Context-Free Grammars and CKY Algorithm

CS114 Lab 10

Kenneth Lai

March 27, 2020

Review: Regular Languages

- A language is regular iff:

Review: Regular Languages

- A language is regular iff:
  - It is recognized by a finite state automaton

Review: Regular Languages

- A language is regular iff:
  - It is recognized by a finite state automaton
  - It is generated by a regular expression

Natural languages are not regular

- Why not?

Context-Free Languages

- More powerful than regular languages

Context-Free Languages

- More powerful than regular languages
- Used in specification of natural and programming languages

Context-Free Languages

- A language is context-free iff:

Context-Free Languages

- A language is context-free iff:
  - It is recognized by a pushdown automaton

Context-Free Languages

- A language is context-free iff:
  - It is recognized by a pushdown automaton
  - It is generated by a context-free grammar

Phrase structure grammars = context-free grammars

- $G = (T, N, S, R)$
  - T is set of terminals
  - N is set of nonterminals
    - For NLP, we usually distinguish out a set $P \subset N$ of preterminals, which always rewrite as terminals
    - S is the start symbol (one of the nonterminals)
    - R is rules/productions of the form $X \rightarrow \gamma$, where X is a nonterminal and γ is a sequence of terminals and nonterminals (possibly an empty sequence)
- A grammar G generates a language L.

Phrase structure grammars = context-free grammars

- $S \rightarrow aSb$
- $S \rightarrow \epsilon$

Phrase structure grammars = context-free grammars

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

Source

Probabilistic CFGs

- Augment each rule in R with a conditional probability

Probabilistic CFGs

- Augment each rule in R with a conditional probability
  - $P(\text{LHS} \rightarrow \text{RHS}) = P(\text{RHS}|\text{LHS})$

Probabilistic CFGs

- Augment each rule in R with a conditional probability
  - $P(\text{LHS} \rightarrow \text{RHS}) = P(\text{RHS}|\text{LHS})$
- The probability of a parse T is the product of the probabilities of all of the n rules used to generate T

Probabilistic CFGs

- Augment each rule in R with a conditional probability
  - $P(\text{LHS} \rightarrow \text{RHS}) = P(\text{RHS}|\text{LHS})$
- The probability of a parse T is the product of the probabilities of all of the n rules used to generate T
  - $P(T) = \prod_{i=1}^{n} P(\text{RHS}_i|\text{LHS}_i)$

Probabilistic CFGs

Grammar		Lexicon
S → *NP VP*	[.80]	*Det* → *that* [.10] | *a* [.30] | *the* [.60]
S → *Aux NP VP*	[.15]	*Noun* → *book* [.10] | *flight* [.30]
S → *VP*	[.05]	| *meal* [.05] | *money* [.05]
NP → *Pronoun*	[.35]	| *flight* [.40] | *dinner* [.10]
NP → *Proper-Noun*	[.30]	*Verb* → *book* [.30] | *include* [.30]
NP → *Det Nominal*	[.20]	| *prefer* [.40]
NP → *Nominal*	[.15]	*Pronoun* → *I* [.40] | *she* [.05]
Nominal → *Noun*	[.75]	| *me* [.15] | *you* [.40]
Nominal → *Nominal Noun*	[.20]	*Proper-Noun* → *Houston* [.60]
Nominal → *Nominal PP*	[.05]	| *NWA* [.40]
VP → *Verb*	[.35]	*Aux* → *does* [.60] | *can* [.40]
VP → *Verb NP*	[.20]	*Preposition* → *from* [.30] | *to* [.30]
VP → *Verb NP PP*	[.10]	| *on* [.20] | *near* [.15]
VP → *Verb PP*	[.15]	| *through* [.05]
VP → *Verb NP NP*	[.05]	
VP → *VP PP*	[.15]	
PP → *Preposition NP*	[1.0]	

Figure 14.1 A PCFG that is a probabilistic augmentation of the $\mathscr{L}_1$ miniature English CFG grammar and lexicon of Fig. **??**. These probabilities were made up for pedagogical purposes and are not based on a corpus (since any real corpus would have many more rules, so the true probabilities of each rule would be much smaller).

