Natural Language Processing

Chapter 4 Syntax and Parsing

Dr Raymond Lee
Associate Professor, DST
BNU-HKBU United International College

Syntax and Parsing

- Introduction and Motivation
- What is Syntax?
- Types of Constituents in Sentences
- Syntax Analysis
- Context-Free Grammars (CFG)
- Derivations using CFGs
- CFG Parsing
- Lexical and Probabilistic Parsing
- Probabilistic CFGs
- The Lexicon and Lexical Parsing
- Summary

Part 1
Syntax Analysis

Introduction and motivation
Basic Issues in NLP

- Linguistic Aspect
  - What are the facts about language?
  - Any rules and patterns ?
  - Derive word classification scheme – Word Classes (POS)
  - Derive rule of creating sentences (utterances) – Grammars and Grammatic rules -> Syntax (Syntactic rules)
- Algorithmic Aspect
  - What are effective computational procedures for dealing with those facts?
  - Stochastic-based, Rule-based and Machine Learning-based

Motivation:

- Any method and algorithm to build a manual and even automatic scheme to do Syntactic Level analysis of natural languages
  - Building parsers

Fig 4.1 shows the relation between grammars, syntax and the corresponding Parse Tree for the simple sentence: Tom pushed the car.

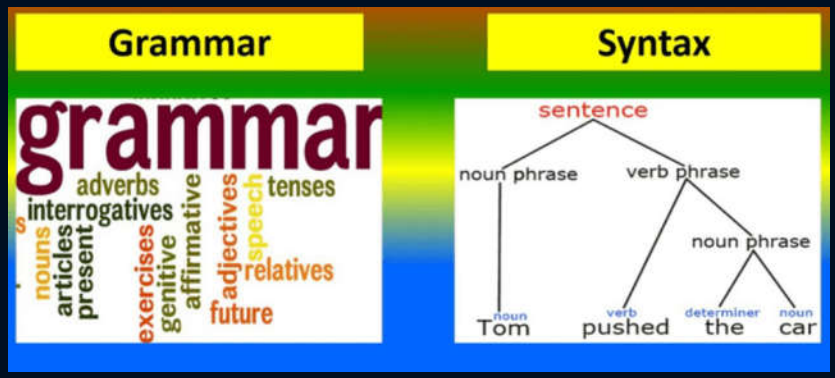

Fig. 4.1 Grammar, Syntax and Parse Tree

What is Syntax?

What is Syntax?

- In linguistics, "syntax" refers to the rules that govern the ways in which words combine to form phrases, clauses, and sentences.
- The term "syntax" comes from the Greek word “σύνταξη”, meaning "arrange together."
- The term is also used to mean the study of the syntactic properties of a language.
- Two meanings:
  - Syntax is the proper order of words in a phrase or sentence.
  - Syntax is a tool used in writing proper grammatical sentences.
- Native speakers of a language learn correct syntax without realizing it, it is the “nature” part of our language system, which constitute the language itself.
- The complexity of a writer's or speaker's sentences creates a formal or informal level of phrase and clause that is presented to its audience.
- In NLP, the term refers to the proper ordering of symbols and tokens so that the computer can understand what instructions are telling it to do – not the meaning.
- In fact, POS is the basic components (entities) for us to construct the syntactic rules.

What is Syntax?

Syntactic Rules

- For English Language, the POS often follow ordering patterns in sentences and clauses.
- Such as compound sentences are joined by conjunctions (by using: and, or, with) or that multiple adjectives modifying the same noun follow a particular order according to their class (E.g. "The big black dog").
- The syntactic rules of how to order words help the language parts make sense.
- In English language, sentences (utterances) are often start with a subject, followed by a predicate (or just a verb in the simplest sentences) and contain an object or a complement (or both), which shows, for example, what's being acted upon.
- For example: "John chased the dog" – a typical sentence with a "subject-verb-object" pattern – basic syntactic rule in English.
- However in "John quickly chased the dog in wild green field." – adverbs and adjectives try to take their places in front of the what they're modifying ("quickly chased", "wild green field") to make the description more lively and informative.

What is Syntax?

7 Commonly Used Syntactic Patterns

1. Subject → Verb

   [4.1] The cat meowed.

   This is the standard syntactic pattern, including the minimum requirements of just a subject and verb. The subject always comes first.

2. Subject → Verb → Direct Object

   [4.2] The cat plays the ball.

   If the verb is transitive and uses a direct object, the direct object always goes after the verb.

3. Subject → Verb → Subject Complement

   [4.3] The cat is playful.

   The subject complement comes after the verb. Subject complements always use linking verbs, like be or seem.

4. Subject → Verb → Adverbial Complement

   [4.4] The cat paced slowly

   Like subject complements, adverbial complements come after the verb (if there are no objects).

What is Syntax?

7 Commonly Used Syntactic Patterns

5. Subject → Verb → Indirect Object → Direct Object

   [4.5] The cat gave me the ball.

   Some sentences have both a direct object and an indirect object. In this case, the indirect object comes right after the verb, and the direct object comes after the indirect object. For example, you can say, The cat gave the ball to me.

6. Subject → Verb → Direct Object → Object Complement

   [4.6] The dog made the ball dirty.

   Object complements come after the direct object, similar to other complements.

7. Subject → Verb → Direct Object → Adverbial Complement

   [4.7] The dog perked its ears up.

   When the sentence uses both a direct object and an adverbial complement, the direct object comes first, followed by the adverbial complement. In this syntax example, up is the adverbial complement because it describes how the cat perked its ears.

The main purpose of Syntactic Parsing is study how to create and generalize such rules using POS Tags and to perform automatic (semi-automatic) sentence parsing.

Why Do We Care about Syntax & Parsing?

Importance of Syntax & Parsing in NLP

- Recall Fig 4.2 The major components in NLU, Syntax (and Parsing) plays a central roles in the whole NLP link up the natural language with its syntactic structure before we try to "understand" its semantic meaning.
- It can be considered as the "first-tier" checking or analysis of whether the sentences (utterance) "make sense" or not.
- In other words, if a sentence and dialog has syntactic error e.g. "John buys" (buy what?) It simply doesn't make sense.
- Its sole process is also useful in many NLP applications such as
  - Grammatic Checking – word processing applications such as Ms Word.
  - Speech Recognizer – real-time syntactic level of human speech especially in noisy environment.
- Play vital role other high-level NLP application such as Machine Translation and Q&A chatbot systems.

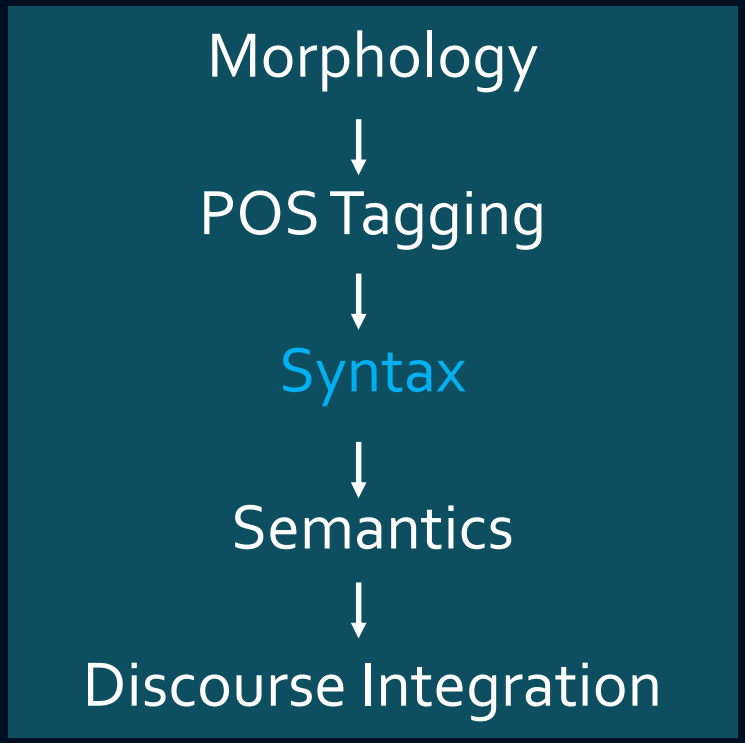

Fig. 4.2 Major Components in NLU

Constituents in Sentences

What is Constituent?

