

Statistical Language Processing

July 19, 2005

CS 486/686

University of Waterloo

Outline

- Statistical Language Processing
 - Probabilistic language models
 - Information retrieval
- Reading: R&N Sect. 23.1, 23.2

Symbolic Language Processing

- Generally fails because...
 - Grammars too difficult to specify
 - Natural language ambiguous/context dependent
 - Logical approaches unsuitable for ambiguous/imprecise speech
- But something changed in the mid 90's...

Statistical Language Processing

- World Wide Web:
 - Huge corpus (large collection of text)
 - Need to retrieve information quickly
- Statistical Language Processing:
 - Train probabilistic language models with abundant www data
 - Rough language models sufficient for basic information retrieval such as text classification and clustering

Probabilistic Language Models

- Defines a probability distribution over a (possibly infinite) set of strings.
- Examples:
 - N-gram models: distribution over sequences of n words
 - Probabilistic context free grammars

Unigram model

- **Unigram:** independent distribution $P(w)$ for each word w in the lexicon
- Given a document D ,
 - $P(w) = \#w \text{ in } D / \sum_i \#w_i \text{ in } D$
 - Word sequence: $\prod_i P(w_i)$
- 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 ,
 - $P(w_i | w_{i-1}) = \#(w_i, w_{i-1}) \text{ in } D / \#w_{i-1} \text{ in } D$
 - Word sequence: $P(w_0) \prod_i P(w_i | w_{i-1})$
- 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 ,
 - $P(w_i | w_{i-1}, w_{i-2}) = \#(w_i, w_{i-1}, w_{i-2}) \text{ in } D / \#(w_{i-1}, w_{i-2}) \text{ in } D$
 - Word sequence: $P(w_0) P(w_1 | w_0) \prod_i P(w_i | w_{i-1}, w_{i-2})$
- 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

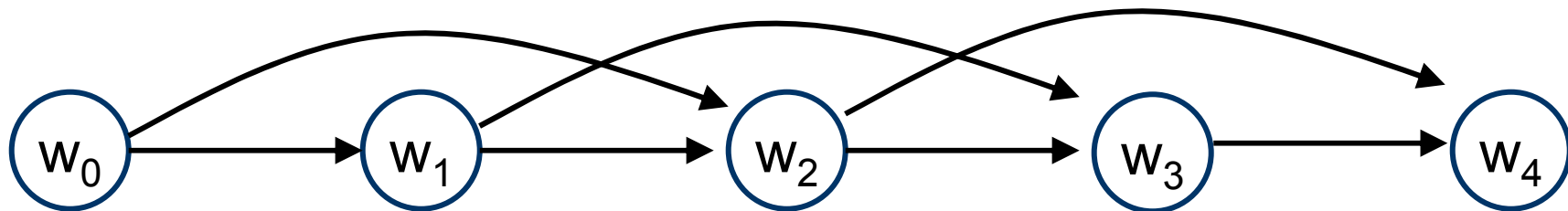
- Unigram: zeroth-order Markov process



- 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: $\prod_i P(w_i | w_{i-1}, w_{i-2}, \dots) = 0$ as soon as \exists_i such that $P(w_i | w_{i-1}, w_{i-2}, \dots) = 0$
- 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 $c_1 + c_2 = 1$

Segmentation

- Segmentation: find word boundaries in a text with no spaces
 - Necessary in Japanese and Chinese
 - Application of N-gram models
 - Ex: I tiseasytoreadwordswithoutspaces
- Hidden Markov Model:
 - Hidden states are words
 - Observations are characters
 - Compute most-likely word sequence:
 $\max_{\text{words}} P(\text{words}|\text{characters})$

