Module -3
Syntax Analysis

Introduction

- Syntax Analysis-It is the study of formal relationships between words.
- Part-of-Speech: When words are grouped into similar classes which can be called as Part of speech,word classes,morphological classes or lexical tags,these classes gives information about a word and its neighbours.
- Greek/Traditional Grammar has 8 POS(noun,verb,pronoun,preposition,adverb,conjunction,adjective,article)
- Penn Treebank-45 POS tags
- Brown Corpus-87 POS tags
- C7 tagset-146 POS tags

- Penn Tree PoS Tagging

https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

- Using Penn Treebank tags, tag the following sentence from the Brown Corpus:

The grand jury commented on a number of other topics.

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

How many word classes are there?

- A basic set:
  - N, V, Adj, Adv, Prep, Det, Aux, Part, Conj, Num
- A simple division: open/content vs. closed/function
  - Open: Noun, Verb, Adjective, Adverb
  - Closed: Preposition(on,under), Determiner(the), Auxilary/modal(can,may,should), Conjunction(and,but,or) , Numerals(one,two)
- Many subclasses, e.g.
  - eats/V $\Rightarrow$ eat/VB, eat/VBP, eats/VBZ, ate/VBD, eaten/VBN, eating/VBG, ...
  - Reflect morphological form & syntactic function

- Types of POS Methods

1.Rule Based POS Tagging

2.Stochastic POS Tagging

3.Transformation Based tagging

Rule Based POS Tagging

- They use dictionary or lexicon for getting possible tags for tagging each word.
- If the word has more than one possible tags,then it use hand written rules to identify the correct tag.
- Disambiguation can also be performed in rule based tagging by analyzing the linguistic features of a word along with its preceding as well as following words.
- For example: suppose if the preceding word of word is a determiner then that word will be a noun.
- The can was rusted.

- All such kind of information in rule based POS tagging is coded in the form of rules.
- It Uses ENGTWOL tagger which is based on a two-stage architecture.
- The first stage used a dictionary to assign each word a list of potential part of speech.
- The second stage used large lists of hand written disambiguation rules to select the part of speech for each word.
- The ENGTWOL lexicon is based on the two-level morphology, and has about 56000 entries for English word stems,counting a word with multiple part of speech.
- Each entry is annotated with a set of morphological and syntactic features.

Sample ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

2. Stochastic Part–of–Speech Tagging

- The model that includes frequency or probability is called stochastic.
- Stochastic taggers generally resolve tagging ambiguities by using a training corpus to compute the probability of a given word having a given tag in a given context.
- It is also called HMM tagger or a Maximum Likelihood Tagger or a Markov Model HMM tagger.
- Approached:

1.Word frequency approach:

Stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag.

2.Tag sequence probability:

Here the tagger calculates the probability of a given sequence of tags occurring.

3. Transformation Based tagging

A hybrid approach

- Like rule-based taggers, this tagging is based on rules
- Like (most) stochastic taggers, rules are also automatically induced from hand-tagged data

Basic Idea: do a quick and dirty job first, and then use learned rules to patch things up

Overcomes the pure rule-based approach problems of being too expensive, too slow, too tedious etc…

A Simple Example

- From the Brown Corpus
- Secretariat/NNP is/VBZ expected/VBN to/TO *race*/VB tomorrow/NN
- People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT *race*/NN for/IN outer/JJ space/NN

Assume previous words have been tagged, and we want to tag the word *race*.

Bigram tagger

- to/TO *race*/?
- the/DT *race*/?

Examples

- Race
  - “race” as NN: .98
  - “race” as VB: .02
- So you’ll be wrong 2% of the time, which really isn’t bad
- Patch the cases where you know it has to be a verb
  - Change NN to VB when previous tag is TO

Advantages:

1.Use of learned rules

2.Less complex

3.Faster than stochastic tagging

Disadvantage:

