Natural Language Processing (NLP): written and oral
Involves processing of written text using computer models at lexical, syntactic, and semantic level.

Introduction:

1. Understanding (of language) – map text/speech to immediately useful form.
2. Generation (of language)

Sentence Analysis Phases:

1. Morphological Analysis: extracts root word from declined/inflection form of word after removing suffices and prefixes.
2. Syntactic Analysis: builds a structural description of a sentence using MAP (Morphological Analysis Process) based on grammatical rules, called Parsing.
3. Semantic Analysis: Creates a semantic structure by ascribing a literal meaning to sentence using parse structure obtained in syntactic phase. It maps individual words into corresponding objects in the knowledge base. Focus is on creation of target representation of the meaning of a sentence.
4. Pragmatic Analysis: to establish meaning of a sentence in different contexts.
5. Discourse Analysis: Interpretation based on belief during conversation.

Grammars and Parsers:
Parsing: Process of analyzing an input sequence in order to determine its structure with respect to a given grammar.
Types of Parsing:

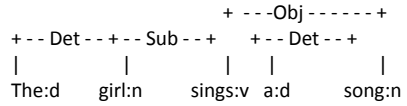
1. Rule-based parsing: syntactic structure of language is provided in the form of linguistic rules which can be coded as production rules that are similar to context-free rules.
    1. Top-down parsing: Start with start symbol and apply grammar rules in the forward direction until the terminal symbols of the parse tree correspond to the words in the sentence.
    2. Bottom-up parsing: Start with the words in the sentence in the sentence and apply grammar rules in the backward direction until a single tree is produced whose root matches with the start symbol.
2. Statistical parsing: requires large corpora and linguistic knowledge is represented as statistical parameters or probabilities, which may be used to parse a given sentence.

Types of Parsers:

1. Link Parser: system assigns to a sentence, a syntactic structure which consists of a set of labeled links connecting pairs of words. There are two basic parameters in this representation, directionality and distance. In an SVO (Subject, Verb, and Object) language like English, the verb would look left to form a subject link, and right to form an object link. Nouns would look right to complete subject link, and right to form an object link.
   Example:

SVO Language	
Link Grammar Rules	Interpretation of Rules
$< Determiner >: Det +$	Determiner is defined as label *Det* connected to word to its right in a sentence.
$< Noun\_Sub >$: {Det-} & Sub+,	Subject noun is defined as label *Sub* connected to word on its right in a sentence and connected to label *Det* on left in a sentence.
$< Noun\_Obj >$: {Det-} & Obj+,	Object noun is defined as the last word in a sentence and labeled as *Obj* is connected to label *Det* on left in a sentence
$< Verb >$: Sub- & {Obj+}	Verb is defined as a word connected to label *Obj* on right in a sentence and connected to label *Sub* on left.

   The parsed sentence "the girl sings a song." Appears as follows:



2. Chart Parser
3. Simple Transition Networks
4. Recursive Transition Networks
5. Augmented Transition Networks

Chart Parser

Chart parser store parses of intermediate constituents to be reused while trying alternative parsing path. A data structure called *chart* is maintained to keep a record of the state of a bottom-up parse traversed. This structure is a record of the positions of the words and new structure derived from the sentence. Chart also maintains the record of rules that have been applied previously but are not complete. These rules are recorded as active arcs on the chart. Chart parser provides exponential reductions of search space and enables coping with ambiguity in parsing. It is compatible with a high degree of flexibility relative to search and control strategies.

Parsing by Chart Parser

There are two valid parse structures for the sentence, "*the girls saw a man in the park with a cat"* using grammar defined below in Table 1 with propositional phrases. Both parse structures are shown in the figures 1 and 2 below.