- In English grammar, a constituent is a linguistic part of a larger sentence.
- For instance, all the words and phrases that make up a sentence are said to be constituents of that sentence.
- A constituent can be a morpheme, word, phrase, or clause.
- Sentence analysis identifies the subject or predicate or different parts of speech, a process known as parsing the sentence into its constituents.

For example:

How to describe this cat?

[4.8] The milky cat with long tail is meowing ….

•What can be chopped out and replaced by a single word?

[4.9] Coco is meowing.

[4.10] Coco with long tail is meowing.

Or more complex constituent:

[4.11] The milky cat with long tail is meowing in the late afternoon.

[4.12] The milky cat with long tail is mewing in the late afternoon while Jack is sleeping.

Fig. 4.3 The milky cat with long tail meowing

Finding Constituents in Sentences

Instead of unit of object in a Noun-Phrase (NP), constituents can also be "Unit of Time"?

Example: Constituents as Unit-of-Time with different variations of usage, some syntactically okay but some with syntactic errors.

[4.13] Jane wants to go to Greece late this winter.

[4.14] Late this winter Jane wants to go to Greece.

[4.15] Jane wants late this winter to go to Greece.

[4.16] Late Jane wants to go Greece this winter. [syntax errors]

[4.17] Jane wants late to go to Greece this winter. [syntax errors]

[4.18] The late this winter Jane wants to go to Paris. [syntax errors, why?]

How Many Kinds of Constituents are There?

How Many Kinds of Constituents in English language?

- Every sentence (and every phrase and clause) has constituents.
- In other words, every sentence is made up of parts of other things that work together to make the sentence meaningful.
- Although there may be an infinite number of possible constituent tokens and their combination inside a sentence, there's quite a small number of constituent types.
- Commonly used constituent types include: NP (Noun-Phrase), VP (Verb-Phrase) and PP (Preposition-Phrase).
- For example, in the sentence:
  [4.19] My cat Coco scratch the UPS courier on the ankle.
  - This constituent parts are the subject, made up of a Noun Phrase ("my cat Coco"), and the predicate, the Verb Phrase ("scratch the UPS courier on the ankle.")

A Noun Phrase (NP)

- Made up of a noun and its modifiers.
- Modifiers that come before the noun include articles, possessive nouns, possessive pronouns, adjectives, or participles.
- Modifiers that come after include prepositional phrases, adjective clauses, and participle phrases.
- In [4.19]:
  - "My cat Coco" is the NP which consists of: DT (Determiner, My) + NN (Noun, cat) + NNP (Proper noun, Coco)
  - In which DT + NN are group together as PP (Possessive pronoun/noun) such as "My cat".
- In fact, NPs occur as subjects, objects of verbs, and objects of prepositions.

Example:

[4.20] The milky cat with long tail is meowing.
[4.21] Very few cat wore a collar.
[4.22] The long tail is brought to room.
[4.23] Many places hear meowing.
[4.24] A cat with a long tail and a collar meowing.
[4.25] Jane saw so many cats in the room.

How Many Kinds of Constituents are There?

A Verb Phrase (VP)

- A verb phrase is a group of words, including the main verb and any other linking verbs or modifiers, that act as a sentence's verb.
- Modifiers are words that can change, adapt, limit, expand upon, or help define a certain word in a sentence.
- In the case of verb phrases, the modifiers are usually auxiliary verbs (helping verbs), such as is, has, am, and are, that work alongside (or help) the main verb.
- In verb phrases, the main verb holds information about the event or activity that is being referred to, and the auxiliary verbs add meaning by relating to the time or aspect of the phrase.

Commonly Types of VP

1. VP with only the main verb – VP as a singular main verb on its own.
   [4.26] Jack catch a deer.
2. Auxiliary verb (to be) + main verb (-ing form) - When the main verb is being used in its -ing form (e.g. walking, talking), it expresses a continuous aspect. The use of auxiliary verbs will show whether the continuing action is in the past, present, or future.
   [4.27] Jack is singing.
3. Auxiliary verb (have) + main verb (past participle form) - VP includes the verb 'to have' (including all of its forms e.g. have, has, had) and the past participle form of the main verb.
   [4.28] Jack has broken the vase.
4. Modal verb + main verb - Modal verbs are a type of auxiliary verb that express modality. Modality includes things such as possibility, probability, ability, permission, ability, and obligation. Example modal verbs include: must, shall, will, should, would, can, could, may, and might.
   [4.29] Jack will leave.
5. Auxiliary verb (have + been) + main verb (-ing form) – VP with both the continuous aspect and the perfect aspect are expressed. The continuous aspect comes from the '-ing' verb, and the perfect aspect comes from the auxiliary verb 'have been'.
   [4.30] Jack has been washing the floor.
6. Auxiliary verb (to be) + main verb (past participle form) – VP with the verb 'to be' and a past participle form of the main verb expresses a passive voice. The passive voice is used to show that an action is happening to the subject of the sentence rather than the subject performing the action.
   [4.31] The lunch was served.
7. Negative and interrogative verb phrases – For sentences that have a negative or interrogative nature (i.e. they express a negative or ask a question), the verb phrase gets separated.
   [4.32] Jack is not answering the exam questions.
8. Emphasised verb phrases – VP with the using of auxiliary verbs 'do, does, did' to add emphasis to a sentence.
   [4.33] Jack did enjoy the vacation.
9. Composite VP – VP with consist of other VP or NP.

Example: [4.19] My cat Coco scratch the UPS courier on the ankle.

- "scratch" is the main verb in the VP that describe the "main action or event" happens to the "UPS courier" (the object).
- "on the ankle" is just the "auxiliary information" to further explain the event. With or without it, the sentence still make sense.
- In term of VP, this VP includes the verb ("scratch"), the NP "the UPS courier," and the prepositional phrase "on the ankle."

Single Word Constituents

Single word constituents are exactly the parts of speech that we have already considered in Chapter 3.

How many of these single word constituent types are there? Look at sizes of tagsets.

Some other most complex design decisions:

[4.34] Jane bought the big red handbag.

[4.35] Jane bought the red big handbag. [x]

Are *big* and *red* the same POS?

Simple Constituent Types Don’t Capture Everything

[4.36] *The cat with a long tail meowing a collar. [x]*

[4.37] *Jane imagined a cat with a long tail.*

[4.38] *Jane decided to go.*

[4.39] *Jane decided a cat with a long tail. [x]*

[4.40] *Jane decided a cat with a long tail would be her next pet.*

[4.41] *Jane gave lily the food.*

[4.42] *Jane decided lily the food. [x]*

Verb Phrase Subcategorization

What is Verb Phrase Subcategorization?

- One might wonder, is there any "universal pattern" or "structure" to classify the verbs in VPs? The answer is Yes.
- In linguistics, subcategorization denotes the ability or necessity for lexical items (usually verbs) to allow the presence and types of the syntactic arguments with which they co-occur.
- Although traditional English grammars subcategorize verbs into these two subcategories (transitive and intransitive), modern English grammars distinguish over 100 different subcategories.
- In fact, the relation between the verb and these other constituents can be considered the verb as a predicate and the constituents as arguments of the predicate.
- Subcategorization frames are sets of rules used to generate syntactic structures out of the base form.
- Fig 4.4 shows some examples of VP with different frames of subcategorization.
- As shown in Fig. 4.4, the verb "talk", "sleep" can "standalone" as the only verb in a VP:
  [4.43] He talks. (VP -> VB)
  [4.44] I laugh. (VP -> VB)
- While the verb "find" or "see" might need to "integrate" with one (or more than one) NP:
  [4.45] He find a clue. (VP -> VB + NP)
  [4.46] She sees Jack. (VP -> VB + NP)
- Some verbs like "show" needs 2 NPs (one as specifier and other as complement) to form a proper sense:
  [4.47] Please show me the map. (VP -> VB + NP + NP)
- Some verbs like "insist" needs 2 NPs (one as specifier and other as prepositional phrase) to form a proper sense:
  [4.48] Jack insists to land here. (VP -> VB + PP + NP)
- Some verbs such as "mean" or "think" even needs to integrate with some complex clause and sentence to more a complete sentence (utterance):
  [4.49] Do you mean that I need to attend the exam? (VP -> VB + S) in which "I need to attend the exam" itself can be a standalone sentence (clause) with proper sense.