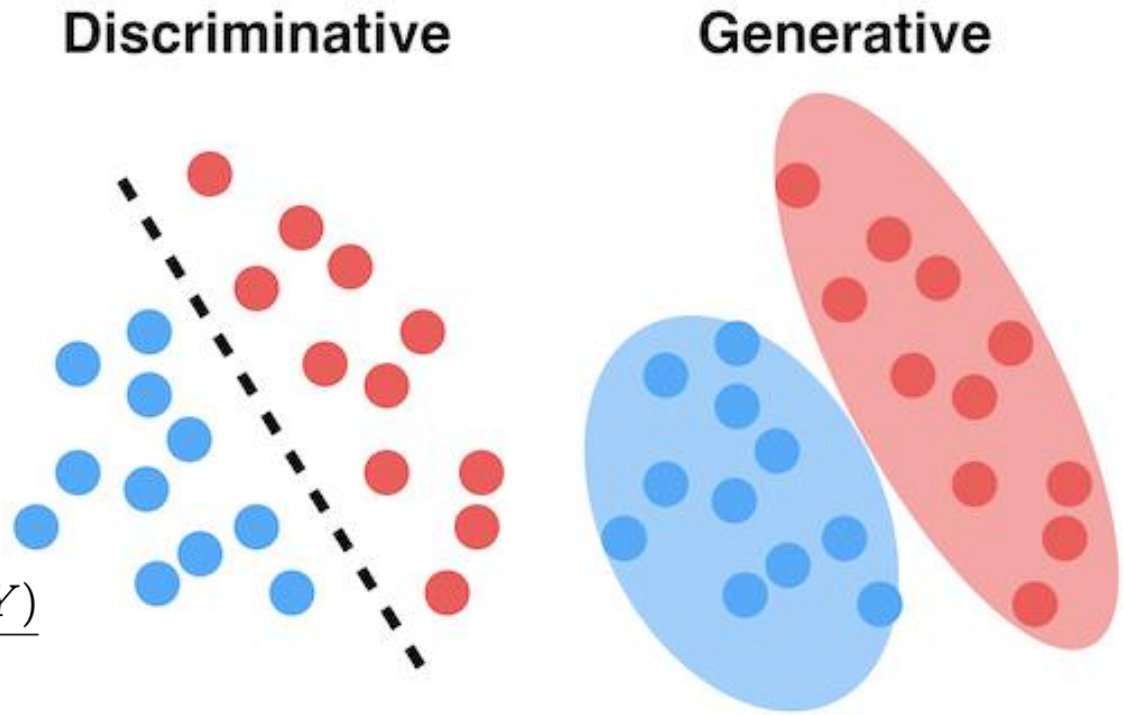
Training time takes more time if corpus is big.

Discriminative and Generative Model

- **Generative models**, to find the conditional probability P(Y|X), they estimate the prior probability P(Y) and likelihood probability P(X|Y) with the help of the training data and uses the Bayes Theorem to calculate the posterior probability P(Y |X):

$$posterior = \frac{prior \times likelihood}{evidence} \Rightarrow P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

- **Discriminative model** refers to a class of models used in Statistical Classification, mainly used for supervised machine learning. These types of models are also known as conditional models since they learn the boundaries between classes or labels in a dataset.



Hidden Markov Model(Generative Model)

- A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process — call it X — with unobservable ("hidden") states.
- As part of the definition, HMM requires that there be an observable process Y whose outcomes are "influenced" by the outcomes of X in a known way.
- Since X cannot be observed directly, the goal is to learn about X by observing Y.
- HMM (Hidden Markov Model) is a Stochastic technique for POS tagging. Hidden Markov models are known for their applications to reinforcement learning and temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following, partial discharges, and bioinformatics.

Use of Hidden Markov Model for POS tagging

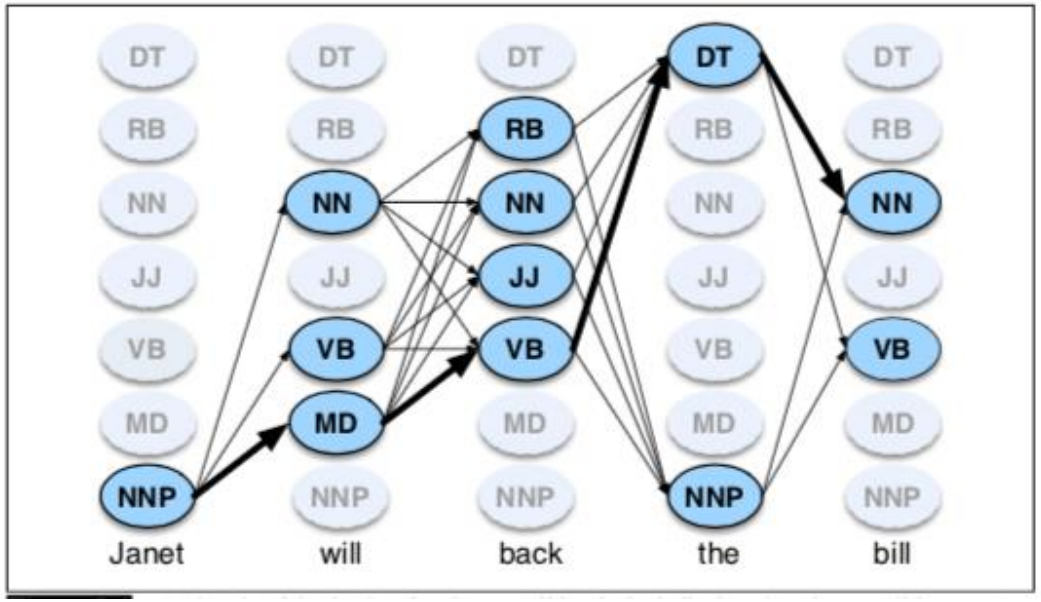
- The simplest stochastic approach finds out the most frequently used tag for a specific word in the annotated training data and uses this information to tag that word in the unannotated text.
- But sometimes this approach comes up with sequences of tags for sentences that are not acceptable according to the grammar rules of a language.
- One such approach is to calculate the probabilities of various tag sequences that are possible for a sentence and assign the POS tags from the sequence with the highest probability.
- Hidden Markov Models (HMMs) are probabilistic approaches to assign a POS Tag.

