Internal Assessment 1 solution

Sub:	Natural Language Processing					Sub Code:	15CS741	Branch:
Date:	20.09.18	Duration:	90 min's	Max Marks:	50	Sem / Sec:	VII/C	

Answer any FIVE FULL Questions

MARKS

1 (a) What is NLP? Explain the challenges and applications of NLP. [04]

⇒ NLP is concerned with development of computational models of aspects of human lang. processing. 2 main reasons for development:

i) Develop automated tools for language processing.

ii) Gain a better understanding of human comm.

Challenges of NLP:

→ No. of factors that make NLP difficult.

→ Relate to problems of representation + interpretation

→ Lang. comming ⇒ precise representation of context.

NL. ⇒ highly ambiguous, vague ⇒ achieving such representation can be difficult.

→ Inability to capture all the required knowledge.

impossible to embody all sources of knowledge that humans use.

not possible to write procedure that imitate work done by humans.

⇒ greatest difficulty → identifying its semantics.

⇒ Principle of compositional semantics

conside the meaning of a sentence to be a composition of the meaning of words.

⇒ words alone donot make a sentence.

eg: I do not like icecream

do not I icecream like. ×

(or)

Kabir and Ayan are married } both give different

Kabir + Suha " " } meaning even if

has same structure.

⇒ Only words along with their syntactic + semantic relation give meaning to a sentence.

→ Inability to capture all the required knowledge.

impossible to embody all sources of knowledge that humans use.

not possible to write procedure that imitate work done by humans.

⇒ greatest difficulty → identifying its semantics.

⇒ Principle of compositional semantics

conside the meaning of a sentence to be a composition of the meaning of words.

⇒ words alone donot make a sentence.

eg: I do not like icecream

do not I icecream like. ×

(or)

Kabir and Ayan are married } both give different

Kabir + Suha " " } meaning even if

has same structure.

⇒ Only words along with their syntactic + semantic relation give meaning to a sentence.

=> Idioms, Metaphor + ellipses => more complexity to identify the meaning of the written text.
eg: oldman finally kicked the bucket.
(2) Quantifier Scoping – scope of quantifiers is not clear.
(3) Ambiguity of natural languages.
(4) Incorporating Contextual + world langu Knowledge poses the greatest difficulty.

NLP applications:
1.MachineTranslation 2.Speech Recognition 3.Speech synthesis 4.Natural language interfaces to databases 5.Information retrieval 6.Information extraction 7.Question Answering.

(b) Explain the stages of transformational grammar for the sentence "The boy hit the girl". [06]

=> Transformational grammar has 3 components
(1) phrase structure
(2) Transformational rules.
(3) Morphophonemic rules -> match each sentence represent to a string of phonemes.

S → NP + VP
VP → V + NP
NP → Det + Noun.
V → Aux + Verb.
Det → the, a, an ...
Verb → Catch, write, eat ...
Noun → police, Snatcher
Aux → will, is, can ...

S => Sentence
NP => Noun phrase
VP => Verb phrase
Det -> Determiner

2 (a) Explain various grammar based language models. [03]

1.Generative grammar 2.Hierarchical grammar 3.Government and binding 4.Lexical Functional grammar 5.Paninian framework

(b) Explain the types of statistical language model with an example. [07]

Statistical Language model

- A statistical language model is a probability distribution P(s) over all possible words sequences.

i) n-gram model:
- The goal is to estimate the probability of sentence.
- By decomposing sentence probability into a product of conditional probabilities using the chain rule as follows:

$$P(S) = P(w_1, w_2, w_3 \cdots, w_n)$$
$$= P(w_1)\, P(w_2/w_1)\, P(w_3/w_1 w_2)\, P(w_4/w_1 w_2 w_3) \cdots P(w_n/w_1 w_2 w_3 \cdots w_{n-1})$$
$$= \prod_{i=1}^{n} P(w_i/h^i)$$

h^i is history of word w_i defined as $w_1 w_2 \cdots w_{i-1}$

An n-gram model simplifies the task by approximating the probability of a word gn. all the previous words by the conditional probability gn previous n-1 words only.

$$P(w_i/h^i) \approx P(w_i/w_{i-n+1} \cdots w_{i-1})$$

Thus, an ngram model calculate $P(w_i/h^i)$ by modelling language as Markov model of order n-1 ie) by looking at previous n-1 words only.

A model that limits the history to the previous one word only, is termed a bi-gram (n=1) model.

Using bi-gram and tri-gram estimate, the probability of sentence can be calculated as,

$$P(s) \approx \prod_{i=1}^{n} P(w_i | w_{i-1}) \text{ and}$$

$$P(s) \approx \prod_{i=1}^{n} P(w_i | w_{i-2} \cdot w_{i-1})$$

eg: The bigram approximation of P(east / The Arabian knights are fairy tales of the) is

P(east / the).