Frame Rule	Description	Examples
ϕ	VP with single verb as member.	talk, sleep, eat, laugh, etc.
VS(NP);	The verbal phrase requires a noun phrase (NP) as a specifier (VS) - intransitive verbs	find, see, leave, get, etc.
VS(NP)VC(NP);	The verbal phrase requires a noun phrase (NP) as a specifier (VS) and a noun phrase (NP) as a complement (VC) (direct transitive verbs) - direct transitive verbs	show, make, read, write, etc
VS(NP)VC(PH([on]));	The verbal phrase requires a noun phrase (NP) as a specifier (VS) and a prepositional phrase (PP) headed by "on" as a complement (VC) - indirect transitive verbs governing "on"	depend, insist, operate, suggest etc.
VS(NP)VC(NP)VC(PH([to]));	The verbal phrase requires a noun phrase as a specifier (VS), a noun phrase as a complement (VC), and a prepositional phrase headed by "to" as a complement (VC) - ditransitive verbs	give, mean, think, etc.

Fig. 4.4 Examples of Verbs with different frames of subcategorization in VP syntax

The Role of the Lexicon in Parsing

What is Lexicon?

- A lexicon is the vocabulary of a language or branch of knowledge (such as medical and computer science).
- In linguistics, a lexicon is a language's inventory of lexemes.
- The word lexicon derives from Greek word λεξικόν (lexikon) meaning 'of or for words'
- Linguists believe the all human languages as consisting of two parts: a lexicon, essentially a catalogue of a language's words; and a grammar, a system of rules which allow for the combination of those words into meaningful sentences.
- Items within a lexicon are called lexemes, and groups of lexemes are called lemmas, which are often the unit used for describing the size of a lexicon.
- Lexical analysis is the process of trying to understand what words mean, intuit their context, and note the relationship of one word to others.
- In terms of NLP, lexical analysis the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of lexical tokens (strings with an assigned and thus identified meaning).
- A program that performs lexical analysis may be termed at tokenizer, lexer or scanner.
- Although scanner is also a term for the first stage of a lexer. A lexer is generally combined with a parser, which together analyze the syntax of the sentences, texts or dialogues under investigation.

The Role of the Lexicon in Parsing

The role of Lexicon in Parsing

- Serves as the starting point for POS tagging.
- Provides additional information such as subcategorization with frames and hence syntactic rules.
- For Verbs:
  The different subcategorizations of the verb "think" vs "laugh".
- For adjectives:
  [4.50] I'm angry with Mary. vs I'm angry at Mary.
  [4.51] I'm mad at Mary. vs I'm mad with Mary. [x]
- For nouns:
  [4.52] Jane has a passion for old movies.
  [4.53] Jane has an interest in old movies.

Recursion in Grammar Rules

- The set of sentences in English is large (maybe even infinite).
- We want a concise (i. e., much shorter than a list of sentences) definition of it.
- We have a finite (in fact quite small) set of constituent types (NP, VP, etc.) from which to build our description.

So we appeal to recursion and write grammar rules such as:

S → NP VP	[4.54] My good friend John buys a flat.
VP → V NP	[4.55] buys a flat.
NP → NP PP	[4.56] My good friend
NP → NP S	[4.57] The boy who come early today won the game.
PP → prep NP	[4.58] The cupcake with sprinkles is yours.

A Context-Free Grammar (CFG) for English

If we ignore:

- subcategorization
- agreement
- gapping

Then we can build a context-free grammar for English that does a pretty good job of:

- generating all and only the acceptable sentences, and of
- building reasonable parse trees for those sentences.

We'll look at whether English is formally context free later.

What is Context-Free-Grammar (CFG)?

What is Context Free Language (CFL)?

- Context-free languages (CFLs) are generated by context-free grammars.
- Context Free Language is a superset of Regular Language.
- This means every Regular Language is a Context Free Language but not all Context Free Language is a Regular Language.
- In short:
  - Recursively enumerable Language is the largest set.
  - Context Sensitive Language is a subset of Recursively enumerable Language
  - Context Free Language is a subset of Context Free Language
  - Regular Language is a subset of Context Free Language
- The set of all context-free languages is identical to the set of languages accepted by pushdown automata, and the set of regular languages is a subset of context-free languages.
- An input language is accepted by a computational model if it runs through the model and ends in an accepting final state.
- Most arithmetic expressions are generated by context-free grammars, and are therefore, context-free languages.
- Context Free Language is closed in a specific operation if applying the operation on CFL results in a Language that also CFL. Such operations are:
  - Union Operation, Concatenation, Kleene closure, Reversal operation, Substitution, Prefix operation, Quotient with regular language, Cycle operation, Union with regular language, Intersection with regular language, Difference with regular language, Homomorphism, Inverse Homomorphism
- Context-free languages and context-free grammars have applications in computer science and linguistics such as natural language processing and computer language design.

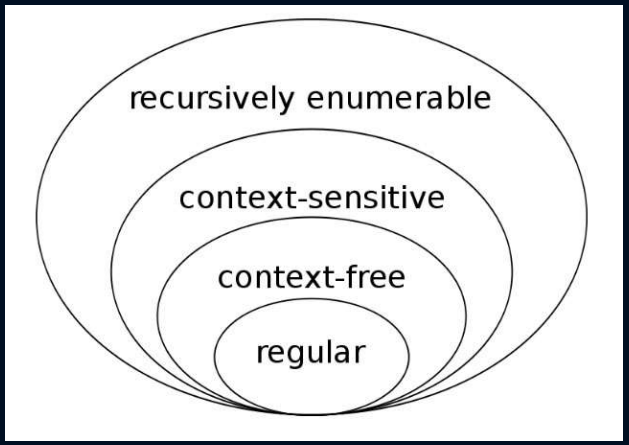

Fig. 4.5 Level of Languages

Context-Free Grammars (CFG)

What is Context Free Grammar (CFG)?

- Context-free grammars (CFGs) are used to describe context-free languages.
- A context-free grammar is a set of recursive rules used to generate patterns of strings.
- Context-free grammars are named as such because any of the production rules in the grammar can be applied regardless of context—it does not depend on any other symbols that may or may not be around a given symbol that is having a rule applied to it.
- A context-free grammar can describe all regular languages and more, but they cannot describe all possible languages.
- Context-free grammars are studied in fields of theoretical computer science, compiler design, and linguistics.
- CFG's are used to describe programming languages and parser programs in compilers can be generated automatically from context-free grammars.

Properties of CFG

- A context-free grammar (CFG) has FOUR major components:
    1. A set of non-terminal symbols N - which are placeholders for patterns of terminal symbols that can be generated by the nonterminal symbols. These are the symbols that will always appear on the left-hand side of the production rules, though they can be included on the right-hand side. The strings that a CFG produces will contain only symbols from the set of nonterminal symbols.
    2. A set of terminals Σ (disjoint from N) - Terminal symbols are the characters that appear in the language/strings generated by the grammar. Terminal symbols never appear on the left-hand side of the production rule and are always on the right-hand side.
    3. A set of productions P, each of the form $A \rightarrow \alpha$, where A is a non-terminal and α is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
    4. A designated start symbol S - A start symbol which is a special nonterminal symbol that appears in the initial string generated by the grammar.
- In our grammar of English
    - Σ is the set of POS discussed in Chapter 3, and
    - N is the set of remaining constituent types, e.g., NP, VP, PP

Derivations Using CFGs

The standard formal definition:

Assume: L_G generated by grammar G is the set of strings composed of terminal symbols which can be derived from the designated start symbol S.

$$L_G = \{w \mid w \text{ is in } \Sigma^* \text{ and } S \Rightarrow w\} \qquad (4.1)$$

But we won't generally want our grammar to have to all the way to words.

We want to let the lexicon do that.

That's why we let Σ be the set of POS.

So, the grammar may generate strings such as:

N V Det N (4.2)

Derivations Using CFGs

So, we will use the following definition:

L_G = {s | w is in Σ^* and S $\Rightarrow$ w and s can be derived from w by substituting words for POS as licensed by the lexicon} (4.3)

Note that this doesn't change the formal picture.

We could instead augment our grammar with tens of thousands of rules of the form:

$S \rightarrow NP\ VP$ (4.4)

It is the basic grammar rule in which a sentence is generated from a Noun-Phrase and a Verb-Phrase, which can be further "de-composed" into the grammar rules of the NP and VP, and so on, as shown in Fig. 4.6.