Table 1. Sample Grammar

Grammar Rules	Rule Number
$< S > \rightarrow < NP >< VP >$	1
$< NP > \rightarrow < Det >< Noun >$	2
$< NP > \rightarrow < Det >< Noun >< PP >$	3
$< VP > \rightarrow < Verb >< NP >$	4
$< VP > \rightarrow < Verb >< NP >< PP >$	5
$< PP > \rightarrow < Prep >< NP >$	6

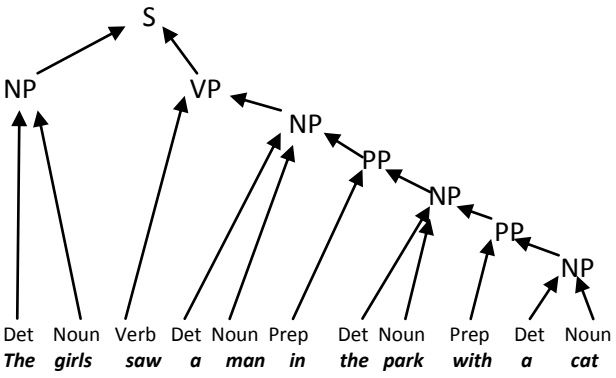


Figure 1. Parse Structure 1.

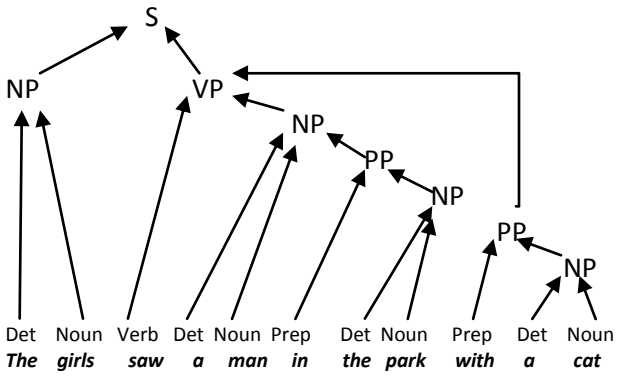


Figure 2. Parse Structure 2.

Transition Network TN: a convenient representation of grammar as network of nodes and labeled arcs.

Parsing:

It starts with start node and traversal takes place through arcs. If the current word in the sentence is in the category on that arc then move to the next state in TN and continue the process till *Pop* arc is reached. If there are no words left in the sentence at *Pop* arc then it is assumed to be correctly parsed.

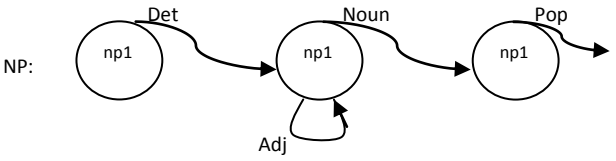


Figure 1. Transition Network for Noun Phrase NP.

Table 1. Context-Free Grammar corresponding to TN.

<NP> → <Det><NP1 >
<NP1 > → < Adj ><NP1 >
<NP1 > → < Noun >

Augmented Transition Network

Descriptive power of CFG is obtained by introducing recursion in network grammar allowing arc labels to refer to other networks rather than word categories (as in TN). Such a network is called as Recursive Transition Networks (RTN). RTN with registers and tests on these registers is an Augmented Transition Network (ATN).

Network Representation for Natural Language Parsing		Automata	describing Language
Transition Network, TN	is equivalent to	Finite State Automata	Regular Languages
Recursive Transition Network, RTN		Push Down Automata (Stack)	Context Free Languages
Augmented Transition Network, ATN (arc is augmented with conditions and sequence of actions)		Turing Machine (Tape/Register)	Recursively Enumerable Languages

We can record sentence structure while parsing through ATN that can be used for further analysis. For instance, we can identify one particular noun phrase as the syntactic subject (SUBJ) of a sentence and another NP as the syntactic object of the verb (OBJ). Within noun phrase we might identify the determiner, adjective, head noun, and so on. Thus, the sentence, "Jack found a bag", when parsed using ATN may produce the following structure:

```
(S    (    SUBJ        (NP NAME       jack)
MAIN-V    found
TENSE     PAST
OBJ       ( NP   DET     a
HEAD  bag
              )
        )
)
```