Source

Chomsky Normal Form

- Every rule is of the form

Chomsky Normal Form

- Every rule is of the form
  - $S \to \epsilon$
  - $A \to BC$
  - $A \to a$

Chomsky Normal Form

- Every rule is of the form
  - $S \to \epsilon$
  - $A \to BC$
  - $A \to a$
- Where S is the start symbol, A is a nonterminal, B and C are nonterminals (except for S), and a is a terminal

Converting a CFG into Chomsky Normal Form

- Add a new start symbol

Converting a CFG into Chomsky Normal Form

- Add a new start symbol
- Remove ϵ-rules

Converting a CFG into Chomsky Normal Form

- Add a new start symbol
- Remove ϵ-rules
- Remove unary rules

Converting a CFG into Chomsky Normal Form

- Add a new start symbol
- Remove ϵ-rules
- Remove unary rules
- Break up rules with more than 3 things on the right hand side

Converting a CFG into Chomsky Normal Form

- Add a new start symbol
- Remove ϵ-rules
- Remove unary rules
- Break up rules with more than 3 things on the right hand side
- Replace terminals with nonterminals and add new rules as needed

Converting a CFG into Chomsky Normal Form

- Add a new start symbol
- Remove ϵ-rules
- Remove unary rules
- Break up rules with more than 3 things on the right hand side
- Replace terminals with nonterminals and add new rules as needed

  - We can modify CKY algorithm to handle unary rules

Parsing Context-Free Languages

- Bottom-up

Parsing Context-Free Languages

- Bottom-up
  - CKY algorithm

Parsing Context-Free Languages

- Bottom-up
  - CKY algorithm
- Top-down

Parsing Context-Free Languages

- Bottom-up
  - CKY algorithm
- Top-down
  - Earley algorithm

CKY Algorithm

- Given tables *table* and *back*:

CKY Algorithm

- Given tables *table* and *back*:
  - Base case

CKY Algorithm

- Given tables *table* and *back*:
  - Base case
    - Fill in *table* cells $[i, i+1]$ with all possible nonterminals that can generate that word

CKY Algorithm

- Given tables *table* and *back*:
  - Base case
    - Fill in *table* cells $[i, i+1]$ with all possible nonterminals that can generate that word
  - Recursive case

CKY Algorithm

- Given tables *table* and *back*:
  - Base case
    - Fill in *table* cells $[i, i+1]$ with all possible nonterminals that can generate that word
  - Recursive case
    - In *table*, if $A \rightarrow BC$ and B is in cell $[i, j]$ and C is in cell $[j, k]$, fill in cell $[i, k]$ with A

CKY Algorithm

- Given tables *table* and *back*:
  - Base case
    - Fill in *table* cells $[i, i+1]$ with all possible nonterminals that can generate that word
  - Recursive case
    - In *table*, if $A \rightarrow BC$ and B is in cell $[i, j]$ and C is in cell $[j, k]$, fill in cell $[i, k]$ with A
    - In *back*, fill in cell $[i, k]$ with backpointers (e.g. A: j, B, C)

Probabilistic CKY Algorithm

- Given tables *table* and *back*:
  - Base case
    - Fill in *table* cells $[i, i+1]$ with all possible nonterminals that can generate that word, and their probabilities
  - Recursive case
    - In *table*, if $A \rightarrow BC$ and B is in cell $[i, j]$ and C is in cell $[j, k]$, and $table[i, j, A] < P(A \rightarrow BC) \times table[i, j, B] \times table[j, k, C]$, fill in cell $[i, k]$ with A:
      $P(A \rightarrow BC) \times table[i, j, B] \times table[j, k, C]$
    - In *back*, fill in cell $[i, k]$ with backpointers (e.g. A: j, B, C)