Whereas a trigram approximation is

P(east / of the)

Count a particular n-gram in the training corpus & divide it by the sum of all n-grams that share the same prefix.

$$P(w_i | w_{i-n+1} \cdots w_{i-1}) = \frac{C(w_{i-n+1}, \cdots w_{i-1}, w_i)}{\sum_{w} C(w_{i-n+1}, \cdots w_{i-1}, w)}$$

The sum of all n-grams that share first n-1 words is equal to the count of the common prefix $w_{i-n+1} \cdots w_{i-1}$)

$$P(w_i | w_{i-n+1} \cdots w_{i-1}) = \frac{C(w_{i-n+1}, \cdots, w_{i-1}, w_i)}{C(w_{i-n+1}, \cdots w_{i-1})}$$

The model parameta we get using these estimates, maximizes the probability of the training set T gn the model M. ie) P(T/M).

eg:

training set :-

The Arabian Knights
These are the fairy tales of the east
The stories of the arabian knights are translated in many languages.

Bigram model:

P(the / <s>) = 0.67 P(Arabian / the) = 0.4
P(knights / Arabian) = 1.0

P(are / there) = 1.0 P(the / are) = 0.5
P(fairy / the) = 0.2

P(tales / fairy) = 1.0 P(of / tales) = 1.0
P(the / of) = 1.0

P(east / the) = 0.2 P(stories / the) = 0.2
P(of / stories) = 1.0

P(are / knights) = 1.0 P(translated / are) = 0.5
P(in / translated) = 1.0

P(many / in) = 1.0

P(languages / many) = 1.0

Test Sentence (S): The Arabian Knights are the fairy tales of the east.

P(The | <s>) x P(Arabian | the) x P(Knights | Arabian) x P(are | Knights) x P(the | are) x P(fairy | the) x P(tales | fairy) x P(of | tales) x P(the | of) x P(east | the)

= 0.67 x 0.5 x 1.0 x 1.0 x 0.5 x 0.2 x 1.0 x 1.0 x 1.0 x 0.2

= 0.0067

Add8-one Smoothing

It adds a value of one to each n-gram frequency before normalizing them into probabilities. Thus the conditional probability becomes:

$$P(w_i | w_{i-n+1} \ldots w_{i-1}) = \frac{c(w_{i-n+1} \ldots w_{i-1}, w_i)}{c(w_{i-n+1}, \ldots w_{i-1}) + V}$$

V is the vocabulary size (or) size of the set of all the words being considered.

Good-Turing Smoothing

- adjusts the frequency f of an n-gram using the count of n-grams having a frequency of occurence f+1

$$f^* = (f+1)\frac{n_{f+1}}{n_f}$$

n_f - no. of n-gram that occur exactly f times in the training corpus.

eg: occur 4 times is 25108 & the no. of n-gram that occur 5 times is 20542, smoothed count for 5 will be

$$\frac{20,542}{25108} \times 5 = 4.09$$

Caching Technique

- The frequency of n-gram is not uniform across the text segments or corpus.

- Certain words occur more frequently in certain segments and rarely in others eg: the frequency of the word 'n-gram' is high, whereas it occurs rarely

Caching Technique

- The frequency of n-gram is not uniform across the text segments or corpus.

- Certain words occur more frequently in certain segments and rarely in others eg: the frequency of the word 'n-gram' is high, whereas it occurs rarely

The basic n-gram model ignores this sort of variation of n-gram frequency.

The cache model combines the most recent n-gram frequency with the std n-gram model to improve its performance locally.

3. Explain briefly about Government and binding grammar. [10]

iv) Government and binding

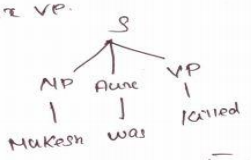
- Chomsky - 1981
- The structure of a language is understood at the level of its meaning. The sentences are gn. at the syntactical level and the transformation from meaning to syntax or vice versa is not well understood.
- Transformational grammar has two level of existence of sentences
  - (i) Surface level
  - (ii) Deep root level
- Government and binding theories renamed as:
  - (i) S-level
  - (ii) d-level and identified 2 more levels.
  - iii) phonetic form
  - iv) Logical form

d-Structure
|
S-Structure
|
phonetic form logical form

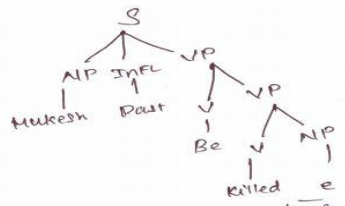
Different levels of representation in GB.

Ex: Mukesh was killed

i) In transformational grammar, this can be represented as S-NP Aux VP.