Context-Free Grammars and Parse Trees

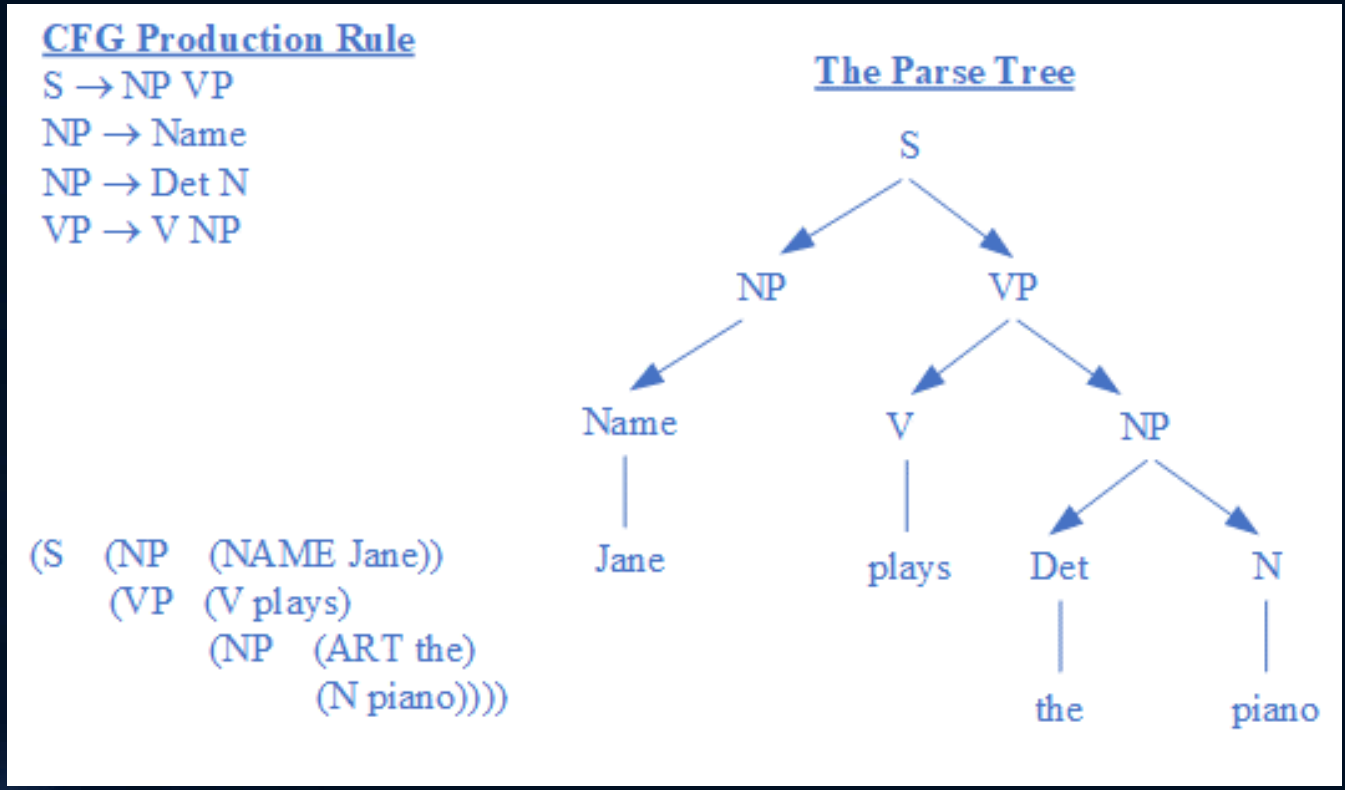

Fig. 4.6 CFG rule and corresponding Parse Tree for the sentence "Jane plays the piano"

Part 2
CFG Parsing

Analyzing Linguistic Units

Morphological parsing:

- It is the process of determining the morphemes from which a given word is constructed.
- The goal of morphological parsing is to find out what morphemes a given word is built from.
- For example, a morphological parser should be able to tell us that the word cats is the plural form of the noun stem cat, and that the word mice is the plural form of the noun stem mouse.
- So, given the string cats as input, a morphological parser should produce an output that looks similar to cat N PL.
- rule-based, FSA (Finite State Automata), FST (Finite State Transducer) which inputs words and outputs their stem and modifiers. The FST is initially created through algorithmic parsing of some word source, such as a dictionary, complete with modifier markups.
- With the advancement of neural networks in natural language processing, it became less common to use FST for morphological analysis, especially for languages for which there is a lot of available training data.

Phonological parsing:

- Phonological processing is the use of the sounds of one's language (i.e., phonemes) to process spoken and written language (Wagner & Torgesen, 1987).
- The broad category of phonological processing includes: 1) phonological awareness, 2) phonological working memory, and 3) phonological retrieval.
- All three components of phonological processing are important for speech production as well as the development of spoken and written language skills.
- Therefore, it is important and necessary to monitor the spoken and written language development of children with phonological processing difficulties.
- Phonological parsing is based on the analyze sounds into words and phrases for the creation of parser.

Analyzing Linguistic Units

Syntactic parsing:

- identify component parts and how related
- to see if a sentence is grammatical
- to assign an abstract representation of meaning
- Declarative formalisms like CFGs define the legal strings of a language but don't specify how to recognize or assign structure to them
- Parsing algorithms specify how to recognize the strings of a language and assign each string one or more syntactic structures
- Parse trees useful for grammar checking, semantic analysis, MT, QA, information extraction, speech recognition…and almost every task in NLP

Parsing is a Form of Search

- Searching FSAs
  - Finding the right path through the automaton
  - Search space defined by structure of FSA
- Searching CFGs
  - Finding the right parse tree among all possible parse trees
  - Search space defined by the grammar
- Constraints provided by the input sentence and the automaton or grammar

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → this | that | the | a
S → VP	N → play | piano | guitar | flute
NP → Det Nom	V → play | include | prefer
NP →PropN	Aux → does, do
Nom → N Nom	Prep →on | from | to
Nom → N	PropN → Germany | Italy | Yamaha
Nom → Nom PP	
VP → V NP	

Fig. 4.7 A simplified example on English grammar and lexicon

Parse Tree for "Play the piano" for Prior CFG

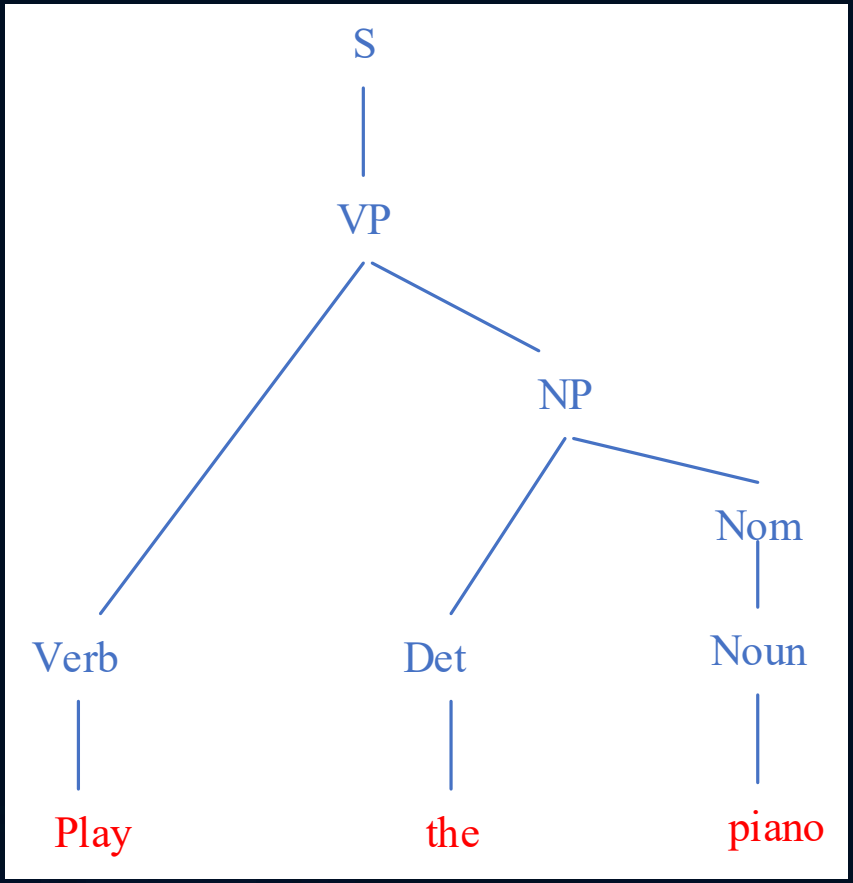

Fig. 4.8 Parse tree for the simple sentence "Play the piano"