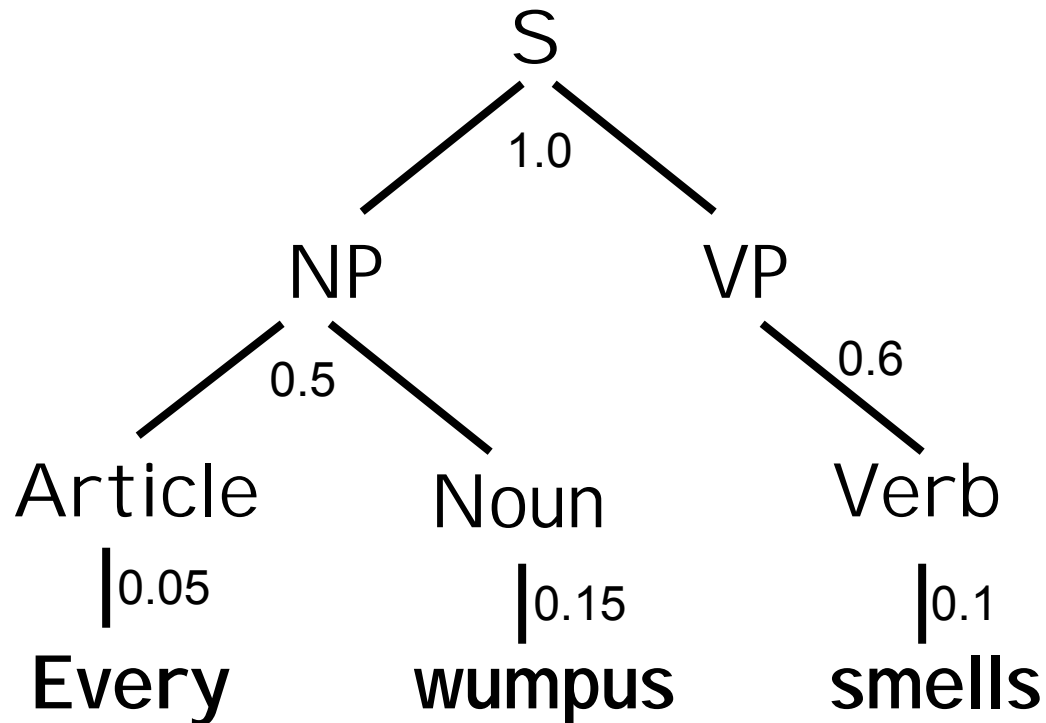
Probabilistic Context-Free Grammar (PCFG)

- 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

- $S \rightarrow NP VP$ [1.00]
- $NP \rightarrow$ Pronoun [0.10]
 - | Name [0.10]
 - | Noun [0.20]
 - | Article Noun [0.50]
 - | NP PP [0.10]
- $VP \rightarrow$ Verb [0.60]
 - | VP NP [0.20]
 - | VP PP [0.20]
- Noun \rightarrow breeze[0.10] | wumpus[0.15] | agent[0.05] | ...
- Verb \rightarrow sees [0.15] | smells [0.10] | goes [0.25] | ...
- Article \rightarrow 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
 $P(\text{lhs} \rightarrow \text{rhs}) = \#(\text{lhs} \rightarrow \text{rhs}) / \#(\text{lhs})$
- 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:
 - $X \rightarrow YZ$
 - $X \rightarrow t$
 - 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:
 - $P(Q|D_i) = P(\text{Bayes, information, retrieval, model} \mid \text{chapter } i)$
 - $P(Q|D_i')$: same as $P(Q|D_i)$ 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 = $1 - \text{precision}$
 - **Recall** measures the proportion of all relevant documents in the result set
 - false negative rate = $1 - \text{recall}$

Evaluation

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

- Precision: $30/(30+10) = 0.75$
 - False positive rate = $1 - \text{precision} = 0.25$
- Recall: $30/(30+20) = 0.6$
 - False negative rate = $1 - \text{recall} = 0.4$

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
 - But recall will be poor

Information Retrieval Refinements

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

Other IR tasks

- Text classification/clustering
- Text summarization
- Machine translation
- Etc.
- Shift towards statistical approaches!

Next Class

- Next Class:
 - Computational Vision
 - Russell and Norvig Ch. 24