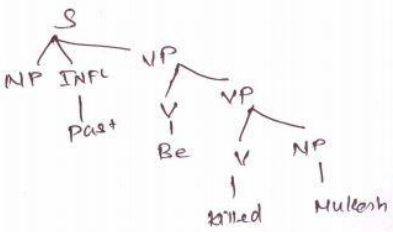


ii) In GB, the S-structure and d-structure:



Infl: Inflection
NP: Noun phrase
VP: Verb phrase
e: empty.

Surface structure



Deep Structure

Components of GB:

- Comprises a set of theories that map the structures from d-structure to S-structure and to logical form.
- Transformational rule: 'Move α' is applied to d-structure level and S-structure level.

d-structure <----- Theories and Principles
| Move α
S-structure <----- Theories and Principles
| Move α
Logical form <----- Theories and Principles

Components of GB

GB ⇒ a series of modules that contain constraints and principles applied at various levels of syn representation and transformation.

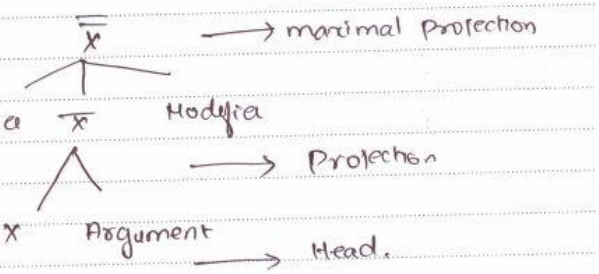
General characteristics of GB:

* X-bar theory
* Projection principle
* θ-theory
* θ - Criterion
* C-Command and government
* Case study or theory
* empty category principle (ECP)
* Binding theory.

X Theory:

- Central concept in GB
- X theory defines both phrase structure and the sentence structure with separate set of rules as maximal projections.
- Entities defined become language independent Thus, noun phrase, verb phrase, adjective phrase, prepositional phrase are maximal projections of nouns, verb, adjective and preposition.

$X = \{N, V, A, P\}$



Sub Categorization:

It is to be noted that GB does not consider traditional phrase structure as an appropriate device for defining language constructs.

It places the burden of ascertaining well formedness to subcategorization frames of heads.

Any maximal projection can be argument of a head, but sub-categorization is used as a filter to permit various heads to select a certain subset of the range of maximal projection.

eg: we know that the verb 'eat' and subcategorize for NP, whereas the verb 'sleep' cannot. Hence 'ate food' is well formed but the sentence 'slept the bed' not.

Projection principle

- places a constraint on the three syntactic representations and their mapping from one to the other.
- The principle states that representations at all syntactic levels (ie) d-level, S-level, LF-level) are projections from the lexicon.

Theta Theory (θ-theory) or The theory of thematic relations:

Subcategorization only places a restriction on syntactic categories which a head can accept. GB

Put another restriction on the lexical heads through which it assign certain roles to its arguments. These roles are preassigned and cannot be violated at any syntactical level as per the Projection Principle. These role assignments are called theta roles and are related to 'Semantic-Selection'.

Theta-role and theta-Criterion:

There are certain thematic roles from which a head can select. These are called θ-roles and they are mentioned in the lexicon, eg: the Verb 'read' can take arguments with θ-roles. (Agent-theme)

C-Command and government:

As 'Government' is a Special case of 'C-Command' we will first define C-command.

C-command: defines the Scope of maximal projection. It is a basic mechanism through which many constraints are defined on Move α.

If any word or phrase falls within the Scope and is determined by a maximal projection. Now if there are two structures α and β related in such a way that 'every maximal projection dominating α dominates β'.

Government:

α governs β iff
α C-Commands β

α is an X (head, eg: Noun, Verb, Preposition, adjective, + Inflection), and every maximal projection dominating β dominates α.

Movement, Empty Category and co-indexing

In GB, Move α is described as 'move anything anywhere', though it provides restrictions for valid movements.

Binding theory:

Binding is defined by Sells (1985)
α binds β iff
α C-Commands β, and
α and β are co-indexed

[e_i INFL Kill Mukesh]
[Mukesh_i was killed (by e_i)]
Mukesh was killed.

Empty clause (e_i) and Mukesh (NP_i) are bound. This theory gives a relationship b/w NPs
Now, binding theory can be given as follows:
a) An anaphor (+a) is bound in its governing category.
b) A pronominal (+P) is free in its governing category.
c) An R-expression (-a, -P) is free.

Empty Category Principle (ECP)

'Proper government':

α Properly governs β iff
α governs β and α is lexical (i.e. N, V, A or P) or
α locally A binds β.

The ECP says 'A trace must be Properly governed'.

This principle justifies the creation of empty categories during NP-trace and Wh-trace and also explains the subject/object asymmetries to some extent.

a) What_i do you think that Mukesh ate e_i?
b) What_i do you think mukesh ate e_i?