Top-Down Parser

- Builds from the root S node to the leaves
- Find a rule to apply by matching the left-hand side (LHS) of a rule
- Build a tree by replacing LHS with the right-hand side (RHS)
- Assuming we build all trees in parallel:
  - Find all trees with root S (or all rules w/lhs S)
  - Next expand all constituents in these trees/rules
  - Continue until leaves are POS tokens.
  - Candidate trees failing to match pos of input string are rejected (e.g. Play the piano can only match subtree 5)

Top Down Space

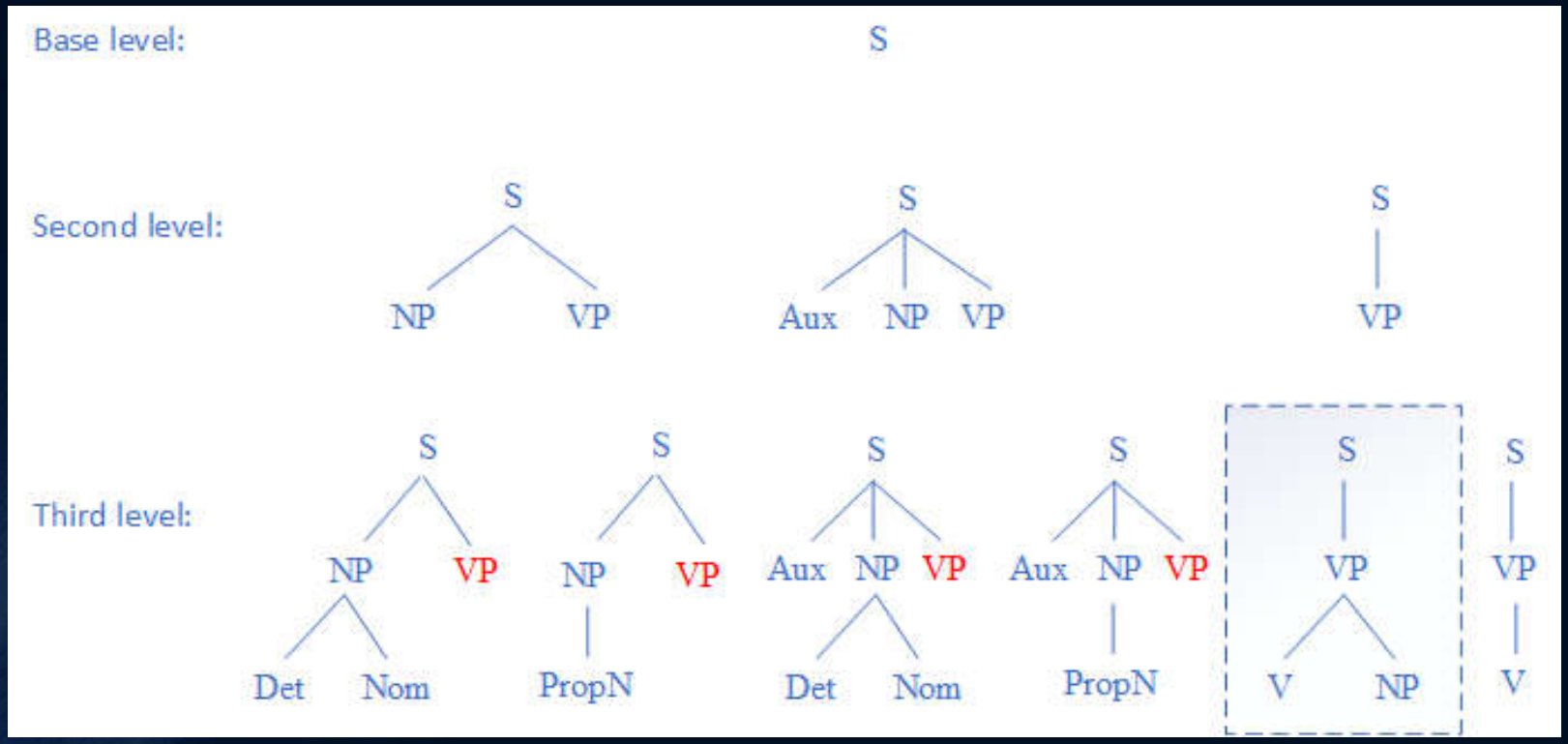

Fig. 4.9 A 3-level expansion of parse tree generation using Top-down Approach

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → this | that | the (5) | a
S → VP (1)	N → play | piano (7) | guitar | flute
NP → Det Nom (4)	V → play (3) | include | prefer
NP →PropN	Aux → does, do
Nom → N Nom	Prep →on | from | to
Nom → N (6)	PropN → Germany | Italy | Yamaha
Nom → Nom PP	
VP → V NP (2)	

Fig. 4.10 CFG Rules and Terminal/non-Terminal Nodes being used with Top-Down Approach Parsing

Bottom-Up Parsing

- Parser begins with words of input and builds up trees, applying grammar rules whose right-hand side (RHS) match
  - Play the piano

N	Det	N	V	Det	N
Play	the	piano	Play	the	piano

  - ‘play’ ambiguous, can be either a Noun or a Verb.
  - Parse continues until an S root node reached or no further node expansion possible

Bottom-Up Parsing Example

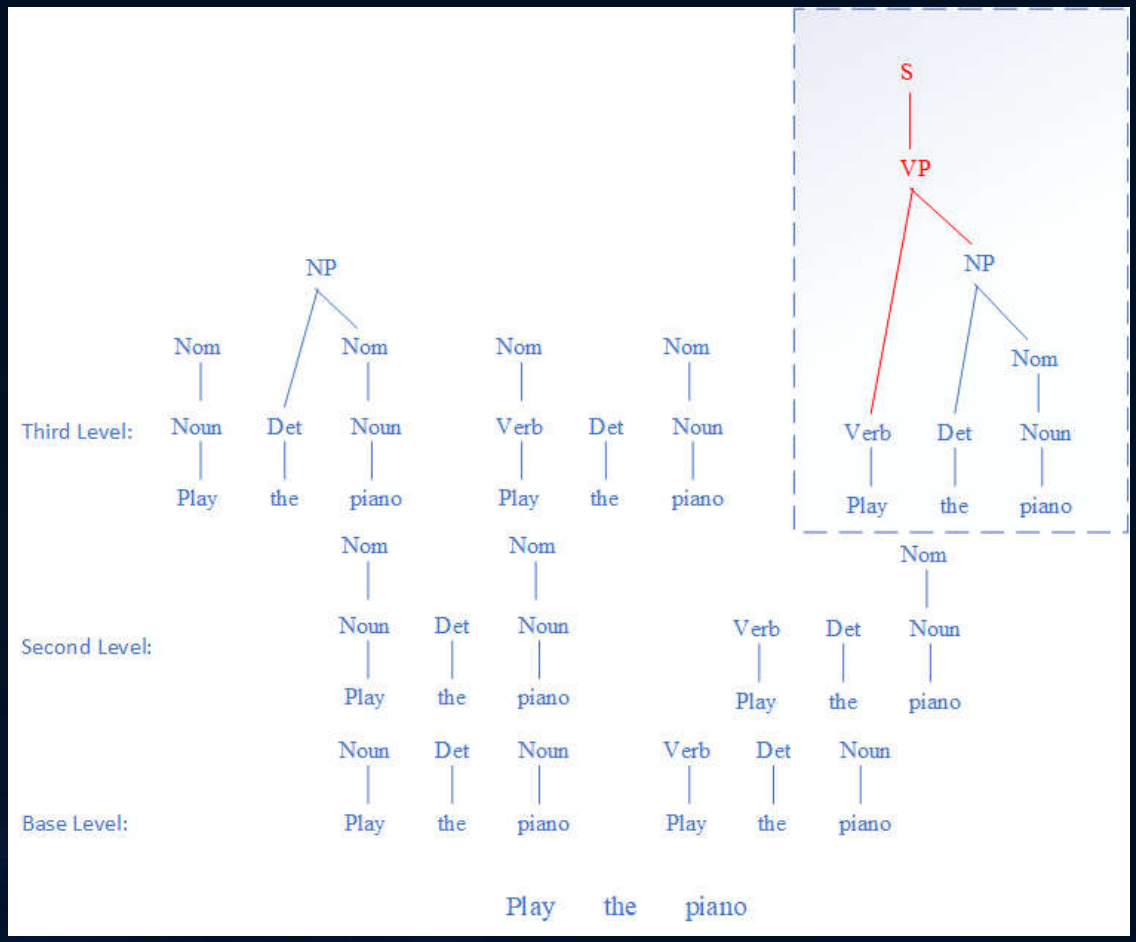

Fig. 4.11 A 3-level expansion of parse tree generation using Bottom-up Approach

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → this | that | the (2) | a
S → VP (7)	N → play | piano (3) | guitar | flute
NP → Det Nom (5)	V → play (1) | include | prefer
NP →PropN	Aux → does, do
Nom → N Nom	Prep →on | from | to
Nom → N (4)	PropN → Germany | Italy | Yamaha
Nom → Nom PP	
VP → V NP (6)	

