USN											

Internal Assessment Test II – December 2022

Sub:	Natural Language Processing					Sub Code:	18CS743	Branch:	ISE	
Date:	02/12/22	Duration:	90 mins	Max Marks:	50	Version/ Sem / Sec:	A/VIII/A, B, C			OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	Define Part of Speech tagging and explain different methods of PoS in detail	[10]	CO2	L2

Part of Speech tagging: Process of converting a sentence to forms – list of words, list of tuples (where each tuple is having a form (word, tag)). The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, adverb etc.

POS 3 types:

1. Rule-based POS Tagging
2. Stochastic POS Tagging
3. Hybrid Tagging

1. Rule-based POS Tagging

Rule-based taggers use dictionary or lexicon for getting possible tags for tagging each word. If the word has more than one possible tag, then rule-based taggers 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.

All kind of information in rule-based POS tagging is coded in the form of rules. These rules may be either − Context-pattern rules Or, as Regular expression compiled into finite-state automata, intersected with lexically ambiguous sentence representation.

Rule-based POS taggers possess the following properties −

- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modeling is defined explicitly in rule-based taggers.

2. Stochastic POS Tagging

The model that includes frequency or probability (statistics) can be called stochastic. Any number of different approaches to the problem of part-of-speech tagging can be referred to as stochastic tagger.

The simplest stochastic tagger applies the following approaches for POS tagging −

(i) Word Frequency Approach

In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag. We can also say that the tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word. The main issue with this approach is that it may yield inadmissible sequence of tags.

(ii) **Tag Sequence Probabilities**

	It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring. It is also called n-gram approach. It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags. Stochastic POS taggers possess the following properties − • This POS tagging is based on the probability of tag occurring. It requires training corpus • There would be no probability for the words that do not exist in the corpus. • It uses different testing corpus (other than training corpus). • It is the simplest POS tagging because it chooses most frequent tags associated with a word in training corpus. **3. Hybrid Tagging (Transformation-based Tagging)** It is an instance of the transformation-based learning (TBL), which is a rule-based algorithm for automatic tagging of POS to the given text. TBL, allows us to have linguistic knowledge in a readable form, transforms one state to another state by using transformation rules. It draws the inspiration from both the previous explained taggers − rule-based and stochastic. If we see similarity between rule-based and transformation tagger, then like rule-based, it is also based on the rules that specify what tags need to be assigned to what words. On the other hand, if we see similarity between stochastic and transformation tagger then like stochastic, it is machine learning technique in which rules are automatically induced from data. **Start with the solution** − The TBL usually starts with some solution to the problem and works in cycles. Most beneficial transformation chosen − In each cycle, TBL will choose the most beneficial transformation. **Apply to the problem** − The transformation chosen in the last step will be applied to the problem. The algorithm will stop when the selected transformation in step 2 will not add either more value or there are no more transformations to be selected. Such kind of learning is best suited in classification tasks. The advantages of TBL are as follows − • We learn small set of simple rules and these rules are enough for tagging. • Development as well as debugging is very easy in TBL because the learned rules are easy to understand. • Complexity in tagging is reduced because in TBL there is interlacing of machine learned and human-generated rules. • Transformation-based tagger is much faster than Markov-model tagger. The disadvantages of TBL are as follows − • Transformation-based learning (TBL) does not provide tag probabilities. • In TBL, the training time is very long especially on large corpora.			
2.	Calculate the Min Edit Distance between the given words, **(i) ‘tutor’ and ‘tumour’.** **(ii) ‘paceful’ and ‘peaceful’.** Illustrate the minimum distance algorithm in detail	[10]	CO2	L3

t u **t** o – r

t u **m** o u r

A dash in the upper string indicates insertion. A substitution occurs when the two alignment symbols do not match (shown in bold). We can associate a weight or cost with each operation. The *Levensthein* distance between two sequences is obtained by assigning a unit cost to each operation. Another possible alignment for this sequences is:

t u **t** – o – r

t u – **m** o u r

which has a cost of 3. We already have a better alignment than this one.

The problem of finding minimum edit distance seems quite simple but in fact is not so. A choice that seems good initially might lead to problems later. Dynamic programming algorithms can be quite useful for finding minimum edit distance between two sequences. Dynamic programming refers to a class of algorithms that apply a table-driven approach to solve problems by combining solutions to sub-problems. The dynamic programming algorithm for minimum edit distance is implemented by creating an edit distance matrix. This matrix has one row for each symbol in the source string and one column for each matrix in the target string. The (i, j)th cell in this matrix represents the distance between the first i character of the source and the first j character of the target string. Each cell can be computed as a simple function of its surrounding cells. Thus, by starting at the beginning of the matrix, it is possible to fill each entry iteratively. The value in each cell is computed in terms of three possible paths.