Bounding and control Theory:

There are many other types of constraints on moved. The long distance movement for complement clause can be explained by bounding theory if NP and S are taken to be bounding nodes. The theory says that the application of move α may not cross more than one bounding node. The theory of control involves syntax, semantics and pragmatics.

Case Theory and Case filter:

In GB, case theory deals with the distribution of NPs and mentions that each NP must be assigned a case.

4(a) Explain Lexical functional grammar (LFG) model for the sentence "She saw stars in the sky". [06]

Lexical functional Grammar model

LFG represents sentences at two syntactic levels - constituent structure (C-structure) and functional structure (f-structure).

Based on Wood's Augmented Transition Networks 1970, which used phrase structure trees to represent the surface structure of sentences and the underlying predicate - argument structure

C-Structure and f-structure in LFG:

An LFG is aimed at providing exact computational algorithms. It provides well defined objects called constituent structure (C-structure) and functional structure (f-structure).

The C-structure is derived from the usual phrase and sentence structure syntax, as in CFG. However the grammatical - functional role cannot be derived directly from phrase and sentence structure, functional specifications are annotated on the nodes of c structure, which applied on sentences, results in f-structure, which is the final product

which encodes the information obtained from phrase + sentence structure rules and functional specifications.

eg: She saw stars in the sky

CFG rules to handle this sentence are:

S → NP VP

VP → V {NP} {NP} PP* {S'}

PP → P NP

NP → Det N {PP}

S' → Comp S

where

S: Sentence V: verb P: Preposition N: noun

S': clause Comp: Complement {} optional

*: phrase can appear any number of times including blanks.

When annotated with functional specifications, the rules become:

Rule 1: S → NP VP

↑ subj = ↓ ↑ = ↓

Rule 2: VP → V {NP} {NP} PP* {S'}

↑ obj = ↓ ↑ obj 2 = ↓ ↑ (↓ case) = ↓ ↑ comp = ↓

Rule 3: PP → P NP

↑ obj = ↓

Rule 4: NP → {Det} N {PP}

↑ Adjunct = ↓

Rule 5: S' → comp S

↑ = ↓

↑ – f-structure of the mother node that is on the left hand of the rule.

↓ – f-structure of the node under which it is denoted.

Hence in Rule 1, (↑ subj = ↓) indicates that the f-structure of the first NP goes to the f-structure of the subject of the sentence.

while (↑ = ↓) indicates that the f-structure of the VP node goes directly to the f-structure of the sentence VP.

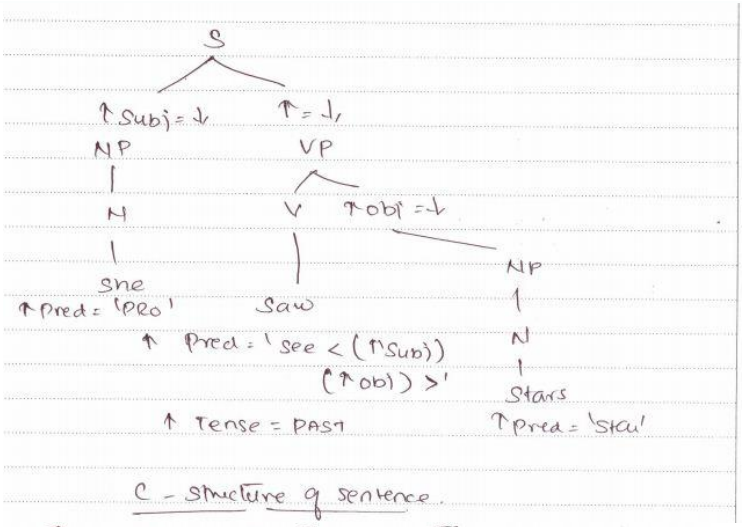
Lastly in Rule 2, the f-structure of VP is defined by the lexical item V, the two optional NPs any no. of PP's and the optional clause (S').

She saw stars.

She N (↑ pred) = 'PRO'
(↑ pers) = 3
(↑ Num) = SG
(↑ Gen) = FEM
(↑ case) = NOM

saw V (↑ Pred = 'see < (↑ Subj) (↑ obj) >'
(↑ Tense = PAST)

Stars N ↑ Pred = 'Star'
↑ Pers = 3
↑ Num = PL



C - Structure of sentence.

$$\begin{bmatrix} \text{Subj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{SG} \\ \text{Gen} & \text{FEM} \\ \text{case} & \text{NOM} \\ \text{Pred} & \text{'PRO'} \end{bmatrix} \\ \text{obj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{PL} \\ \text{Pred} & \text{'Star'} \end{bmatrix} \\ \text{Pred} & \text{'see'}\langle(\uparrow\text{Subj})(\uparrow\text{Obj})\rangle \end{bmatrix}$$

LFG imposes three conditions on f-structure:

Consistency: In a gn f-structure, a particular attribute may have at most one value. Hence, while unifying two f-structures if the attribute Num has value SG in one and PL in the other, it will be rejected.