Probabilistic CKY Algorithm

table:

book	that	flight

back:

book	that	flight

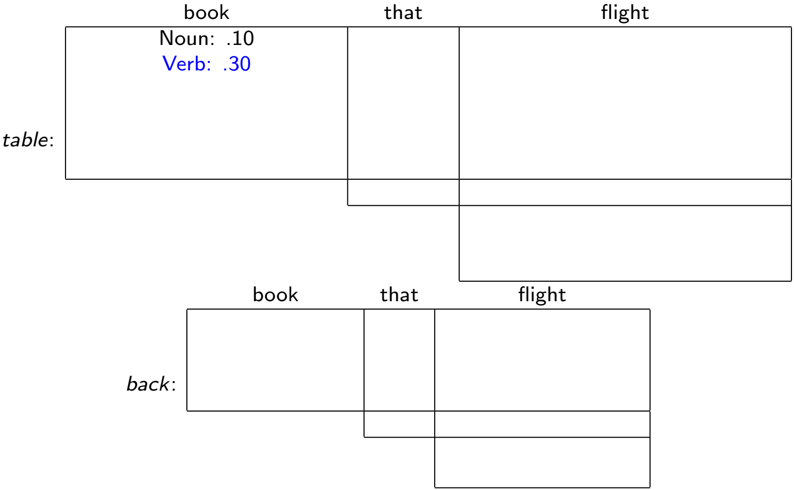
Probabilistic CKY Algorithm

	book	that	flight
table:	Noun: .10		

	book	that	flight
back:			

- Noun $\rightarrow$ book [.10]

Probabilistic CKY Algorithm



- Verb → book [.30]

Probabilistic CKY Algorithm

table:

	book	that	flight
	Noun: .10 Verb: .30		
		Det: .10	

back:

	book	that	flight

- Det $\rightarrow$ that [.10]

Probabilistic CKY Algorithm

table:

book	that	flight
Noun: .10 Verb: .30		
	Det: .10	
		Noun: .30

back:

book	that	flight

- Noun → flight [.30]

Probabilistic CKY Algorithm

- Unary rules

Probabilistic CKY Algorithm

- Unary rules
  - In *table*, if $A \rightarrow B$ and B is in cell $[i, i+1]$, fill in cell $[i, i+1]$ with A: $P(A \rightarrow B) \times \mathit{table}[i, i+1, B]$

Probabilistic CKY Algorithm

- Unary rules
  - In *table*, if $A \rightarrow B$ and B is in cell $[i, i+1]$, fill in cell $[i, i+1]$ with A: $P(A \rightarrow B) \times table[i, i+1, B]$
  - In *back*, fill in cell $[i, i+1]$ with backpointers (e.g. A: B)

Probabilistic CKY Algorithm

table:	book	that	flight
	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$		
		Det: .10	
			Noun: .30

back:	book	that	flight
	Nominal: Noun		

- Nominal $\rightarrow$ Noun [.75]

Probabilistic CKY Algorithm

table:

	book	that	flight
	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$		
		Det: .10	
			Noun: .30

back:

	book	that	flight
	Nominal: Noun NP: Nominal		

- NP $\rightarrow$ Nominal [.15]

Probabilistic CKY Algorithm

table:

	book	that	flight
	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$		
		Det: .10	
			Noun: .30

back:

	book	that	flight
	Nominal: Noun NP: Nominal VP: Verb		

- VP $\rightarrow$ Verb [.35]

Probabilistic CKY Algorithm

	book	that	flight
table:	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		
		Det: .10	
			Noun: .30

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		

- S $\rightarrow$ VP [.05]

Probabilistic CKY Algorithm

table:

book	that	flight
Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		
	Det: .10	
		Noun: .30 Nominal: $.75 \times .30 = .225$

back:

book	that	flight
Nominal: Noun NP: Nominal VP: Verb S: VP		
		Nominal: Noun

- Nominal $\rightarrow$ Noun [.75]

Probabilistic CKY Algorithm

table:

	book	that	flight
	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		
		Det: .10	
			Noun: .30 Nominal: $.75 \times .30 = .225$ NP: $.15 \times .225 = .03375$

back:

	book	that	flight
	Nominal: Noun NP: Nominal VP: Verb S: VP		
			Nominal: Noun NP: Nominal

- NP $\rightarrow$ Nominal [.15]