Figure 1. Parsed structure using ATN

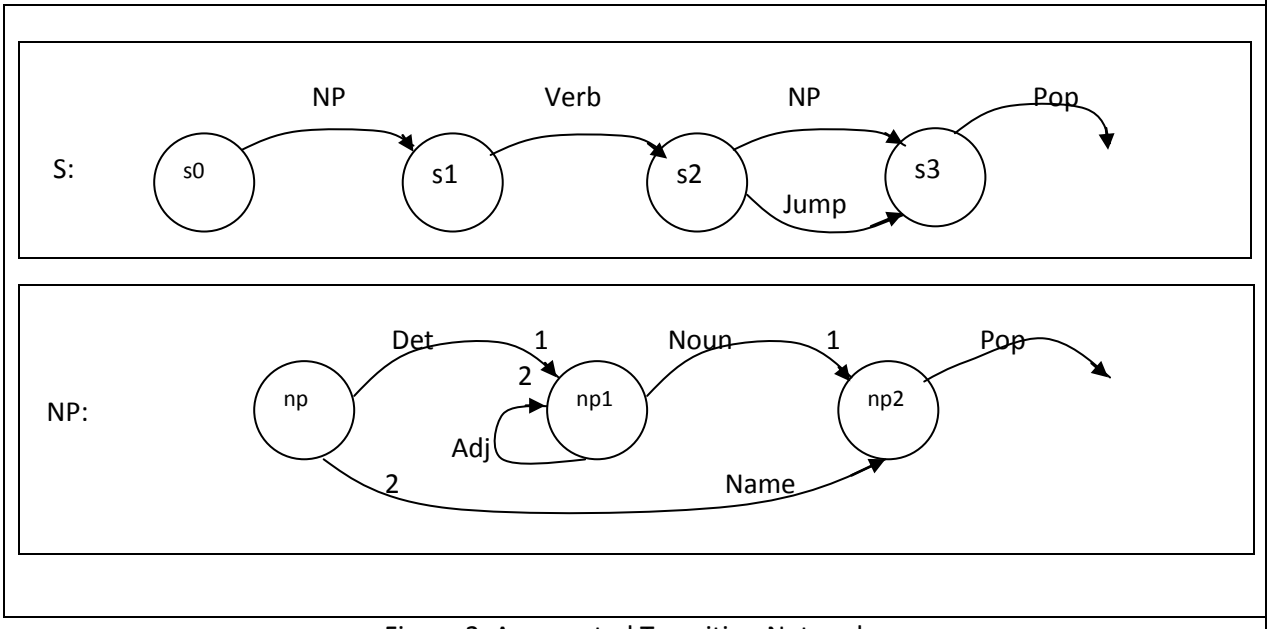


Figure 2. Augmented Transition Network

Following registers and notations are used in the next page:

1. NUM* is the NUM register of the structure in *
2. NUM_{SUBJ} is the NUB register of the structure in SUBJ
3. The value of the registers are often viewed as sets, and the intersection and union of sets are allowed to combine the values of different registers. For the registers that may take a list of values, an append function is permitted
4. Append (ADJ, *) returns the list of adjectives.

	Arc	Test	Actions	Comments
NP	np/1	None	DET ← * NUM ← * NUM*	Current token with its number in DET and NUM registers
	np/2	None	NAME * NUM NUM*	Current token with its number in NAME and NUM registers
	np1/1	NUM ∩ NUM*≠ ϕ {then action is taken else it fails}	HEAD ← * NUM ← NUM ∩ NUM*	Assign current token to HEAD
	np1/2	None	ADJS ← Append (ADJS, *)	Collect adjectives in the list and store it in ADJS
S	s0/1	None	SUBJ ← *	Assign entire structure returned from NP network to SUBJ
	s1/1	NUM_{SUBJ} ∩ NUM* ≠ ϕ {then action is taken else it fails}	MAIN_V ← * NUM ← NUM_{SUBJ} ∩ NUM*	Assign current token to MAIN_V
	s2/1	None	OBJ ← *	Assign entire structure returned from NP network to OBJ

Table 1. Test and Actions of the Arc in ATN.