Note: Practice numerical based on HMM POS tagging

Viterbi Algorithm

- The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).
- **Example:'Janet will back the bill'**
- **Best path/selection:**
- **V_t(j) = max : V_t-1 * a(i,j) * b_j(O_t)**

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

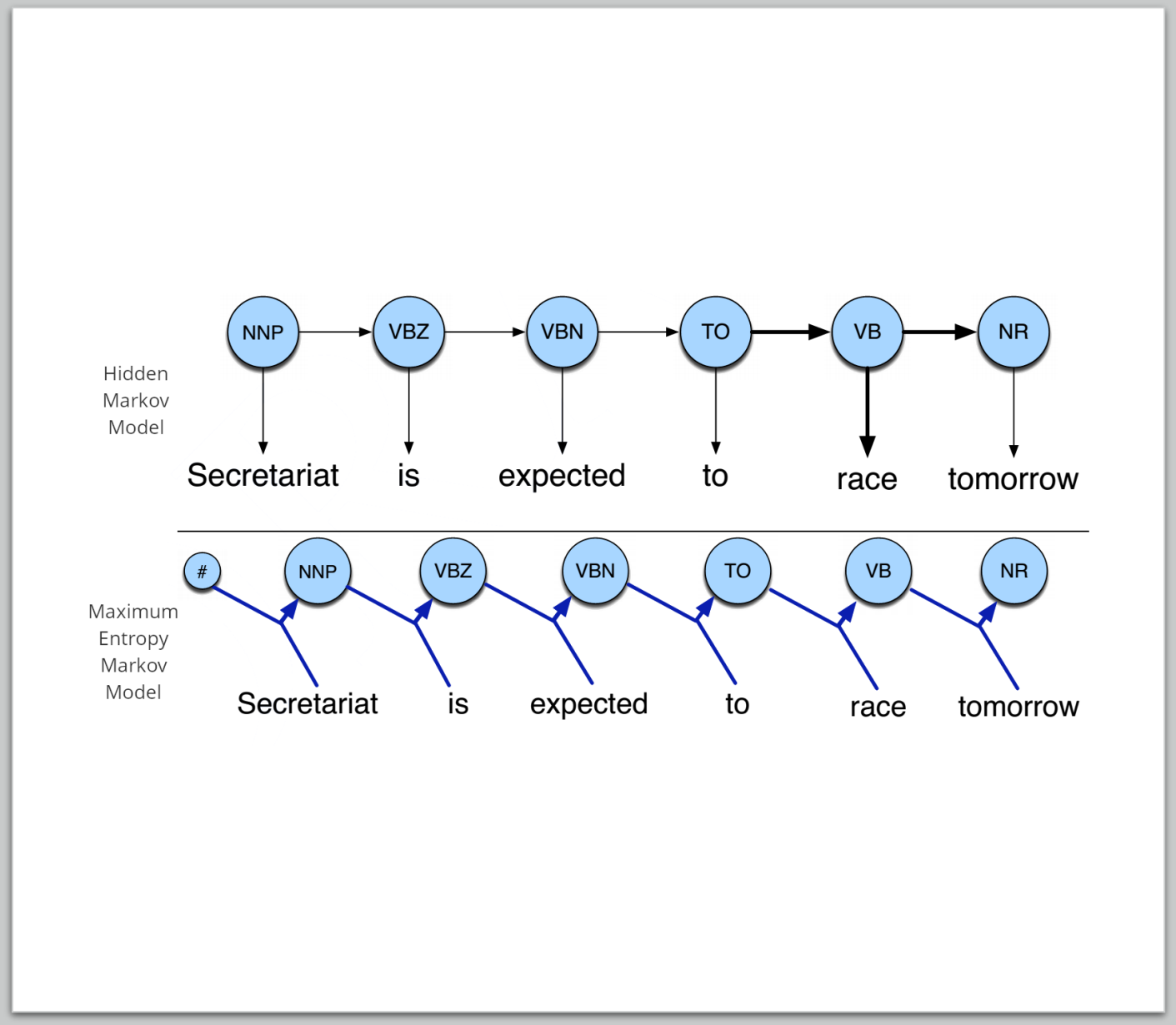


- a is the Transition matrix and b is the emission matrix obtained by applying HMM.