$$dist[i, j] = \begin{cases} dist[i-1, j] + insert\_cost, \\ dist[i-1, j-1] + subst\_cost[source_i, target_j] \\ dist[i, j-1] + delete\_cost \end{cases}$$

```
Input: Two strings, X and Y
Output: The minimum edit distance between X and Y
m <- length(X)
n <- length(Y)
for i = 0 to m do
   dist[i,0] <- i
for j = 0 to n do
   dist[0,j] <- j
for i = 0 to m do
   for j = 0 to n do
      dist[i,j] = min( dist[i-1,j] + insert_cost,
                       dist[i-1,j-1] + subst_cost(Xi,Yj),
                       dist[i,j-1] + delet_cost }
```

Figure 3.13 Minimum edit distance algorithm

	#	t	u	m	o	u	r
#	0	1	2	3	4	5	6
t	1	0	1	2	3	4	5
u	2	1	0	1	2	3	4
t	3	2	1	1	2	3	4
o	4	3	2	2	1	2	3
r	5	4	3	3	2	2	2

Figure 3.14 Computing minimum edit distance

(ii)‘paceful’ and ‘peaceful’.

	#	P	E	A	C	E	F	U	L
#	0	1	2	3	4	5	6	7	8
P	1	0	1	2	3	4	5	6	7
A	2	1	1	1	2	3	4	5	6
C	3	2	2	1	1	2	3	4	5
E	4	3	3	2	1	1	2	3	4
F	5	4	4	3	2	1	1	2	3
U	6	5	5	4	3	2	1	1	1
L	7	6	6	5	4	3	2	1	1

3. Explain Dependency Path Kernel for relation extraction [10] CO3 L2

The pattern examples show the two entity mentions, together with the set of words that are relevant for their relationship. A closer analysis of these examples reveals that all relevant words form a shortest path between the two entities in a graph structure where edges correspond to relations between a word (head) and its dependents. For example, Figure 3.4 shows the full dependency graphs for two sentences from the ACE (Automated Content Extraction) newspaper corpus, in which words are represented as nodes and word-word dependencies are represented as directed edges. A subset of these word-word dependencies captures the predicate-argument relations present in the sentence. Arguments are connected to their target predicates either directly through an arc pointing to the predicate (‘troops → raided’), or indirectly through a preposition or infinitive particle (‘warning ← to ← stop’). Other types of word-word dependencies account for modifier-head relationships present in adjective-noun compounds (‘several → stations’), noun-noun compounds (‘pumping → stations’), or adverb-verb constructions (‘recently → raided’). Word-word dependencies are typically categorized in two classes as follows:

- [Local Dependencies] These correspond to local predicate-argument (or head modifier) constructions such as ‘troops → raided’, or ‘pumping → stations’ in Figure 3.4.
- [Non-local Dependencies] Long-distance dependencies arise due to various linguistic constructions such as coordination, extraction, raising and control. In Figure 3.4, among non-local dependencies are ‘troops → warning’, or ‘ministers → preaching’.

A Context Free Grammar (CFG) parser can be used to extract local dependencies, which for each sentence form a dependency tree. Mildly context sensitive formalisms such as Combinatory Categorial Grammar (CCG) model word-word dependencies more directly and can be used to extract both local and long-distance dependencies, giving rise to a directed acyclic graph,