LFG imposes three conditions on f-structure:

Consistency: In a gn f-structure, a particular attribute may have at most one value. Hence, while unifying two f-structures if the attribute Num has value SG in one and PL in the other, it will be rejected.

Completeness: A function is called governable if it appears within the Pred value of some lexical form. eg: Subj, Obj + Obj 2. Adjunct is not a governable function.

Coherence: coherence maps the completeness property in the reverse direction. It requires that all governable functions of an f-structure, and all its subsidiary f-structures must be governed by their respective Predicates.

(b) Explain in detail about Paninian framework and its layered representation. [04]

Paninian Framework

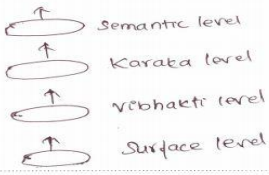
Paninian Grammar-based model was written by Panini in 500BC in Sanskrit, the framework can be used for other Indian languages and possibly some Asian languages.

Unlike English, Asian languages are SOV (subject-object-verb) ordered and inflectionally rich. The inflections provide important syntactic and semantic cues for language analysis.

Layered representation in PG

The GB theory represent three syntactic levels: deep structure, surface structure and logical form, where the LF is nearer to semantics.

Unlike GB, Panian Grammar framework is said to be syntactico-semantic that is one can go from surface layer to deep semantics by passing through immediate layers.



Karaka Theory:

- Central theme of PG framework
- Karaka relations are assigned based on the roles played by various participants in the main activity.
- These roles are reflected in the case markers + post position markers.

Issues in Paninian grammar:

Two problems:

i) Computational implementation of PG and
ii) Adaptation of PG to Indian + other similar languages.

5. Explain the stages of morphological parsing using FST. [10]

Morphological Parsing:

- Morphology is a subdiscipline of linguistics.
- Studies word structure and the formation of words from smaller units (morphemes)
- goal to discover the morphemes that build a given word.
- morphemes => smallest meaning-bearing units in a language.

- morphemes ┬ Stems
  └ affixes

→ Stem - main morpheme ie) The morpheme that contain the central meaning.

→ Affixes - modify the meaning given by the stem

Affixes ÷ i) Prefix
ii) Suffix
iii) Infix
iv) Circumfix

3 main ways of word formation:
- inflection
- derivation
- compounding.

Inflection → a root word is combined with a grammatical morpheme to yield a word of the same class as the original stem.

Derivation → combines a word stem with a grammatical morpheme to yield a word belonging to a different class.
eg: formation of the noun computation from the verb compute.

⇒ the formation of a noun from a verb or adjective is called nominalization.

⇒ Compounding is the process of merging two or more words to form a new word.
eg: personal computer, desktop, overlook.

- A morphological parser uses the following information sources:

i) Lexicon:
- lexicon lists stems and affixes together with basic information about them.

ii) Morphotactics:
- there exists certain ordering among the morphemes that form a word.
- can't be arranged arbitrarily.

eg: restlessness is a valid word
rest-ness-less is invalid.
- This morphotactics deals with the ordering of morphemes, it describes the way morphemes are arranged or touch each other.

iii) Orthographic:
- spelling rules that specify the changes that occur when two given morphemes combine.

eg: y → ier spelling rule changes
easy → easier not to easyer.

- morphological analysis can be avoided if an exhaustive lexicon is available that lists features for all the word-forms of all the roots.

word form	category	Root	Gender	Number	Person
Ghodnaa	noun	Ghodaa	masculin	Singular	3rd
Ghodhii	do	do	feminine	do	do
Ghodhon	do	do	masculine	Plural	do
Ghoelhe	do	do	do	do	do

But this approach has several limitations:

① It puts a heavy demand on memory.

- have to list every form of the word which results in a large number of rendundant entries in the lexicon.

② Exhaustive lexicon fails to show the relationship b/w different roots having similar word forms.

- approach fails to capture linguistic generalization, which is essential to develop a sim capable of understanding unknown words.

③ Morphologically complex languages like Turkish, the no. of possible word forms may be theoretically infinite.

- Not possible to list all possible word forms in these languages.

- Stemming algorithms work in two steps:

i) Suffix removal : this step removes predefined endings from words.

ii) Recoding : this step adds predefined endings to the output of the first step.

- These two steps can be performed sequentially as in Lovin's stemmer or in Porter's stemmer.

eg: porter's stemmer makes use of the following transformation rule:

ational → ate

to transform word such as 'rotational' into 'rotate'.

- In this model, a word is represented in

i) lexical level form → represents the concatenation of its constituent morphemes.

ii) Surface level form. ↓ represents the actual spelling of the word

→ morphological parsing is a mapping from the surface level into morpheme and feature sequences on the lexical level.

