalistic CFGs

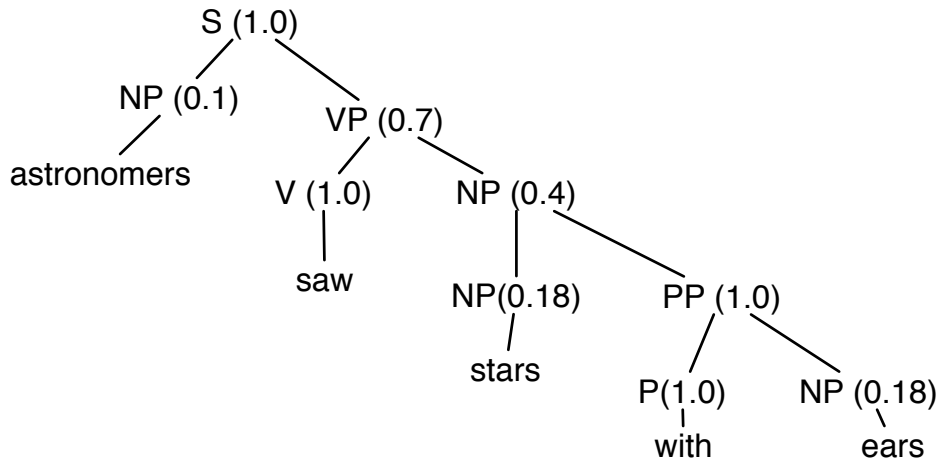
- A probabalisitc CFG is just a regular CFG with probabilities attached to the right-hand sides of rules.
  - The have to sum up to 1
- They indicate how often a particular non-terminal derives that right-hand side.

Example

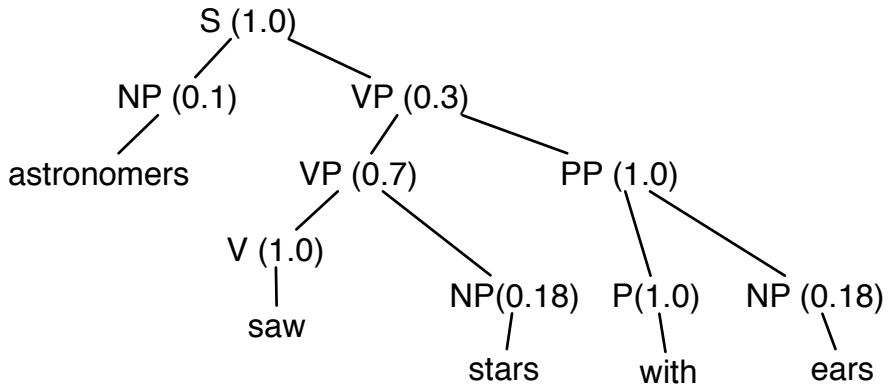
```
S -> NP VP (1.0)
PP -> P NP (1.0)
VP -> V NP (0.7)
VP -> VP PP (0.3)
P -> with (1.0)
V -> saw (1.0)
NP -> NP PP (0.4)
NP -> astronomers (0.1)
NP -> stars (0.18)
NP -> saw (0.04)
NP -> ears (0.18)
NP -> telescopes (0.1)
```

Disambiguation

- The probability of a parse tree being correct is just the product of each rule in the tree being derived.
- This lets us compare two parses and say which is more likely.



P1=1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0009072



P2=1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 1.0 * 0.18 * 1.0 * 0.18 = 0.00068

Faster Parsing

- We can also use probabilities to speed up parsing.
- Recall that both top-down and chart pasring proceed in a primarily depth-first fashion.
  - They choose a rule to apply, and based on its right-hand side, they choose another rule.
- Probabilities can be used to better select which rule to apply, or which branch of the search tree to follow.
- This is a form of best-first search.

Information Extraction

- An increasingly common application of parsing is *information extraction*.
- This is the process of creating structured information (database or knowledge base entries) from unstructured text.
- Example:
  - Suppose we want to build a price comparison agent that can visit sites on the web and find the best deals on flatscreen TVs?
  - Suppose we want to build a database about video games. We might do this by hand, or we could write a program that could parse wikipedia pages and insert knowledge such as madeBy(Blizzard, WorldOfWarcraft) into a knowledge base.

Extracting specific information

- A program that fetches HTML pages and extracts specfic information is called a *scraper*.
- Simple scrapers can be built with regular expressions.
  - For example, prices typically have a dollar sign, some digits, a period, and two digits.
  - $[0-9]+.[0-9]{2}
- This approach will work, but it has several limitations
  - Can only handle simple extractions
  - Brittle and page specific

Steps in information extraction

- A more robust system will need to take advantage of sentence structure.
- A typical system will have the following components:
  - Sentence segmenter.
  - Tokenizer.
  - Part of speech tagger.
  - Chunker.
  - Named Entity detector.
  - Relation extractor.

POS tagging

- There are a number of approaches to part-of-speech tagging.
  - We can write rules based on a word's structure. ("-ed" is a past tense verb)
  - We can learn rules based on labeled data.
    - Most common tag - ZeroR.
    - We can use contextual information - n-grams.
    - We can combine them, and learn more complex rules.

Chunking

- A chunk is a larger part of a sentence, such as a noun phrase.
- This will help us identify entities and relations.
- We can identify chunks with a chunk grammar:
  - $NP : < DT >? < JJ > * < NN >$
- Once we've tagged words with parts of speech, we use a parser to identify chunks.
- This can be done top-down or bottom up.

Named Entities

- These are noun phrases that refer to specific individuals, places, or organizations.
- How can we identify them, and what type of entity they are?
- e.g. University of San Francisco: NP - Organization, Barack Obama: NP - Person.
  - Maybe we have a *gazetteer*, but this is very brittle.
- We can also build a classifier to label entities.
  - Input: token with a part-of-speech label
  - Output: whether it is a Named Entity, and its type.

Relation extraction

- Once we have Named Entities, we would like to know relations between them.
  - In(USF, San Francisco)
- We can write a set of augmented regular expressions to do this.
  - <ORG>(.+)VP in(.+)<CITY> will match <organization> verb-phrase in blah-blah <city>.
- There will be false positives; getting this highly accurate takes some care.
- We can trade off precision and accuracy here - more restrictive regular expressions might miss some relations, but avoid adding false positives.

Summary

- We can combine the best of probabilistic and classical NLP approaches.
- n-grams take advantage of co-occurrence information.
  - language detection
- CFGs can be augmented with probabilities
- Speeds parsing, deals with ambiguity.
- Information extraction is an increasingly common application.