Fig. 4.12 CFG Rules and Terminal/non-Terminal Nodes being used with Bottom-up Approach Parsing

Control

- Of course, we left out how to keep track of the spaces and how to make choices
  - Which node to try to expand next
  - Which grammar rule to use to expand a node

Top-down vs Bottom-up Approach Pros and Cons

- Top-Down Parsing Approach
  - Pros:
    - As it start with the root S node, it always can generate a parse tree unless the sentence has syntactic error.
    - In other words, it never explore the parse won't end up with to the root node S.
  - Cons:
    - As this approach don't consider the final token tags during parsing, it might waste a lot of time for tree that is totally unrelated to the correct result.
    - E.g. In Fig 4.9, as "play" should be parsed as "Verb" instead of "Noun", all the parse tree expansion for the first 4 parse tree with "play" parsed as Verb are totally waste of effort.
- Bottom-up Parsing Approach
  - Pros:
    - As it start from the sentence tokens (and its POS) for parse tree generation, it will always generate parse trees with all the tokens (POS) inside the sentence being fully considered and won't waste time on the rules that are not related to these tokens (POS).
  - Cons:
    - As it start from the leave node instead of the root node S, the bottom-up parse might all the time end-up with broken tree that cannot "merge up" to the root S to complete the parse tree.
    - As shown in Fig. 4.11, all the parse trees except the last one (also the correct one) are end-up with broken tree without merge-up to the root node S, which become waste of time especially when the sentence structure is long and/or exist many possible parsing alternatives.
- In next section, we try to explore the Lexical and Probabilistic Parsing which provides a new horizon on parsing alternatives.

Part 3

Lexicalized and Probabilistic Parsing

Why Using Probabilities in Parsing?

- Resolving ambiguities:

  [4.59] I saw the Jane with the telescope. (Jane with telescope or I use telescope to see Jane?)

  [4.60] I saw the Great Pyramid flying over Giza plateau. vs
  [4.61] I saw UFO flying over Giza plateau.

  - both situation although are pragmatic problem, it can be resolved using probabilities in parsing – why?

- Word prediction in voice recognition:

  [4.62] I have to go. vs.

  [4.63] I half to go. *vs.*

  [4.64] If way thought I'd go.

  - Remember, when we discussed the N-gram probabilities on Adventures of Sherlock Holmes, we have studied the usage of: I have, I should, I would, etc and their Bigram probabilities, such idea can be used in parsing.

It's Mostly About Semantics

[4.65] Jack drew one card from a desk. [?]

[4.66] Jack drew one card from a deck. ("drew one card" clearly semantic)

[4.67] I saw the Great Pyramid flying over Giza plateau. [?]

[4.68] I saw a UFO flying over Giza plateau. (Movable vs non-movable objects)

[4.69] The workers dumped sacks into a pin. [?]

[4.70] The workers dumped sacks into a bin. ("Dump" looks for a locative complement)

[4.71] John hit the ball with the pen. [?]

[4.72] John hit the ball with the bat.

[4.73] Visiting relatives can be boring. [?]

[4.74] Visiting museums can be boring. [?]

Note: Visiting relatives is genuinely ambiguous. Visiting museums is easy since only animate things can visit. With enough data, we don't even need the abstraction "animate thing".

How to Add Semantics to Parsing?

The classic approach to this problem:

Ask a semantics module to choose.

Two ways to do that:

1. Cascade the two systems. Build all the parses, then pass them to semantics to rate them. Combinatorially awful.
2. Do semantics incrementally. Pass constituents, get ratings and filter.

In either case, we need to reason about the world.

The "Modern" Approach

The modern approach:

Skip "meaning" and the corresponding need for a knowledge base and an inference engine.

Notice that the facts about meaning manifest themselves in probabilities of observed sentences ***if*** there are enough sentences.

Why is this approach in vogue?

- Building world models is a lot harder than early researchers realized.
- But we do have huge text corpora from which we can draw statistics.

Probabilistic Context-Free Grammars (PCFG)

What is PCFG?

- A probabilistic context free grammar (or PCFG) is a context free grammar that associates a probability with each of its rules.
- It generates the same set of parses for a text that the corresponding context free grammar does, and assigns a probability to each parse.
- The probability of a parse generated by a PCFG is simply the product of the probabilities of the rules used to generate it.
- A PCFG is a context-free grammar in which each rule has been augmented with a probability, given as:

  $A \rightarrow \beta$ [p] (4.5)

  is the probability that a given nonterminal symbol A will be rewritten as β *via this rule*.
- Another way to think of this is:

  $P(A \rightarrow \beta|A)$ (4.6)

  Note: the sum of all the probabilities of rules with left hand side A must be 1.
- 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.

A Simple Example of Buy Coffee from Starbucks

CFG Rules	Prob	CFG Rules	Prob
S → NP VP	[0.82]	Det → a[.12] | that[.03] | the[.75] | this[.10]	
S → Aux NP VP	[0.12]	N → coffee	[0.75]
S → VP	[0.06]	N → tea	[0.13]
NP → Det Nom	[0.21]	N → food	[0.12]
NP →Proper-N	[0.37]	V → buy	[0.41]
NP → Nom	[0.06]	V → pay	[0.27]
NP →Pronoun	[0.36]	V → order	[0.32]
Nom → Noun	[0.72]	Aux → do	[0.31]
Nom → N Nom	[0.23]	Aux → does	[0.26]
Nom → Proper-N Nom	[0.05]	Aux → can	[0.43]
VP → V	[0.58]	Proper-N → Starbucks	[0.63]
VP → V NP	[0.36]	Proper-N → KFC	[0.37]
VP → V NP NP	[0.06]	Pronoun → I[.42]|you[.36]| he[.12]|she[.10]	

Fig. 4.13 Sample CFG rules and their probabilities in AI Chatbot dialogues (food ordering at campus)

How Can We Use These?

In a top-down parser, we can follow the more likely path first.

In a bottom-up parser, we can build all the constituents and then compare them.

The Probability of Some Parse Trees

$$P(T) = \prod_{n \in T} p(r(n))$$ (4.7)

where $p(r(n))$ means the probability that rule r will apply to expand the nonterminal n.

Note the independence assumption.

So what we want is:

$$\hat{T}(S) = \arg\max_{T \in \tau(S)} P(T)$$ (4.8)

where τ(S) is the set of possible parses for S.

Can you buy Starbucks coffee?
Which one is more probable?