Probabilistic CKY Algorithm

table:

	book	that	flight
	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		
		Det: .10	NP: $.20 \times .10 \times .225 = .0045$
			Noun: .30 Nominal: $.75 \times .30 = .225$ NP: $.15 \times .225 = .03375$

back:

	book	that	flight
	Nominal: Noun NP: Nominal VP: Verb S: VP		
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal

- NP $\rightarrow$ Det Nominal [.20]

Probabilistic CKY Algorithm

	book	that	flight
table:	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		VP: $.20 \times .30 \times .0045 = .00027$
		Det: .10	NP: $.20 \times .10 \times .225 = .0045$
			Noun: .30 Nominal: $.75 \times .30 = .225$ NP: $.15 \times .225 = .03375$

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal

- VP $\rightarrow$ Verb NP [.20]

Probabilistic CKY Algorithm

table:

	book	that	flight
table:	Noun: .10 Verb: .30 Nominal: $.75 \times .10 = .075$ NP: $.15 \times .075 = .01125$ VP: $.35 \times .30 = .105$ S: $.05 \times .105 = .00525$		VP: $.20 \times .30 \times .0045 = .00027$ S: $.05 \times .00027 = .0000135$
		Det: .10	NP: $.20 \times .10 \times .225 = .0045$
			Noun: .30 Nominal: $.75 \times .30 = .225$ NP: $.15 \times .225 = .03375$

back:

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal

- S $\rightarrow$ VP [.05]

Probabilistic CKY Algorithm

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal

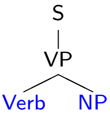
Probabilistic CKY Algorithm

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal

S
|
VP

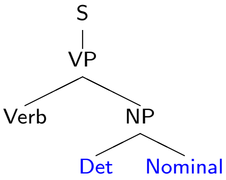
Probabilistic CKY Algorithm

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal



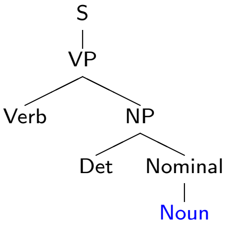
Probabilistic CKY Algorithm

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal



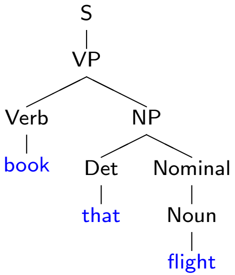
Probabilistic CKY Algorithm

	book	that	flight
back:	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal



Probabilistic CKY Algorithm

back:	book	that	flight
	Nominal: Noun NP: Nominal VP: Verb S: VP		VP: 1 Verb NP S: VP
			NP: 2 Det Nominal
			Nominal: Noun NP: Nominal



Earley Algorithm

- Fill in a chart with dotted rules

Earley Algorithm

- Fill in a chart with dotted rules
  - Predictor
    - If the nonterminal after the dot is not a preterminal (POS tag), add a state for each possible expansion of the nonterminal

Earley Algorithm

- Fill in a chart with dotted rules
  - Predictor
    - If the nonterminal after the dot is not a preterminal (POS tag), add a state for each possible expansion of the nonterminal
  - Scanner
    - If the nonterminal after the dot is a preterminal (POS tag), read the next word in the input and, if the word can be tagged as that tag, add a state to the next position in the chart

Earley Algorithm

- Fill in a chart with dotted rules
  - Predictor
    - If the nonterminal after the dot is not a preterminal (POS tag), add a state for each possible expansion of the nonterminal
  - Scanner
    - If the nonterminal after the dot is a preterminal (POS tag), read the next word in the input and, if the word can be tagged as that tag, add a state to the next position in the chart
  - Completer
    - If the dot is at the end of the rule, add states corresponding to previous states that were “looking” for the rule to be completed, and advancing the dot

Pushdown Automata

- Finite state automaton with stack

Pushdown Automata

- Finite state automaton with stack
  - Infinite memory that can only be used last in, first out

Pushdown Automata

- Finite state automaton with stack
  - Infinite memory that can only be used last in, first out
- Equivalent to CFGs

Pushdown Automata

- Finite state automaton with stack
    - Infinite memory that can only be used last in, first out
- Equivalent to CFGs
- For more information, see theory of computation book

Natural languages are not context-free

- Why not?