Statistical Natural Language Processing

July 18, 2006

CS 486/686

University of Waterloo

Outline

- Introduction to Statistical NLP
- Statistical Language Models
- Information Retrieval
  - Evaluation Metrics
- Other Applications of Statistical NLP

- Reading: R&N Sect. 23.1, 23.2

Symbolic NLP Insufficient

- Symbolic NLP generally fails because…
  - Grammars too complex to specify
  - NL is vague, imprecise, and ambiguous
  - NL is often context dependent

Motivation behind Statistical NLP

- Symbolic NLP involves:
  - Constructing a set of "rules" (eg. a grammar) for the language and the NLP task.
  - Applying the rules to the data.
- Success depends on how well the rules describe the data.
- How to ensure the rules fit the data well? Derive the rules from the data – statistical natural language processing.

Statistical NLP

- Statistical NLP involves:
  - Analyzing some (training) data to derive patterns and rules for the language and the NLP task.
  - Applying the rules to the (test) data.
- Symbolic NLP specifies how a language should be used, while statistical NLP specifies how a language is usually used.
- Often both are needed – hybrid models.

Statistical Language Models

- One of the most fundamental tasks in statistical NLP.
- A statistical / probabilistic language model defines a probability distribution over a (possibly infinite) set of strings.
- We'll look at two popular examples:
  - N-gram models: distribution over words
  - Probabilistic context free grammar

Unigram model

- Unigram: independent distribution $P(w)$ for each word w in the lexicon
- Given a document D,
  - $P(w) = \#w \text{ in } D / \Sigma_i \#w_i \text{ in } D$
  - Word sequence: $\Pi_i P(w_i)$
- Ex. 20-word sequence generated at random from a unigram model of the textbook:
  - logical are as are confusion a may right tries agent goal the was diesel more object then information-gathering search is

Bigram model

- Bigram: conditional distribution $P(w_i|w_{i-1})$ for each word w_i given the previous word w_{i-1}
- Given a document D,
  - $P(w_i|w_{i-1}) = \#(w_i,w_{i-1})$ in D / $\#w_{i-1}$ in D
  - Word sequence: $P(w_0) \Pi_i P(w_i|w_{i-1})$
- Ex. word sequence generated at random from a bigram model of the textbook:
  - planning purely diagnostic expert systems are very similar computational approach would be represented compactly using tic tac toe a predicate

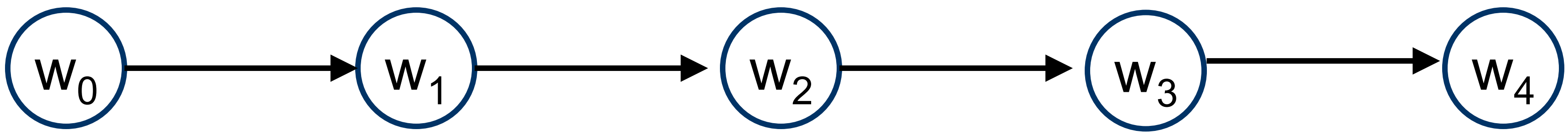
Trigram model

- Trigram: conditional distribution $P(w_i|w_{i-1},w_{i-2})$ for each word w_i given the previous two words
- Given a document D,
  - $P(w_i|w_{i-1},w_{i-2}) = \#(w_i,w_{i-1},w_{i-2})$ in D / $\#(w_{i-1},w_{i-2})$ in D
  - Word sequence: $P(w_0)\, P(w_1|w_0)\, \Pi_i\, P(w_i|w_{i-1},w_{i-2})$
- Ex. word sequence generated at random from a trigram model of the textbook:
  - planning and scheduling are integrated the success of naïve bayes model is just a possible prior source by that time

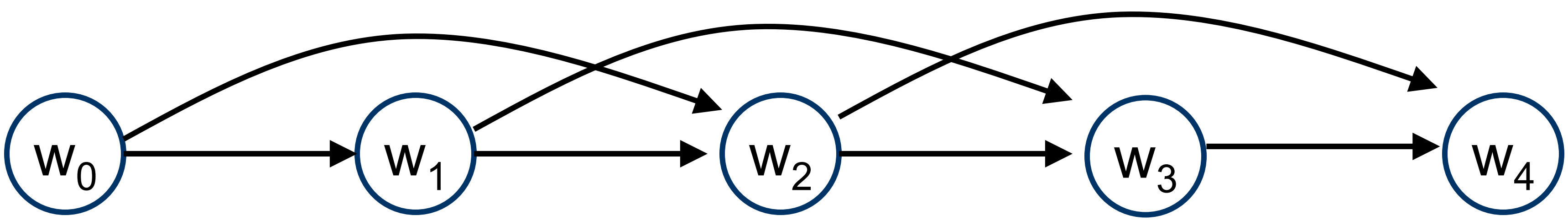
Graphically

- Unigram: zeroth-order Markov process

- Bigram: first-order Markov process



- Trigram: second-order Markov process



N-gram models

- N-gram models:
  - Quality: language model improves with n
  - Learning: amount of data necessary increases exponentially with n