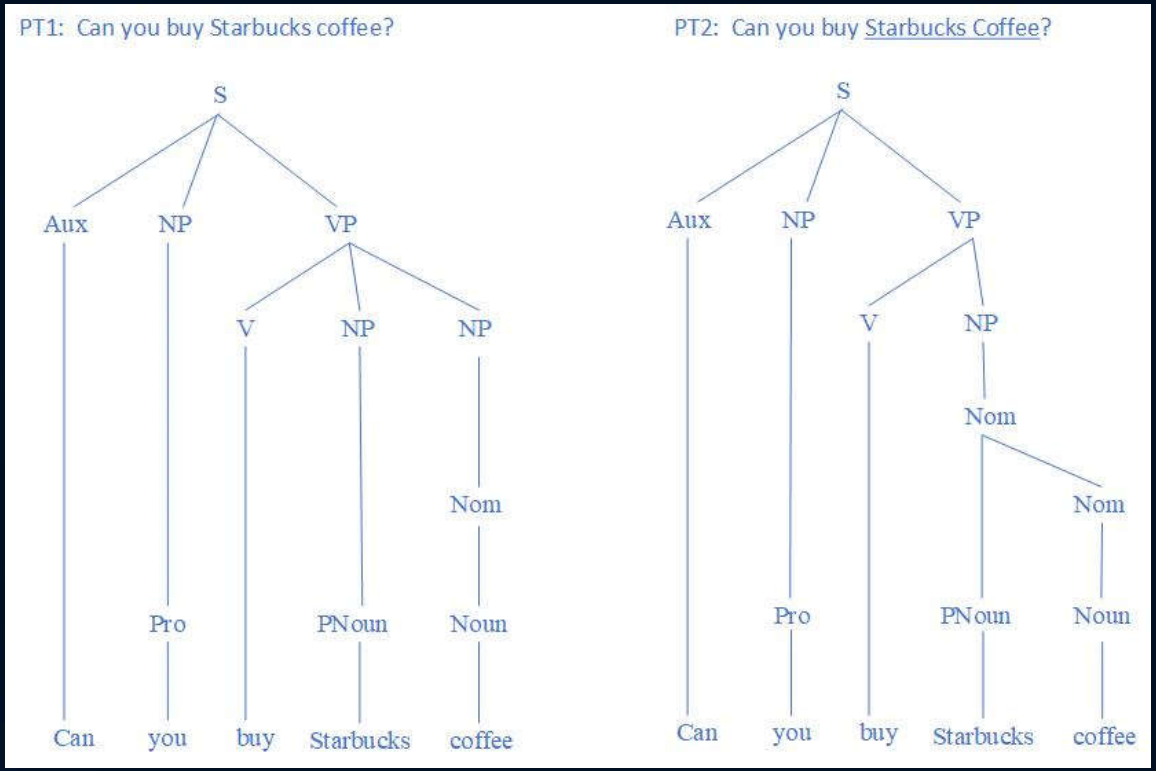

Fig. 4.14 Two possible parse trees for the utterance "Can you buy Starbucks coffee"?

Two ambiguous meaning (which is which?)

1) "Can you buy coffee from Starbucks?" – "Starbucks" and "Coffee" are treated as TWO separated NPs.

2) "Can you buy food (can be coffee or not) from Starbucks Coffee? "- Starbuck Coffee" is treated as a single NP.

Which meaning is more probable?

CFG Rules for TP1	Prob	CFG Rules for TP2	Prob
S → Aux NP VP	[0.12]	S → Aux NP VP	[0.12]
NP →Pronoun	[0.36]	NP →Pronoun	[0.36]
VP → V NP NP	[0.06]	VP → V NP	[0.36]
NP → Nom	[0.06]	NP → Nom	[0.06]
NP →Proper-N	[0.37]	Nom → Proper-N Nom	[0.05]
Nom → Noun	[0.72]	Nom → Noun	[0.72]
Aux → can	[0.43]	Aux → can	[0.43]
Pronoun → you	[0.36]	Pronoun → you	[0.36]
V → buy	[0.41]	V → buy	[0.41]
Proper-N → Starbucks	[0.63]	Proper-N → Starbucks	[0.63]
N → coffee	[0.75]	N → coffee	[0.75]

Fig. 4.15 CFG rules and associated probabilities for the two possible parse trees PT1 vs PT2

P(PT1) = .12*.36*.06*.06*.37*.72*.43*.36*.41*.63*.75 = 1.242×10^{-6}

P(PT2) = .12*.36*.36*.06*.05*.72*.43*.36*.41*.63*.75 = 1.007×10^{-6}

Which one we pick?

Note how small the probabilities are, even with this tiny grammar.

Using Probabilities for Language Modeling

- In fact, one can consider the probabilistic parsing is a kind of integration of the N-gram frequency probability concept together with the parse tree formation.
- Since there are fewer grammar rules than there are word sequences, it can be useful, in language modeling, to use grammar probabilities instead of pure n-gram frequencies.
- So, the probability of some sentence S is the sum of the probabilities of its possible parses:

$$P(S) = \sum_{T \in \tau(S)} P(T) \qquad (4.9)$$

Contrast with that from the N-gram probability calculation mentioned in Chapter 3 :

$$P(S) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1 w_2) \times P(w_4 \mid w_1 w_2 w_3) \ldots \qquad (4.10)$$

Adding Probabilities to a Parser

- Adding probabilities to a top-down parser, e.g., Earley Parsing:

  This is easy since we're going top-down, we can choose which rule to prefer.

- Adding probabilities to a bottom-up parser:

  At each step, build the pieces, then add probabilities to them.

Limitations to Attaching Probabilities Just to Rules

Sometimes it's enough to know that one rule applies more often than another:

Can you buy Starbucks coffee? vs. Can you buy KFC chicken?

But often it matters what the context is. Consider:

$S \rightarrow NP\ VP$

$NP \rightarrow Pronoun$ [0.80]

$NP \rightarrow LexNP$ [0.20] (4.11)

But, when the NP is the subject, the true probability of a pronoun is .91.
When the NP is the direct object, the true probability of a pronoun is .34.

Often the Probabilities Depend on Lexical Choices

[4.75] I saw the Great Pyramid flying over Giza Plateau vs

[4.76] I saw a UFO flying over Giza Plateau.

[4.77] Farmer dumped sacks into the bin.

[4.78] Farmer dumped sacks of apples.

[4.79] John hit the ball with the bat vs

[4.80] John hit the ball over the net.

[4.81] Visiting relatives can be boring vs

[4.82] Visiting museums can be boring.

[4.83] There were boys in park and girls vs

[4.84] There were boys in park and shops.

Boys in Park and Girls Example

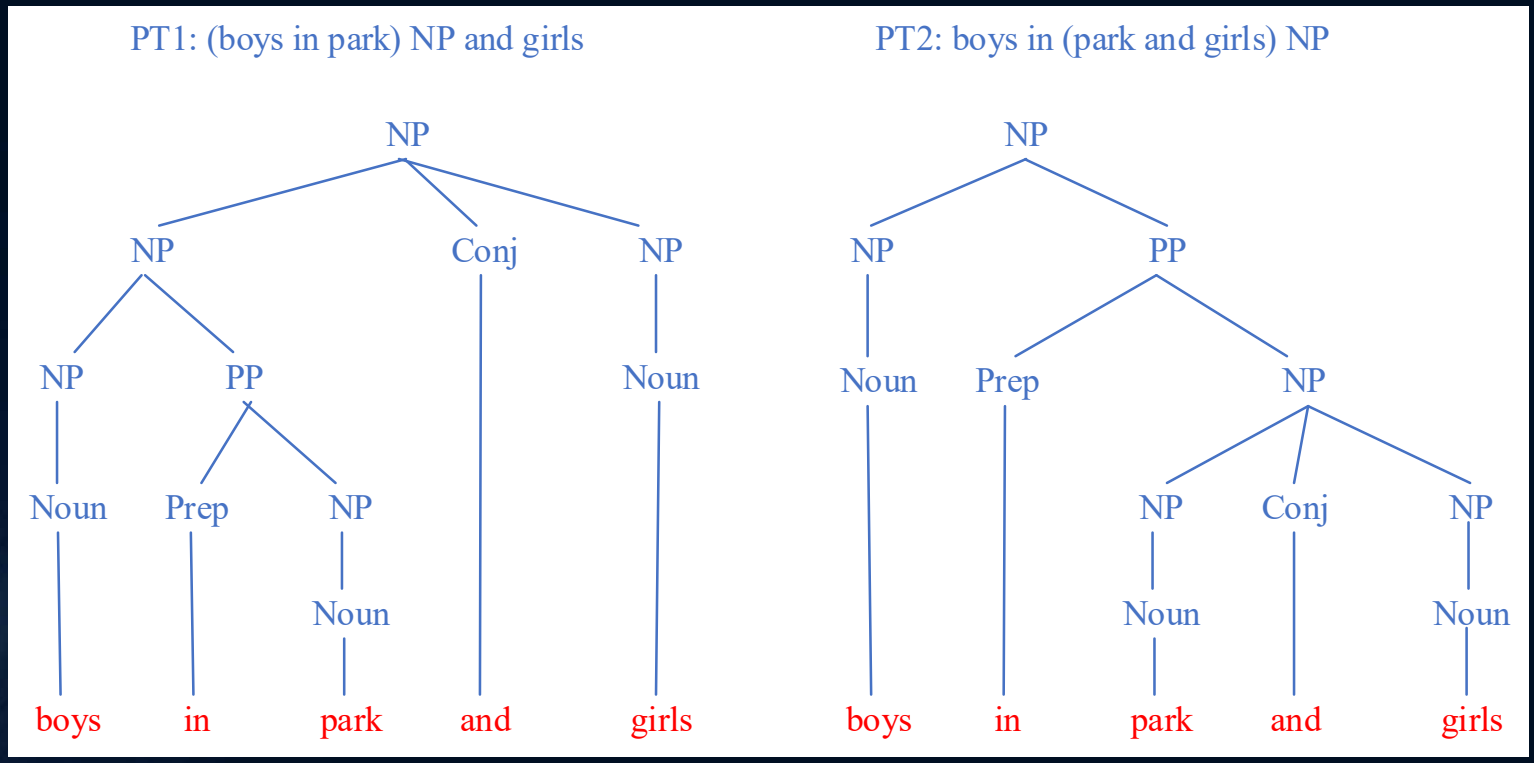

Fig. 4.16 Two interpretations of the utterance "boys in park and girls"

The problem is that both parses used the same rules so they will get the same probabilities assigned to them.