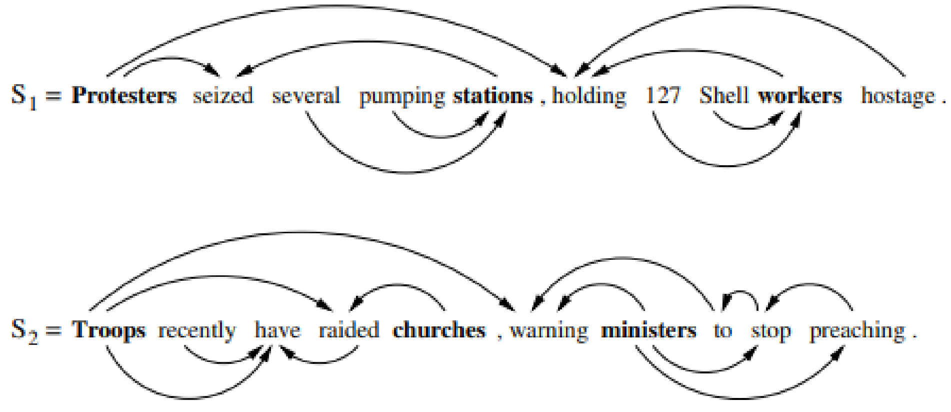
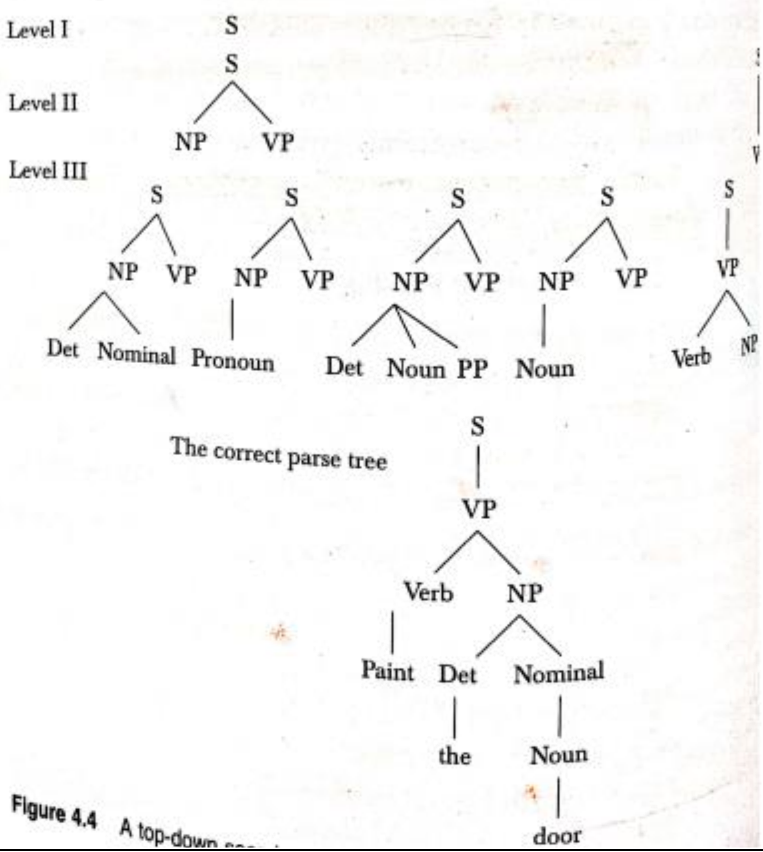


Fig. 3.4. Sentences as dependency graphs.

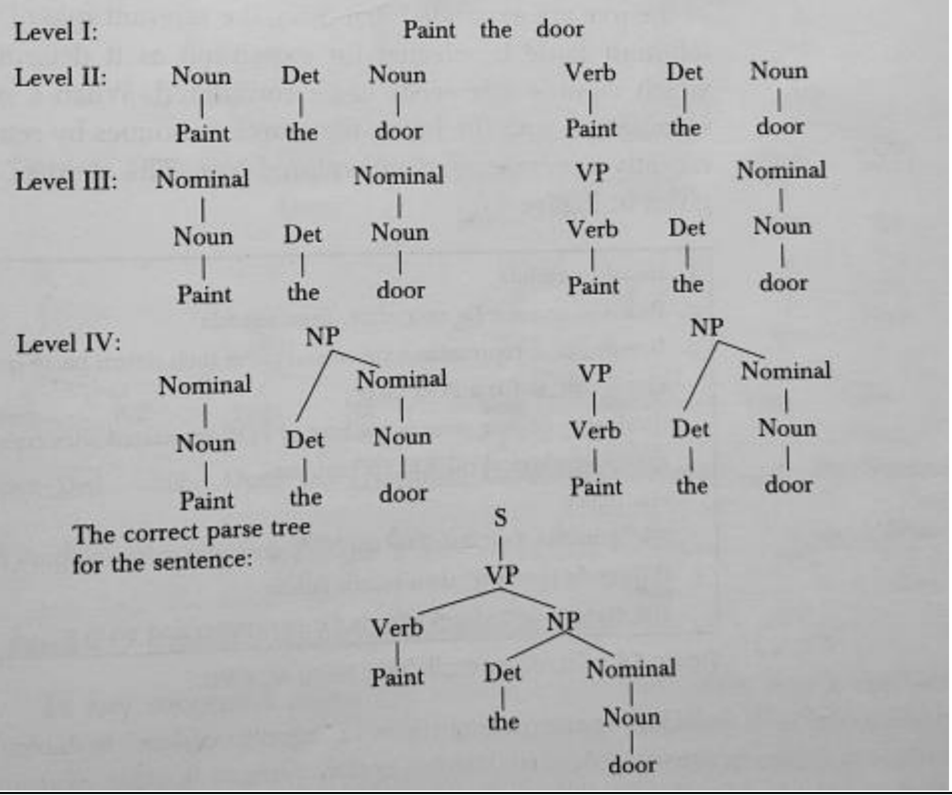
4	Illustrate various shortest dependency path among the relations in the **“Jellisca created an atmosphere of terror in the camp by killing abusing and threatening the detainees”.** If e1 and e2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, then the contribution of the sentence dependency graph to establishing the relationship R(e1, e2) is almost exclusively concentrated in the shortest path between e1 and e2 in the undirected version of the dependency graph. If entities e1 and e2 are arguments of the same predicate, then the shortest path between them will pass through the predicate, which may be connected directly to the two entities, or indirectly through prepositions. If e1 and e2 belong to different predicate-argument structures that share a common argument, then the shortest path will pass through this argument. There may be cases where e1 and e2 belong to predicate-argument structures that have no argument in common. However, because the dependency graph is always connected, we are guaranteed to find a shortest path between the two entities. In general, we shall find a shortest sequence of predicate-argument structures with target predicates P1, P2, ..., Pn such that e1 is an argument of P1, e2 is an argument of Pn, and any two consecutive predicates Pi and Pi+1 share a common argument (whereby “argument” we mean both arguments and complements) Jelisic created an atmosphere of terror at the camp by killing, abusing and threatening the detainees. detainees → killing ← Jelisic → created ← at ← camp detainees → abusing ← Jelisic → created ← at ← camp detainees → threatning ← Jelisic → created ← at ← camp detainees → killing → by → created ← at ← camp detainees → abusing → by → created ← at ← camp detainees → threatening → by → created ← at ← cam	[10]	CO3	L3
5	**Frame Semantics**: In frame semantics theory, a frame is a “script-like conceptual structure that describes a particular type of situation, object, or event and the participants involved in it. Based on this theory, the Berkeley FrameNet Project1 is creating an online lexical resource for the English language by annotating text from the 100 million words British National Corpus. The structure of a frame contains lexical units (pairs of a word with its meaning), frame elements (semantic roles played by different syntactic dependents), as well as annotated sentences for all lexical units that evoke the frame. Annotation of text with frames and roles in FrameNet has been performed manually by trained linguists. An effort to handle this task automatically is being carried out by research in semantic role labeling, as described in the next subsection. **Semantic Role Labeling**: After acknowledging the success of information extraction systems that try to fill in domain specific frame-and-slot templates, the need for semantic frames that can capture the meaning of text independently of the domain was expressed. The semantic interpretation of text in terms of frames and roles would contribute to many applications, like question answering, information extraction, semantic dialogue systems, as well as statistical machine translation or automatic text summarization, and finally also to text mining.			