- surface form

'playing' → play + V + PP

- lexical form

⇒ Stem 'play' followed by the morphological information +V, +PP

Surface level : playing

Lexical level : play +VPP

⇒ Surface form "books"

Lexical form book + N + PL

↓

morphological information.

↓

Surface level form is plural noun.

- This model is usually implemented with a kind of finite state automata ⇒ finite state transducer (FST)

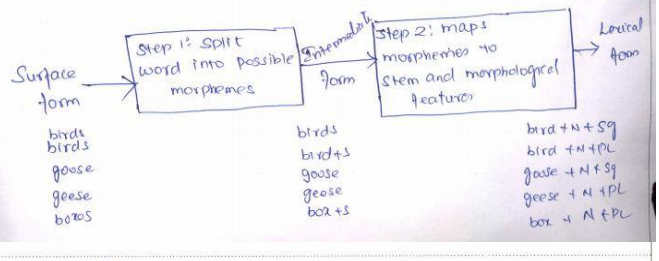
- Transducer maps a set of symbol to another.

- FST - 2 state automaton
  - recognizes or generates a pair of strings.
  - Passes over the input string by consuming the input symbols on the tape it traverses and consists it to the output string in the form of symbols.

- Finite state transducer is a 6 tuple
  $(\Sigma_1, \Sigma_2, Q, \delta, S, F)$

Q - is a set of states
S - is the initial state
$F \subseteq Q$ - set of final states
Σ_1 - input alphabet
Σ_2 - Output alphabet
δ - function mapping

Two Step morphological parser



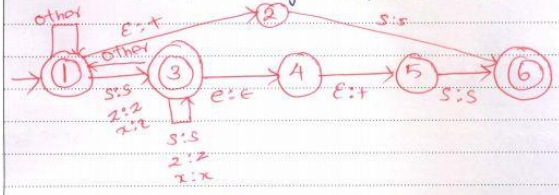
FST

⇒ FST represents the information that the comparative form of adjective less is lesser, 'ε' here is empty string.

⇒ The automaton is inherently bi-directional. the same transducer can be used for analysis ⇒ surface i/p upward for generation lexical i/p, downward appln.

→ The plural form of regular nouns usually end with -s or -es
However a word ending in 's' need not be plural there are number of singular words ending in 's' like miss, ass.

→ One of the required translations is the deletion of the 'e' when introducing a morpheme boundary. This deletion is usually required for words ending in 'xes, ses, zes' eg: (Suppixes, boxes)

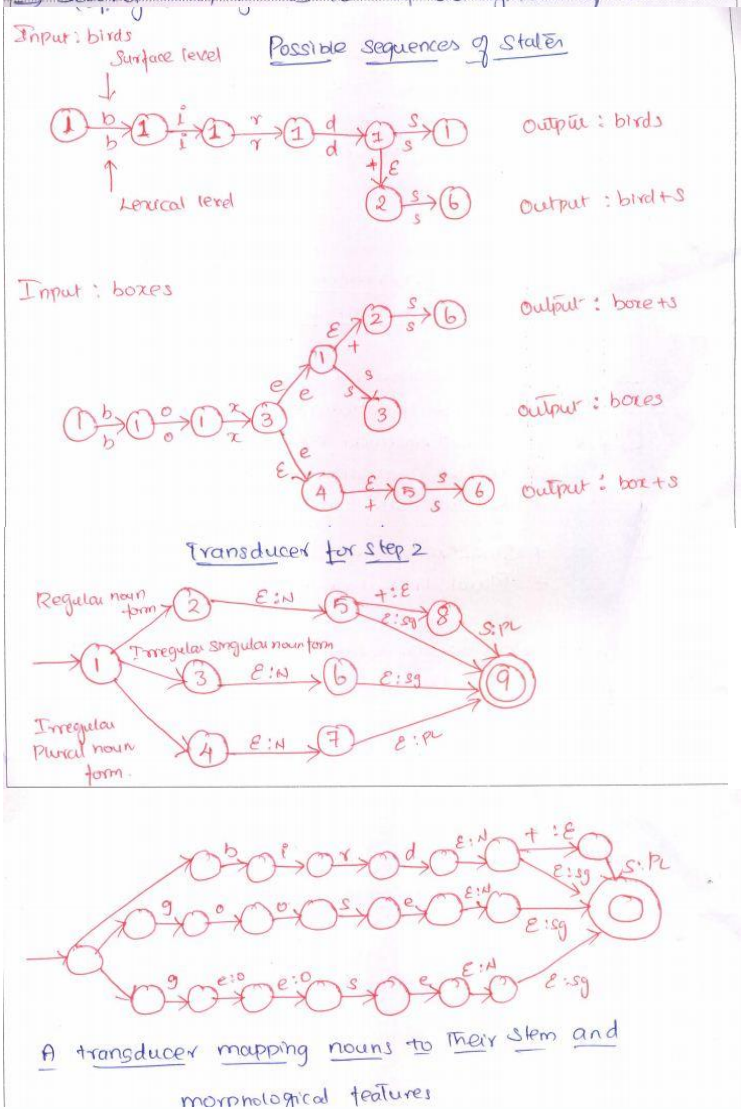


Simplified FST, mapping english nouns to the intermediate form.