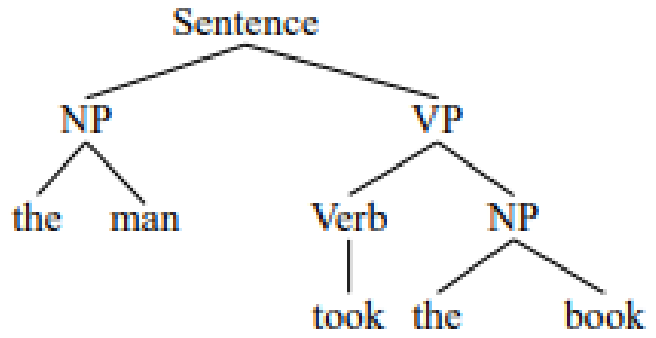
Janet/NNP will/MD back/VB the/DT bill/NN.

Maximum-Entropy Markov Model

- MEMM is a model that makes use of state-time dependencies. It uses predictions of the past and the current observation to make current prediction.
- HMM is a generative model because words as modelled as observations generated from hidden states. Even the formulation of probabilities uses likelihood P(W|T) and prior P(T). On the other hand, MEMM is a discriminative model. This is because it directly uses posterior probability P(T|W); that is, probability of a tag sequence given a word sequence. Thus, it discriminates among the possible tag sequences



Context Free Grammar



- Example: The man took the book.
- A context-free grammar consists of a set of rules or productions, each

RULES of which expresses the ways that symbols of the language can be grouped and ordered together, and a lexicon of words and symbols. For example, LEXICON the following productions expresses that a NP (or noun phrase), can be NP composed of either a ProperNoun or of a determiner (Det) followed by a Nominal; a Nominal can be one or more Nouns.

- The symbols that are used in a CFG are divided into two classes. The symbols that correspond to words in the language (‘the’, ‘nightclub’) are called terminal symbols; the lexicon is the set of rules that introduce these TERMINAL terminal symbols. The symbols that express clusters or generalizations of these are called nonterminals. In each context-free rule, the item to the right NONTERMINAL of the arrow is an ordered list of one or more terminals and nonterminals, while to the left of the arrow is a single nonterminal symbol expressing some cluster or generalization

$$NP \rightarrow Det\ Nominal$$
$$NP \rightarrow ProperNoun$$
$$Nominal \rightarrow Noun \mid Noun\ Nominal$$

$$Det \rightarrow a$$
$$Det \rightarrow the$$
$$Noun \rightarrow flight$$

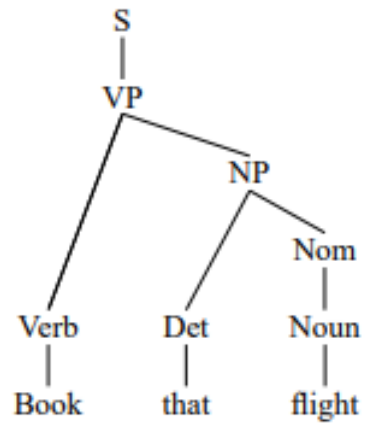


Figure 10.1 The correct parse tree for the sentence *Book that flight* according to the grammar in Figure 10.2.

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	
$Nominal \rightarrow Noun\ Nominal$	$Prep \rightarrow from \mid to \mid on$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	$Nominal \rightarrow Nominal\ PP$

Figure 10.2 A miniature English grammar and lexicon.

Parsing Technique

- Top Down Parsing: It is a parsing technique which starts from the top of the parse tree, move downwards, evaluates rules of grammar(E.g. Predictive parser, Early Parser)
- Bottom Up Parsing : It is a parsing technique which starts from the lowest level of the parse tree, move upwards and evaluates rules of grammar(E.g. PCFG,CYK ,Shift reduce parser)

Sr. No.	Key	Top Down Parsing	Bottom Up Parsing
1	Strategy	Top down approach starts evaluating the parse tree from the top and move downwards for parsing other nodes.	Bottom up approach starts evaluating the parse tree from the lowest level of the tree and move upwards for parsing the node.
2	Attempt	Top down parsing attempts to find the left most derivation for a given string.	Bottom up parsing attempts to reduce the input string to first symbol of the grammer.
3	Derivation Type	Top down parsing uses leftmost derivation.	Bottom up parsing uses the rightmost derivation.
4	Objective	Top down parsing searches for a production rule to be used to construct a string.	Bottom up parsing searches for a production rule to be used to reduce a string to get a starting symbol of grammer.