	Frame *Evidence* Definition: The Support, a phenomenon or fact, lends support to a claim or proposed course of action, the Proposition, where the Domain_of_Relevance may also be expressed. Lexical units: *argue.v, argument.n, attest.v, confirm.v, contradict.v, corroborate.v, demonstrate.v, disprove.v, evidence.n, evidence.v, evince.v, from.prep, imply.v, indicate.v, mean.v, prove.v, reveal.v, show.v, substantiate.v, suggest.v, testify.v, verify.v* Frame Elements: Proposition [PRP] — This is a belief, claim, or proposed course of action to which the Support lends validity. Support [SUP] — Support is a fact that lends epistemic support to a claim, or that provides a reason for a course of action. ... Examples: And a [$_{SUP}$ sample tested] **REVEALED** [$_{PRP}$ some inflammation]. It says that [$_{SUP}$ rotation of partners] does not **DEMONSTRATE** [$_{PRP}$ independence]. Information on the frame Evidence from FrameNet.			
6	Analyze the top-down and bottom-up search space for the sentence 'paint the door' by applying following grammar and list out the advantages and disadvantages of Top-Down and Bottom-Up parsing S → NP VP S → VP NP → Det Nominal NP → Noun NP → Det Noun PP Nominal → Noun Nominal → Noun Nominal VP → Verb NP VP → Verb PP → Preposition NP Det → this | that | a | the Verb → sleeps | paint | open | sings Preposition → from | with | on | to Pronoun → She | he | they **Paint the door.** A top-down search begins with the start symbol of the gramma the first level (ply) search tree consists of a single node labelled grammar in Table 4.2 has two rules with S on their left hand side  **Figure 4.4** A top-down ...			

rules are used to expand the tree, which gives us two partial trees at the second level search, as shown in Figure 4.4. The third level is generated by expanding the non-terminal at the bottom of the search tree in the previous ply. Due to space constraints, only the expansion corresponding to the left-most non-terminals has been shown in the figure. The subsequent steps in the parse are left, as an exercise, to the readers. The correct parse tree shown in Figure 4.4 is obtained by expanding the fifth parse tree of the third level.

4.4.2 Bottom-up Parsing

A bottom-up parser starts with the words in the input sentence and attempts to construct a parse tree in an upward direction towards the root. At each step, the parser looks for rules in the grammar where the right hand side matches some of the portions in the parse tree constructed so far, and reduces it using the left hand side of the production. The parse is considered successful if the parser reduces the tree to the start symbol of the grammar. Figure 4.5 shows some steps carried out by the bottom-up parser for sentence (4.7).



CI CCI HOD