Step	Node	Position	Arc followed	Registers
1	s0	1	s0/1	-
2	np	1	np/1	DET ← the NUM ← {sing.[plur} HEAD ← dogs NUM ← {plur}
3	np1	2	np/1 {check {sing.plur} ∩ {plur} ≠ ϕ}	
4	np2	3	np2/1	return structure SUBJ ← (NP(DET ← the HEAD ← dogs NUM ← {plur}))
5	- s1	3 3	s0/1 Succeeds s1/1 {check {plur} ∩ {plur} ≠ ϕ}	MAIN_V ← love NUM ← {plur} MAIN_V ← love NUM ← {plur}
6	s2	4	s2/1	OBJ ← *
7	np	4	np/2	NAME ← john NUM ← {plur}
8	np2	5	np/1	return structure OBJ← (NP {NAME ← john NUM ← {plur})
9	s3	5	s3/1 Succeeds	return (S SUB← (NP(DET ← the HEAD ← dogs, NUM ← {plur}) (MAIN_V ← love, OBJ (NP(NAME← john NUM ← {plur})

Table 2. Trace of Parsing Sentence using ATN

Universal Networking Language

Project – Universal Networking Language (UNL) was initiated by Institute of Advanced Studies in 1995 by United Nations University (UNU).
UNL is an electronic language that provides a uniform representation of a natural language. The objective of UNL is generic, language-neutral formalism in systematic and coordinated way. UNL is a controlled language which has predefined vocabulary, building rules in an electronic content mark-up language. It basically provides a platform for language neutral mark-up language. UNL has many applications such as: Machine translation, multilingual information service, Information retrieval system, Search engine, and Expert systems etc.

UNL consists of:

1. **Vocabulary (Universal Words (UW))**
2. **Syntax (Relations & Attributes)**
3. **Semantics (UNL Knowledge Base)**

Example: UNL representation of sentence, "Monkey eats bananas.".

```
[S]
[org]
        Monkey eats a mango
[/org]
[unl]
        [W]
            eat(icl>do).@present.@entry:00
            monkey(icl>animal).@generic:01
            banana(icl>food:).@generic:02
        [/W]
[R]
agt(eat(icl>do).@entry, monkey(icl>animal).@generic)
obj(eat(icl>do).@present.@entry,banana(icl>food).@generic)
[/R]
[/unl]
[S]
```

Note: the relation 'icl' (inclusion) is used to allow properties to be inherited from upper UWs so that a UW can be deductively inferred from existing UW.

Universal Words

UWs are made up of a roman character string followed by a list of constraints. For example, *dog* is represented as *dog(icl>animal,* where *animal* is upper UW. General definition of UW (BNF like grammar) is given below:

Universal Word UW Definition	Meta symbol Interpretation
<UW> ::= <Head Word> [<Constraint List]	< > For non-technical symbol or a variable
<Head Word> ::= <character> . . .	" " for enclosed string is literal characters
<Constraint List> ::= "("<Constraint> ["," <Constraint>]")"	::= for defined as
<Constraint> ::= <Relation Label> {">"} <UW> [<Constraint List] . . .	| for disjunction, "or"
<Relation Label> ::= "agt" | "and" |"adj"|"icl"| . . .	[ ] for optional element
<character> ::= "A"| . . . |"Z"|"a"| . . .|0| . . . |9|"_"|" "|"#"|"!"| "$"|"%"|"="|"^"|"~"|"|"|"@"|"+"|"-"|"<"|">"|"?"	{ } for alternative element . . . three dots for repetition more than one time

Types of UWs:

1. Basic UW: English word with no restrictions.
2. Restricted UW: with constraint list attached. Example, 'state(icl>situation)' is a sense of 'state' that denotes a kind of situation.
3. Extra UW: mot found in English but used. Example, samba(icl>dance) – a kind of dance.

Binary Relation

Definition	Interpretation
<Binary Relation> ::= <Relation Label> [":" <Compound UW-ID>] "("{<UW1> ":"UW-ID1]>} [":" <Compound UW-ID1>]"."{<UW2> ":"<UW-ID2>} [":" <Compound UW-ID2>] ")"	Relation Label is a string of two or three lower-case alphabetic characters taken from the closed list. Compound UW-ID is a string of two digits used to identify each compound UWs. Compound UW is a group of binary relations (called 'Hyper-Nodes').

The UNL System consists of the Language Servers and basic tools. The language server resides in the network. UNL Dictionary stores concepts represented by the language words. The grammar to define words of the language is followed. The knowledge base of the UNL system is continually being expanded.