Artificial Intelligence

Subject Code: 20A05502T

UNIT IV – Natural Language Communication

- Communication
- Grammer
- Natural Language for Communication: Phrase structure grammars, Syntactic Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition

Communication

- A computer can use natural language to communicate with humans and learn from what they have written.

- Alan Turing proposed his test for intelligence, he based it on language, not art or haberdashery, perhaps because of its universal scope and because language captures so much of intelligent behavior:
- a speaker (or writer) has the goal of communicating some knowledge, then plans some language that represents the knowledge, and acts to achieve the goal.
- The listener (or reader) perceives the language, and infers the intended meaning.
- This type of communication via language has allowed civilization to grow; it is our main means of passing along cultural, legal, scientific, and technological knowledge

- There are three primary reasons for computers to do natural language processing (NLP)
- To communicate with humans.
- To learn.
- To advance the scientific understanding of languages and language use, using the tools of AI in conjunction with linguistics, cognitive psychology, and neuroscience.

Grammar -Backus–Naur Form (BNF)

- write down a grammar for the language of first-order logic.
- A grammar is a set of rules that defines the tree structure of allowable phrases, and a language is the set of sentences that follow those rules.
- Natural languages do not work exactly like the formal language of first-order logic—
- they do not have a hard boundary between allowable and unallowable sentences, nor do they have a single definitive tree structure for each sentence.

- Hierarchical structure is important in natural language.
- The word “Stocks” in “Stocks rallied on Monday” is not just a word, nor is it just a noun;
- in this sentence it also comprises a noun phrase, which is the subject of the following verb phrase.
- Syntactic categories such as noun phrase or verb phrase help to constrain the probable words at each point within a sentence, and
- the phrase structure provides a framework for the meaning or semantics of the sentence.

The probabilistic context-free grammar, or PCFG.

- A probabilistic grammar assigns a probability to each string, and “context-free” means that any rule can be used in any context:
- the rules for a noun phrase at the beginning of a sentence are the same as for another noun phrase later in the sentence, and
- if the same phrase occurs in two locations, it must have the same probability each time.
- We will define a PCFG grammar for a tiny fragment of English that is suitable for communication between agents exploring the wumpus world.

- We call this language E_0, A grammar rule such as

$$\begin{array}{lcll} Adjs & \rightarrow & Adjective & [0.80] \\ & | & Adjective\ Adjs & [0.20] \end{array}$$

The Wumpus World

- The grammar for E_n, with example phrases for each rule.
- The syntactic categories are sentence (S),
- noun phrase (NP),
- verb phrase (VP),
- list of adjectives (Adjs),
- prepositional phrase (PP) and
- relative clause (RelClause)

S	→	*NP VP*	[0.90]	I + feel a breeze
	|	*S Conj S*	[0.10]	I feel a breeze + and + It stinks
NP	→	*Pronoun*	[0.25]	I
	|	*Name*	[0.10]	Ali
	|	*Noun*	[0.10]	pits
	|	*Article Noun*	[0.25]	the + wumpus
	|	*Article Adjs Noun*	[0.05]	the + smelly dead + wumpus
	|	*Digit Digit*	[0.05]	3 4
	|	*NP PP*	[0.10]	the wumpus + in 1 3
	|	*NP RelClause*	[0.05]	the wumpus + that is smelly
	|	*NP Conj NP*	[0.05]	the wumpus + and + I
VP	→	*Verb*	[0.40]	stinks
	|	*VP NP*	[0.35]	feel + a breeze
	|	*VP Adjective*	[0.05]	smells + dead
	|	*VP PP*	[0.10]	is + in 1 3
	|	*VP Adverb*	[0.10]	go + ahead
Adjs	→	*Adjective*	[0.80]	smelly
	|	*Adjective Adjs*	[0.20]	smelly + dead
PP	→	*Prep NP*	[1.00]	to + the east
RelClause	→	*RelPro VP*	[1.00]	that + is smelly

- the grammar overgenerates: that is, it generates sentences that are not grammatical, such as “Me go I.”
- It also undergenerates: there are many sentences of English that it rejects, such as “I think the wumpus is smelly.”
- We will see how to learn a better grammar later; for now we concentrate on what we can do with this very simple grammar.

The lexicon of E_0

- The lexicon, or list of allowable words,
- Each of the lexical categories ends in … to indicate that there are other words in the category.
- nouns, names, verbs, adjectives, and adverbs, it is infeasible even in principle to list all the words.