- Suppose corpus of k unique words and K total words:
  - Unigram model: $K > k$
  - Bigram model: $K > k^2$
  - Trigram model: $K > k^3$

Textbook

- Textbook has:
  - 15,000 unique words
  - 500,000 total words

- Model complexity:
  - Unigram model: 15,000 probabilities
  - Bigram model: $15{,}000^2$ = 225 million probabilities
    - 99.8% of probabilities are zero!
  - Trigram model: $15{,}000^3$ = 3.375 trillion probs
    - 99.9999% of probabilities are zero!

Smoothing

- Zero probabilities can be problematic:
  - Word sequence: $\Pi_i P(w_i|w_{i-1},w_{i-2},\ldots) = 0$ as soon as $\exists_i$ such that $P(w_i|w_{i-1},w_{i-2},\ldots) = 0$

- Solutions:
  - Add-one smoothing
    $\hat{P}(w_i|w_{i-1}) = [\#(w_i,w_{i-1})+1] / [\#w_{i-1}+k^2]$
  - Linear interpolation smoothing
    $\hat{P}(w_i|w_{i-1}) = c_2 P(w_i|w_{i-1}) + c_1 P(w_i)$
    where $c_1 + c_2 = 1$

Probabilistic Context-Free Grammar (PCFG)

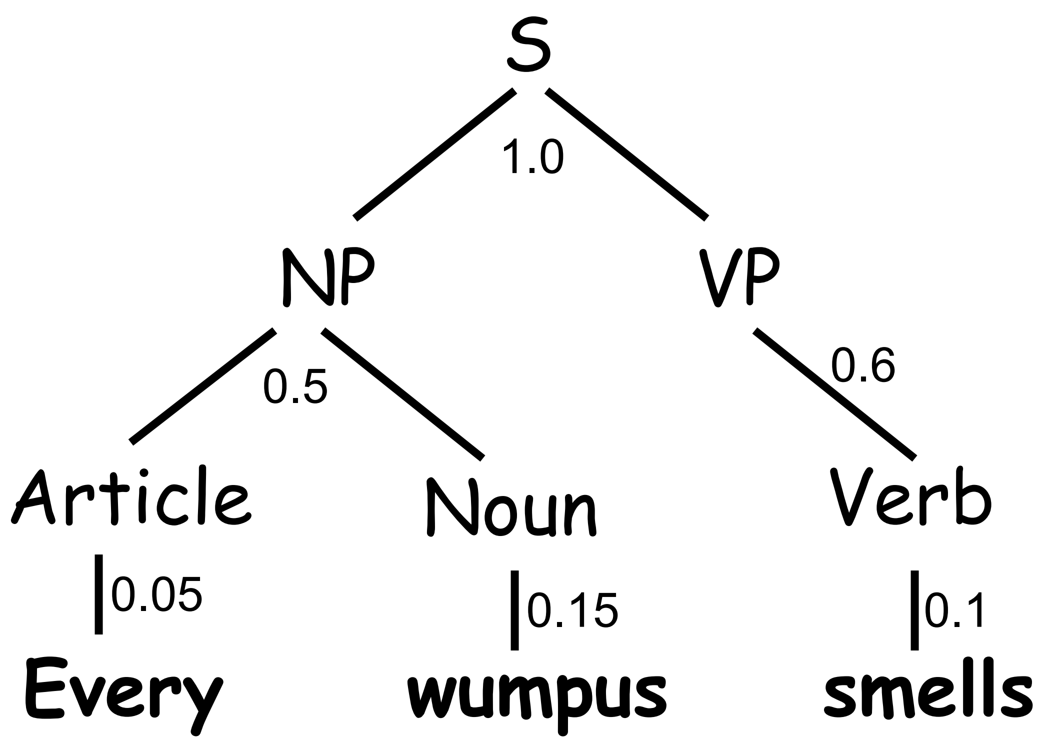
- N-gram models:
  - Basic probabilistic language models
- Context-free grammars:
  - Sophisticated symbolic language models

- Probabilistic context free grammars:
  - Sophisticated probabilistic language models
  - Assign probabilities to rewrite rules

Example PCFG

- S → NP VP [1.00]
- NP → Pronoun [0.10]
  | Name [0.10]
  | Noun [0.20]
  | Article Noun [0.50]
  | NP PP [0.10]
- VP → Verb [0.60]
  | VP NP [0.20]
  | VP PP [0.20]
- Noun → breeze[0.10] | wumpus[0.15] | agent[0.05] | …
- Verb → sees [0.15] | smells [0.10] | goes [0.25] | …
- Article → the [0.30] | a [0.35] | every [0.05] | …