The Fix – Use the Lexicon

- The lexicon is an approximation to a knowledge base.
- It will let us treat *into* and *to* differently with respect to *throws* without any clue what *throws* means or what *into* and *to* mean.
- Note the difference between this approach and subcategorization rules, e.g.,

throw [SUBCAT NP] (4.12)
[SUBCAT LOCATION]

Subcategorization rules specify *requirements*, not preferences.

Lexicalized Trees

Key idea: Each constituent has a HEAD word:

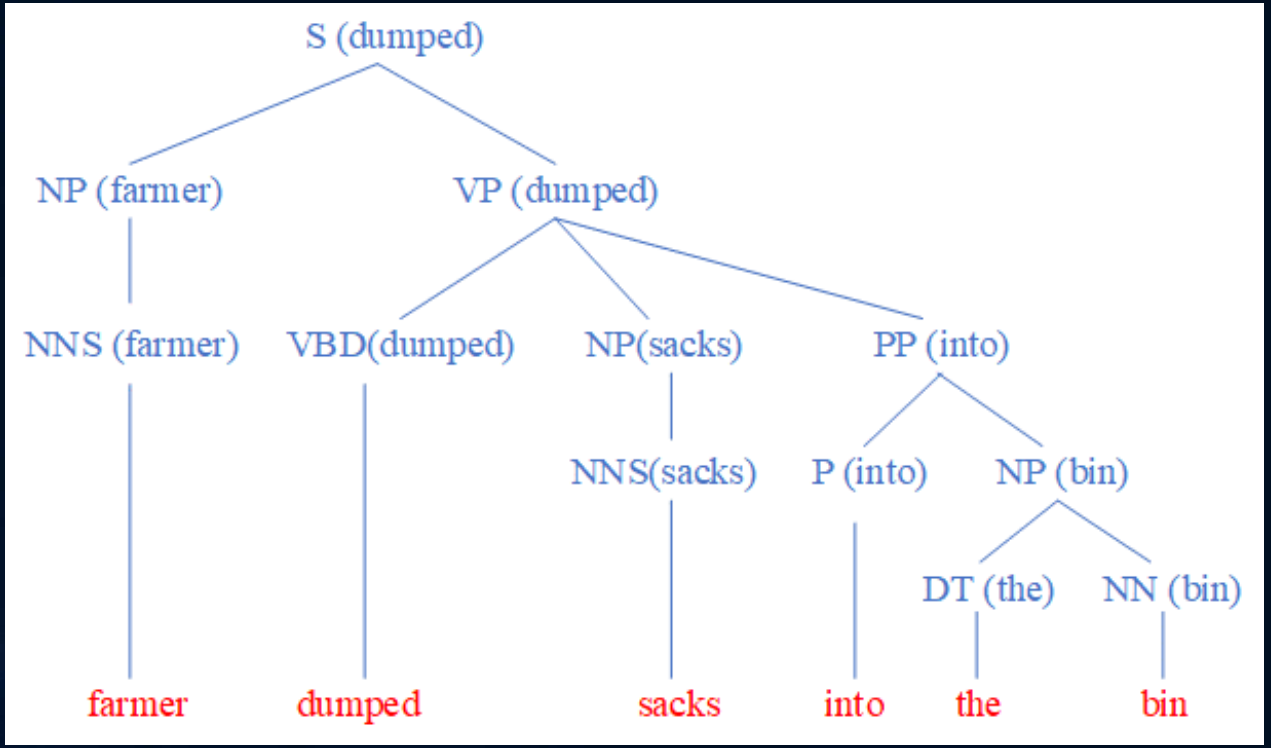

Fig. 4.17 Lexical tree for the utterance "workers dumped sacks into a bin"

Adding Lexical Items to the Rules

VP(dumped) → VBD (dumped) NP (sacks) PP (into)	8×10^{-10}	(4.13)
VP(dumped) → VBD (dumped) NP (cats) PP (into)	1×10^{-10}	(4.14)
VP(dumped) → VBD (dumped) NP (stones) PP (into)	2×10^{-10}	(4.15)
VP(dumped) → VBD (dumped) NP (sacks) PP (above)	1×10^{-12}	(4.16)

We need fewer numbers than we would for N-gram frequencies:

The farmer dumped sacks of apples into a bin.
The farmer dumped sacks of peaches into a bin.
The farmer dumped all the sacks of apples into a bin.

But there are still too many and most will be 0 in any given corpus.

Collapsing These Cases

Instead of caring about specific rules like:

$VP(dumped) \rightarrow VBD\ (dumped)\ NP\ (sacks)\ PP\ (into) \quad 8 \times 10^{-10}$ (4.13)

Or about very general rules like:

$VP \rightarrow VBD\ NP\ PP$ (4.17)

We'll do something partway in between:

$VP(dumped) \rightarrow VBD\ NP\ PP \quad p(r(n) \mid n, h(n))$ (4.18)

Computing Probabilities of Heads

We'll let the probability of some node n having head h depend on two factors:

- the syntactic category of the node, and
- the head of the node's mother (h(m(n)))

So, we will compute:

$$P(h(n) = word_i \mid n, h(m(n)))$$

VP (dumped)	VP (dumped)	NP (sacks)
$p = p_1$	$p = p_2$	$p = p_3$
PP (into)	PP (of)	PP (of)

(4.19)

So now we've got probabilistic subcat information.

Revised Rule for Probability of a Parse

Our initial rule:

$$P(T) = \prod_{n \in T} p(r(n)) \qquad (4.20)$$

where $p(r(n))$ means the probability that rule r will apply to expand the nonterminal n.

Our new rule:

$$P(T) = \prod_{n \in T} p(r(n) \mid n, h(n)) \times p(h(n) \mid n, h(m(n))) \qquad (4.21)$$

probability of choosing this rule given the nonterminal and its head ×

probability that this node has head h given the nonterminal and the head of its mother

So We Can Solve the Dumped Sacks Problem

From the Brown corpus:

$$p(VP \rightarrow VBD\ NP\ PP \mid VP, \textit{dumped}) = .67$$
$$p(VP \rightarrow VBD\ NP \mid VP, \textit{dumped}) = 0 \qquad (4.22)$$
$$p(\textit{into} \mid PP, \textit{dumped}) = .22$$
$$p(\textit{into} \mid PP, \textit{sacks}) = 0$$

So, the contribution of this part of the parse to the total scores for the two candidates is:

$$\text{[dumped into]} \qquad .67 \times .22 = .147 \qquad (4.23)$$
$$\text{[sacks into]} \qquad 0 \times 0 = 0$$

It's Mostly About Semantics But It's Also About Psychology

What do people do?

People have limited memory for processing language.

So, we should consider two aspects of language skill:

- competence (what could we in principle do?), and
- performance (what do we actually do, including mistakes?)

Summary

- Discuss motivation and importance of Syntax and Syntactic Analysis in Linguistics and NLP
- Introduce difference types of constituents in English language
- *Discuss what is a Parse Tree and Syntactic Parsing in NLP*
- *Introduce Context Free Grammar (CFG) and its roles in linguistics*
- *Discuss CFG Parsing technique and algorithms*
- *Introduce PCFG Parsing*
- *Discuss Lexical Parsing how its can help to solve the syntactic ambiguous issues*
- *Some other methods including: In addition to Top-down and Bottom-up approach of parsing, other method such as Earley Algorithm, Finite-State Parsing (FSP) and even AI technique using Deep Neural Networks (DNN) and Recurrent Neural Network (RNNs).*

Next

NLP#08 Meaning Representation