Noun	→	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | …
Verb	→	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | …
Adjective	→	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] …
Adverb	→	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | …
Pronoun	→	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | …
RelPro	→	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] | …
Name	→	**Ali** [0.01] | **Bo** [0.01] | **Boston** [0.01] | …
Article	→	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | …
Prep	→	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | …
Conj	→	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] | …
Digit	→	**0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | …

Open Classes and Closed Classes

- The five categories (nouns, names, verbs, adjectives, and adverbs) are called open classes.
- Pronouns, relative pronouns, articles, prepositions, and conjunctions are called closed classes;
- they have a small number of words (a dozen or so), and change over the course of centuries, not months.
- For example, “thee” and “thou” were commonly used pronouns in the 17th century, were on the decline in the 19th century, and are seen today only in poetry and some regional dialects

Parsing

- Parsing is the process of analyzing a string of words to uncover its phrase structure, according to the rules of a grammar.
- It is a search for a valid parse tree whose leaves are the words of the string.

List of items	Rule
S	
NP VP	*S* → *NP VP*
NP VP Adjective	*VP* → *VP Adjective*
NP Verb Adjective	*VP* → *Verb*
NP Verb **dead**	*Adjective* → **dead**
NP **is dead**	*Verb* → **is**
Article Noun **is dead**	*NP* → *Article Noun*
Article **wumpus is dead**	*Noun* → **wumpus**
the wumpus is dead	*Article* → **the**

List of items	Rule
S	
NP VP	*S* → *NP VP*
NP VP Adjective	*VP* → *VP Adjective*
NP Verb Adjective	*VP* → *Verb*
NP Verb **dead**	*Adjective* → **dead**
NP **is dead**	*Verb* → **is**
Article Noun **is dead**	*NP* → *Article Noun*
Article **wumpus is dead**	*Noun* → **wumpus**
the wumpus is dead	*Article* → **the**

- Parsing the string "The wumpus is dead" as a sentence, according to the grammar E_0
- Viewed as a topdown parse, we start with S, and on each step match one nonterminal with a rule of the form (X→ Y ..) and replace X in the list of items with Y…;
- for example replacing with the sequence NPVP.
- Viewed as a bottom-up parse, we start with the words "the wumpus is dead", and on each step match a string of tokens such as (Y…) against a rule of the form (X→ Y ..) and replace the tokens with X; for example replacing "the" with Article or Article Noun with NP.

- start with the S symbol and search top down, or we can start with the words and search bottom up.
- Pure top-down or bottom-up parsing strategies can be inefficient, however, because they can end up repeating effort in areas of the search space that lead to dead ends.
- Consider the following two sentences:
- Have the students in section 2 of Computer Science 101 take the exam.
- Have the students in section 2 of Computer Science 101 taken the exam?

- Even though they share the first 10 words, these sentences have very different parses, because the first is a command and the second is a question.
- A left-to-right parsing algorithm would have to guess whether the first word is part of a command or a question and will not be able to tell if the guess is correct until at least the eleventh word, take or taken.
- If the algorithm guesses wrong, it will have to backtrack all the way to the first word and reanalyze the whole sentence under the other interpretation.

dynamic programming

- To avoid this source of inefficiency we can use dynamic programming: every time we analyze a substring, store the results so we won't have to reanalyze it later.
- For example, once we discover that "the students in section 2 of Computer Science 101" is an , we can record that result in a data structure known as a chart.
- An algorithm that does this is called a chart parser.
- Because we are dealing with context-free grammars, any phrase that was found in the context of one branch of the search tree can work just as well in any other branch of the search tree.
- There are many types of chart parsers;
- A probabilistic version of a bottom-up chart parsing algorithm called the CYK algorithm, after its inventors, Ali Cocke, Daniel Younger, and Tadeo Kasami.

CYK algorithm

- It requires a grammar with all rules in one of two very specific formats:
- lexical rules of the form X → word[p], and syntactic rules of the form X → Y Z [p], with exactly two categories on the right-hand side.
- This grammar format, called Chomsky Normal Form, may seem restrictive, but it is not: any context-free grammar can be automatically transformed into Chomsky Normal Form

- The CYK algorithm for parsing. Given a sequence of words, it finds the most probable parse tree for the sequence and its subsequences.
- The table gives the probability of the most probable tree of category spanning The output table contains the most probable tree of category spanning positions to inclusive. The function SUBSPANS returns all tuples covering a span of with listing the tuples by increasing length of the span, so that when we go to combine two shorter spans into a longer one, the shorter spans are already in the table. LEXICALRULES(word) returns a collection of pairs, one for each rule of the form and GRAMMARRULES gives tuples, one for each grammar rule of the form Chomsky Normal Form P [X, i, k] X wordsi:k. T[X, i, k] X i k (i, j, k) wordsi:k, $i \leq j < k$, i : k (X, p) X → word [p] , (X, Y , Z, p) X → Y Z [p] .