Predictive Parser

- Predictive parser is a recursive descent parser, which has the capability to predict which production is to be used to replace the input string. The predictive parser does not suffer from backtracking.
- To accomplish its tasks, the predictive parser uses a look-ahead pointer, which points to the next input symbols. To make the parser back-tracking free, the predictive parser puts some constraints on the grammar and accepts only a class of grammar known as LL(k) grammar.
- Predictive parsing uses a stack and a parsing table to parse the input and generate a parse tree. Both the stack and the input contains an end symbol $ to denote that the stack is empty and the input is consumed. The parser refers to the parsing table to take any decision on the input and stack element combination.

Note:Practice numericals based on this topics

- Algorithm

Repeat
Begin
Let X be the top of stack symbol & a be the next input symbol;
If X is a terminal or $ then
Pop X from stack and remove a from input
Else ERROR();
Else if X is non-terminal
If M[X,a]=X->Y1|Y2…|Yk
Then begin
Pop X from the stack
Push Yk,Y(k-1)…Y1 onto the stack
Y1 on top
End
Else
ERROR();
End
Until X=$/*stack is empty*/

Example

1. E->TE’
   E’->+TE’|ϵ
   T->(E)|id

2. E->TE’
   E’->+TE’| ϵ
   T->FT’
   T’->*FT’| ϵ
   F-> (E)| id

Early Parser

The Earley Parsing Algorithm: an efficient top-down parsing algorithm that avoids some of the inefficiency associated with purely naive search with the same top-down strategy (cf. recursive descent parser).

- Intermediate solutions are created only once and stored in a chart (dynamic programming).
- Left-recursion problem is solved by examining the input.
- Earley is not picky about what type of grammar it accepts, i.e., it accepts arbitrary CFGs (cf. CKY).

Earley Parsing Algorithm (J&M, p. 444)

```
function EARLEY-PARSE(words,grammar) returns chart

    ENQUEUE((γ → •S, [0,0]),chart[0])
    for i ← from 0 to LENGTH(words) do
        for each state in chart[i] do
            if INCOMPLETE?(state) and NEXT-CAT(state) is not POS then
                PREDICTOR(state)
            elseif INCOMPLETE?(state) and NEXT-CAT(state) is POS then
                SCANNER(state)
            else
                COMPLETER(state)
        end
    end
    return(chart)
```

PCFG(Probabilistic Context Free Grammar)

- PCFGs extend context-free grammars similar to how hidden Markov models extend regular grammars.
- Each production is assigned a probability. The probability of a derivation (parse) is the product of the probabilities of the productions used in that derivation.
- These probabilities can be viewed as parameters of the model, and for large problems, it is convenient to learn these parameters via machine learning. A probabilistic grammar's validity is constrained by context of its training dataset.
- PCFGs have application in areas as diverse as natural language processing to the study the structure of RNA molecules and design of programming languages.
- Designing efficient PCFGs has to weigh factors of scalability and generality. Issues such as grammar ambiguity must be resolved. The grammar design affects results accuracy.
- Grammar parsing algorithms have various time and memory requirements.

NOTE:Practice numerical based on this topic.

PCFG: $G = (T, N, S, R, P)$

- T: set of terminals
- N: set of non-terminals
  - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R: Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)*$
- $P(R)$ gives the probability of each rule.

$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

Features of PCFG

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- In practice, a PCFG is a worse language model for English than an n-gram model
- All else being equal, the probability of a smaller tree is greater than a larger tree

CYK Parser

- CYK means Cocke-Younger-Kasami.
- It is one of the earliest recognition and parsing algorithms. The standard version of CYK can only recognize languages defined by context-free grammars in Chomsky Normal Form (CNF).
- It is also possible to extend the CYK algorithm to handle some grammars which are not in CNF.
- Based on a “dynamic programming” approach –

➢Build solutions compositionally from sub-solutions

➢It uses the grammar directly.

NOTE:Practice numerical based on this topic.

Algorithm

- Let n be the number of words in the input. Think about $n+1$ lines separating them, numbered 0 to n.
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j.
- We build the Table bottom-up.