→ Next step is to develop a transducer, that does the mapping from the intermediate level to the lexical level.
→ The input to transducer has one of the following forms:
  * Regular noun stem, eg: bird, cat
  * Regular noun stem + s, eg: bird + s
  * Singular irregular noun stem, eg: goose
  * Plural irregular noun stem, eg: geese

→ First case, the transducer has to map all symbols of the stem to themselves and ∅ then output N and sg.

→ Second case, it has to map all symbols of the stem



6. Describe the concept of spelling error detection and correction by computing minimum edit distance algorithm. [10]

Spelling Error Detection & Correction:

- In computer-based information sln, errors of typing and spelling causes variation b/w strings.
- These errors are investigated and that are:
  - * single character omission
  - * Insertion
  - * Substitution
  - * Reversal ⇒ most common typing mistake.
- Damerau (1964) reported that over 80% of the typing errors were single-error misspellings:
  - i) Substitution of a single letter
  - ii) Omission of a " "
  - iii) Insertion of " " "
  - iv) Transposition of two adjacent letters.

- Single character Omission occurs when a single character is missed (deleted). eg: when concept is accidentally typed as 'concpt'.

- Insertion error refers to the presence of an extra character in a word. eg: when 'error' is misspell as 'erron'.

- Substitution error occurs when a wrong letter is typed in place of the right one, eg: 'errpr' where 'p' appears in place of 'o'.

- Reversal refers to a situation in which the sequence of characters is reversed. eg: 'aer' instead of 'are', also called as transposition.

⇒ Optical character Recognition (OCR) and other automatic reading device introduce errors of substitution, deletion, and insertion but not of reversal.

⇒ Spelling errors are mainly due to phonetic and [illegible] dictionary

one of two distinct categories:

1. Non word errors
2. Real word errors.

→ when an error results in a word that does not appear in a lexicon or valid orthographic word form, it's termed as non word error.

→ Most of the research found the detection of non word error.

→ 2 main techniques < n-gram analysis / dictionary lookup.

→ Now it's solved pblm.

→ Real word error results in actual words of the language.

→ occurs due to typographical mistakes or spelling errors.

eg: substituting the spelling of a homophone or near homophone. such as Peace for Piece
Meat for meet.

→ Real world errors cause:

i) Local Syntactic errors
ii) Global Syntactic errors
iii) Semantic errors
iv) Discourse errors or pragmatic level.

→ To decide a word is wrong is only by contextual information.

⇒ Spelling correction:

→ Detecting
→ correcting errors.

⇒ Spelling correction:

→ Detecting
→ correcting errors.

Error detection:

- process of finding misspelled words.

Error correction

- process of suggesting correct words to a misspelled words.

⇒ 2 ways:

1) Isolated - error detection and correction
2) Context - dependent error detection + correction.

Isolated error detection and correction:

- Each word is checked separately, independent

② Context dependent error detection and Correction:

- utilize the context of a word to detect and correct error.
- it requires grammatical analysis and is thus more Complex and language dependent.
- Even in context, the list of candidate words must first be obtained using an isolated word before making a selection depending on the context.

Spelling error Correction algorithm:

① Minimum edit distance:

the minimum edit distance between two strings is the minimum number of operations required to transform one string into another.

② Similarity key techniques:

the idea is to change a given string into a key such that similar strings will change into the same key. SOUNDEX SIM (odell + Russell 1918) is an example which is used in phonetic spelling correction applns.

③ n gram based techniques:

- used for both non word and real word error detection, bcz in English bi-grams, tri-grams of letter never occur. ex: trigram - qst and bigram - qd.

Minimum edit distance

- Number of insertions, deletion and substitutions required to change one string into another.
- the distance between two strings is the minimum edit distance.

eg: MED blw tutor and tumor is 2

→ substitute 'm' for 't'

→ Insert 'u' before 'r'

- Edit distance between two strings can be represented as binary function, ed which maps two strings to their edit distance. ed is symmetric. For any two strings s and t, ed(s,t) is always equal to ed(t,s).