Example probabilistic parse tree



Parse tree prob:
1.0*0.5*0.6*0.05*0.15*0.1 = 0.000225

Learning PCFGs

- When corpus of parsed sentences available:
  - Learn probability of each rewrite rule $P(lhs \rightarrow rhs) = \#(lhs \rightarrow rhs) / \#(lhs)$
- Problems:
  - But we need a CFG… which is hard to design
  - We also need to parse by hand lots of sentences… which takes a long time

Learning PCFGs

- Lots of texts are available, but not parsed… can we learn from those?
- Yes: use EM algorithm
  - E step: given rule probabilities, compute expected frequency of each rule in some corpus.
  - M step: given expected frequency of each rule, update the rule probabilities by normalizing the rule frequencies.
- Problems:
  - EM gets stuck in local optima
  - Probabilistic parses often unintuitive to linguists

Learning PCFGs

- Could we also learn without a grammar?
- Yes: for instance assume grammar is in Chomsky normal form (CNF)
  - Any CFG can be represented in CNF
  - Only two types of rule:
    - $X \rightarrow Y\ Z$
    - $X \rightarrow t$
  - But effective only for small grammars

Information Retrieval

- Information retrieval: task of finding documents that are relevant to a user
- Information retrieval components:
  - Document collection
  - Query posed
  - Resulting set of relevant documents
- Examples:
  - www search engines
  - Text classification and clustering

Information Retrieval

- Initial attempts:
  - Parse documents into knowledge base of logical formulas
  - Parse query into a logical formula
  - Answer query by logical inference
- It failed because of …
  - Ambiguity
  - Unknown context
  - Etc…

Information Retrieval

- Alternative:
  - Build unigram model for each document D_i
  - Treat query Q as a bag of words
  - Find document D_i that maximizes $P(Q|D_i)$
- It works!

Example

- Query: {Bayes, information, retrieval, model}
- Documents: each chapter of the textbook
- Build unigram model for each chapter

- Computation:
  - $P(Q|D_i) =$ P(Bayes, information, retrieval, model | chapter i)
  - $P(Q|D_i')$: same as P(Q|Di) but with add-one smoothing

Example

Words	Query	Chapt 1 Intro	Chapt 13 Uncert.	Chapt 15 Time	Chapt 22 NLP	Chapt 23 Current
Bayes	1	5	32	38	0	7
information	1	15	18	8	12	39
retrieval	1	1	1	0	0	17
model	1	9	7	160	9	63
N	4	14,680	10,941	18,186	16,397	12,574
$P(Q\|D_i)$		1.5×10^{-14}	2.8×10^{-13}	0	0	1.2×10^{-11}
$P(Q\|D_i')$		4.1×10^{-14}	7.0×10^{-13}	5.2×10^{-13}	1.7×10^{-15}	1.5×10^{-11}

Evaluation

- Two measures:
  - Precision measures the proportion of documents that are actually relevant
    - false positive rate = 1 - precision
  - Recall measures the proportion of all relevant documents in the result set
    - false negative rate = 1 - recall

Evaluation

	In result set	Not in result set
Relevant	30	20
Not relevant	10	40

- Precision: 30/(30+10) = 0.75
  - False positive rate = 1 – precision = 0.25
- Recall: 30/(30+20) = 0.6
  - False negative rate = 1 – recall = 0.4

Tradeoff

- There is often a tradeoff between recall and precision
- Perfect recall:
  - Return every document
  - But precision will be poor
- Perfect precision:
  - Return only documents for which we are certain about their relevancy, or none at all
  - But recall will be poor

F1 Score

- F1 score (or F measure) combines precision and recall
- Definition: $2pr / (p+r)$
  - If $p = r \neq 0$, $f = p = r$
  - If $p = 0$ or $r = 0$, $f = 0$
  - Otherwise favours compromising

Precision	1	0.9	0.5	0.7
Recall	1	0.2	0.6	0.8
F Measure	1	0.33	0.55	0.75

IR Refinement

- Refinements:
  - Case folding: convert to lower case
    - E.g. COUCH → couch, Italy → italy
  - Stemming: truncate words to their stem
    - E.g. couches → couch, taken → take
  - Synonyms:
    - E.g. sofa → couch
- Improves recall, but worsens precision

Statistical NLP Applications

- Many other NLP tasks are shifting toward statistical / hybrid approaches.
  - Segmentation
  - Part-of-speech tagging
  - Parsing
  - Text classification / clustering
  - Text summarization
  - Machine translation
  - Textual entailment
  - Semantic role labelling

Next Class

- Next Class:
  - Robotics
  - Russell and Norvig Ch. 25