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 }
```

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

Computing minimum edit distance

[4]

7(a) What is Regular Expression and write down the regexes for email address. [6]

Regular Expressions:

- RE or, Regexes.
- pattern matching std for string parsing and replacement.
- RE can be used to * parse dates * urls and email addresses * log files * configuration files * command line switches * programming scripts.

\n – matches a newline character
\t – matches a tab
\d – matches a digit [0-9]
\D – matches a non digit
\w – matches an alphanumeric character
\W – matches a non alphanumeric character
\s – matches a whitespace character
\S – matches a non whitespace character
\ – escape special characters.
\. – matches a dot
* – matches a *
\\ – matches a blackslash

→ email address consist of a non empty sequence of characters followed by 'at' symbol, @, followed by non empty sequence of characters ending with pattern like .xx, .xxx, .xxxx, etc.

The RE for an email address is:

^[A-Za-z0-9_\.-]+@[A-Za-z0-9_\.-]+[A-Za-z0-9_][A-Za-z0-9_]$

^[A-Za-z0-9_\.-]+

// match a positive no. of acceptable characters at the start of the string.

@ ⇒ matches the @ sign

[A-Za-z0-9_\.-]+

// match any domain name, including a dot

[A-Za-z0-9_][A-Za-z0-9_]$

// match two acceptable characters but not a dot. This ensures that the email address ends with .xx, .xxx, .xxxx, etc.

(b) Describe in detail about Finite State Automata with an example.

Finite State Automata

Some games that we have played that fits the following description:

- Pieces are set up on a playing board
- Dice are thrown or a wheel is spun and a number is generated at random.
- Based on the numbers appearing on the dice, the pieces on the board are rearranged specified by the rules of the game.
- Here, all the possible positions of the piece on the board and call them states.
- State begins → initial state
  winning positions → final state.

→ A finite automaton has the following properties.

(i) A finite set of states, one of which is designated the initial or start state, and one or more of which are designated as final state.

(ii) A finite alphabet set, Σ consisting of input symbols.

(iii) A finite set of transitions that specify for each state and each symbol of I/P alphabet, the state to which it next goes.

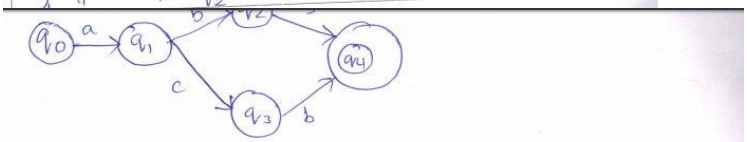
A finite automaton ← deterministic
                   ← Non deterministic
                   ↓
more than one transition out of a state is possible for the same input symbol.

Example:

Suppose $\Sigma = \{a, b\}$, the set of states $= \{q_0, q_1, q_2, q_3, q_4\}$ with q_0 being the start state and q_4 the final state

The transition are:

1. From state q_0 and with input a, go to state q_1
2. " " q_1 " " " b, go to state q_2
3. " " q_1 " " " c, " " " q_3
4. " " q_2 " " " b, " " " q_4



nodes → states, arcs → transitions.

(O) → final state.

⇒ There is exactly one transition leading out of each state hence it is deterministic automaton.

⇒ FSA are used in:

- areas including linguistics
- Electrical engineering
- Computer Science
- Mathematics and logic

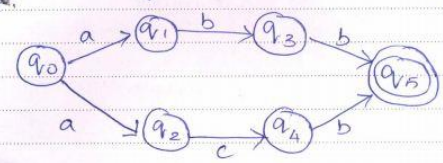
① Deterministic finite state automaton

- 5 tuple $(Q, \Sigma, \delta, S, F)$

Q → Set of states
Σ → alphabet
S → start state
$F \subseteq Q$ → Set of final states
δ → transition function.

→ The transition function of a non-deterministic finite state automaton (NFA) maps $Q \times (\Sigma \cup \{\varepsilon\})$ to a subset of the power set of Q.

→ for each state, there can be more than one transition on a gn. symbol, each leading to a different state.



→ A path is a sequence of transitions beginning with the start state, a path leading to one of the final states is a successful path.

→ FSA's encode regular language.

→ The language that an FSA encodes is the set of strings that can be formed by concatenating the symbols along successful paths. Each string that can be

→ Language defined by this automaton can be regular expression /abb|acb/

→ All transition rules listing is inconvenient, so represent automaton as a state transition table.

Rows → States

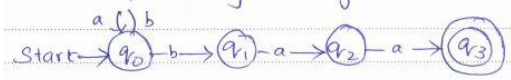
Columns → input

ϕ → missing transition

		Input		
	State	a	b	c
Start:	q_0	q_1	ϕ	ϕ
	q_1	ϕ	q_2	q_3
	q_2	ϕ	q_4	ϕ
	q_3	ϕ	q_4	ϕ
Final:	q_4	ϕ	ϕ	ϕ

... strings containing

NFA implementing this regular expression:



/(a|b)* baa $/

State transition table:

		Input	
	State	a	b
Start:	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	q_1	$\{q_2\}$	ϕ
	q_2	$\{q_3\}$	ϕ
final:	q_3	ϕ	ϕ

Two automata that define the same language are said to be equivalent.

DFA: