Sunanda
Asst. Professor
CSE, DSCE, Blr.

UNIT-01.

NLP approaches

1) Rationalist → person who bases their opinion/action
2) empericist → Experience derived from sensor.

Levels of processing

1) Lexical analysis → Analysis of words. Word level process.
2) Syntactic analysis → Sequence of words.
3) Semantic analysis → Meaning of the language.
4) Discourse Analysis → Interpret the meaning of longer units.
5) Pragmatic analysis → deals with purposeful use of sentences in situation.

Challenges

- not possible to write/imitate as done by humans.
- * multiple meaning associated with the same word
  Ex: bank, can, bat, still.
  ↳ Need to develop models algms to resolve them.
- * Machine will not understand the feelings / Expression of the smile.
- * Try → Sarkasm / others
- Ambiguity: brand of car / hotel / movement

Appln: (grammars of plays)

① Machine Translation ② Speech recognition identify doc. relevant to user query
③ Speech Sythesis (homonym - dece dear) ④ NL Interfaces to DB ⑤ Infor retrieval
↳ ambiguities (rest, interest) ↳ Query using NL sentences
⑥ Infor Extraction ⑦ Question answering ⑧ chatbot
automatic production of summary
⑨ Text summerization ↳ Summarize

Language Modelling

↳ description of some complex entity (or) process.

Language Model → description of language.

approaches

Grammar-based LM Statistical language Modelling.

Grammar-based LM: Uses the grammar of a language to create its model.

↳ Consists of hand-coded rules to define the Structure + ordering of various constituents appearing in unit.

Ex: LFG (Lexical function grammar)

Statistical LM: creates LM by training it from a corpus.

* In order to capture regularities of language, training corpus needs to be large.

Ex: CFG

Various grammar-based language Models:

1. Generative grammars:

* generate sentences in a language by known collection of words & rules in that language.
* Only those sentences that can be generated as per the rules in ~~the language~~ are grammatical.

2) Hierarchical Grammar:

* described classes of grammars in a hierarchical manner where top layer contained the grammars represent -ed by its sub classes.

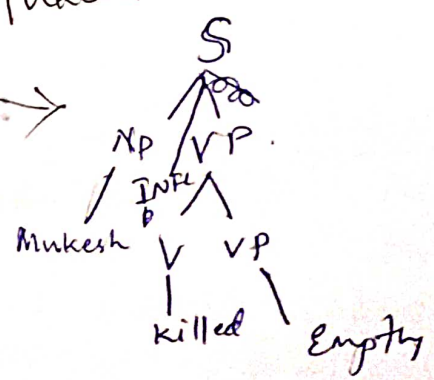
describe grammars at various levels.

- unstructured grammar — type 0
  ↓
- Context-sensitive grammar — type-1
  ↓
- context-free grammar — type-2
  ↓
- Regular grammar — type-3

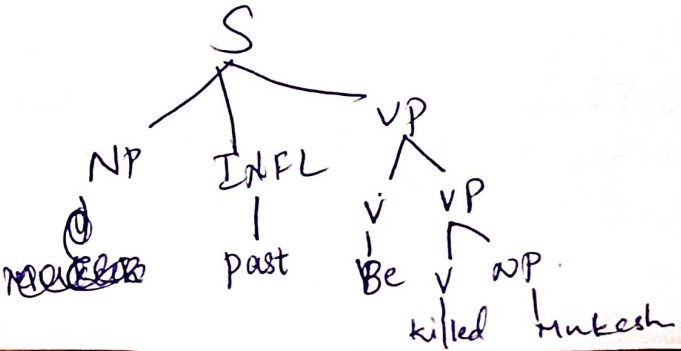
3) Government & Binding:

* ~~How~~ well the sentences are formed by its meaning,
* ~~while~~ sentences are given at the syntactical level and the transformation from meaning to syntax (or) vice versa.
* Tranformational grammars has 2 levels of existence.
  - ↳ Surface level →
  - ↳ Deep root level

Mukesh was killed



INFL → Inflection



i) Components of LFG

A) Lexical Functional grammar (LFG) Model.

* LFG represents sentences at 2 syntactic level.
  - ↳ Constituent Structure (C-structure)
  - ↳ Functional Structure (F-structure)
* LFG is a formal System for grammatical representation in 1982.
* Lexical Functional grammar.

derived from ① ② Composed of 2 terms

the fact that the lexical rules can be formulated to help define the given structure of a sentence.

grammatical functions, such as Subject & object, (or) roles played by various arguments in a sentence.

* C-structure → derived from usual phrase & sentence structure syntax.
* F-structure → final product obtained from C-structure, by functional specifications are annotated.

Statistical Language Model

* It is the probability distribution P(s) over all possible word sequences.

Ex: n-gram model.

(SL) Add-one smoothing

* It is used to "estimate the probability (likelihood) of a sentence"

n-gram Model:

* To estimate the probability of a sentence, decompose Sentence probability into a product of conditional probabilities using chain rule:

$$P(S) = P(w_1, w_2, w_3 \ldots\ldots w_n)$$

$$= P(w_1)\, P(w_2|w_1) P(w_3|w_1 w_2)\, P(w_4|w_1 w_2 w_3) \cdots$$

$$\cdots (P w_n | w_1 w_2 \cdots w_{n-1})$$

$$= \boxed{\prod_{i=1}^{n} P(w_i|h_i)} \quad (h_i \rightarrow \text{history of word } w_i \text{ (or } w_1 w_2 \ldots w_{i-1})$$

* To get Sentence probability, we need to calculate the word probability which is preceding it in a sequence

* The ~~time~~ history (h_i) to the previous one word only.
  ↳ bi-gram ($n=1$)

* previous 2 words → tri-gram ($n=2$)

* $$\boxed{P(S) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})} \rightarrow \text{Bi-gram}$$

$$\boxed{P(S) \approx \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})} \quad \text{Tri-gram.}$$

* Ex: P(east / The Arabian Knights are fairy tales of the)
  ↳ P(east | the) — Bi-gram
  ↳ P(east | of the) — tri gram

* pseudoword (<s>) is introduced to mark the beginning of the sentence
  → bi-gram <s1> trigram → <s1> & <s2>.

Word level Analysis

One way to analyse natural language text is by breaking it down into words, identifying morphological variants, detecting & correcting misspelled words & identify correct part-of-speech of word.

Regular Expressions

pattern-matching standard for string parsing & replacement.

* They are useful tools for the design of language compilers & used for tokenization; describing lexicons, morphological analysis etc..

* R.E are case sensitive. [A-Z] / [a-z] is different.
* Single symbol expression → /a/ [set contains 'a' match.]

RE	Match	Ex
[abc]	Match any of a, b & c	Refresher course will start from the course end in Dec 20
[A-Z]	Any character b/w A & Z	
[^A-Z]	Any char. other than Upper letter	TREC Conference
[abc]	any char. a, b, C	NLP is good.
[a^]	a (or) ^	^ has 3 diff uses

RE	Match	Example
* [Kleene *]	zero or more occurences of previous char	/b*/ → b, bb, ... /[ab]*/ aa, bb, ab aba,
+ [Kleene +]	One or more —11—	/a+/ → ~~a,aa~~ aa, aaa
?	Exactly zero or one occurence of previous char	/example ?/ → example examples.
. (dot) [wildcard char]	any single char.	/.at/ → cat, bat, mat ....
^ [Caret]	match at the beginning of a line	/^The nature \. $/ ↳ exactly for this line
$	match at the end of line	The nature. ---- ---.

Some special characters

RE	Expansion	Description	Example
\d	[0-9]	any digit	Party of 5
\D	[^0-9]	any non-digit	Blue Moon
\w	[a-zA-Z0-9_]	any alphanumeric or space	Daiyu
\W	[^\w]	non-alphanumeric	!!!
\s	[*\t\n\f]	white space (space, tab)	
\S	[^\s]	non-whitespace.	in Concord
\	[*\.\?\$|]	escape special characters	what is this?

→ n-gram <s1> — trigram

Finite State Automata

* ~~No~~ Machine has no. of states ~~are finite~~ & the alphabet of input Symbol is finite [limited / not infinite]
* Machine moves automatically i.e., the change of state is governed by the input.
* FSA recognize the regular languages represented by regular expressions.
* Ex Sheep Talk : /baa+!/
* FSA is a 5-tuple

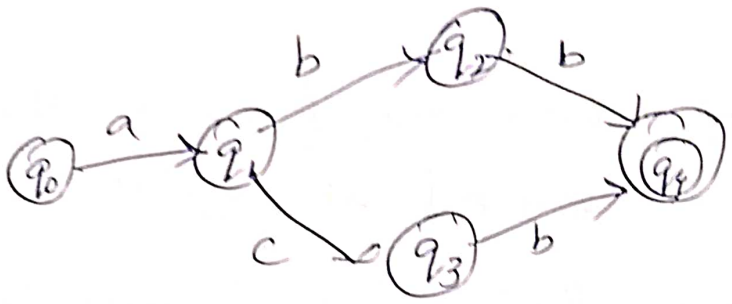
Q : set of states $\{q_0, q_1, q_2 \cdots q_n\}$
Σ : alphabet of symbols $\{a, b, !, \ldots\}$ (set of i/p symbol)
q_0 : A start / initial state
F : set of final states in Q $\{q_1\}$
$\delta(q, i)$: Transition function mapping $Q \times \Sigma$ to Q

* FSA can be deterministic (DFA) or nondeterministic (NFA)

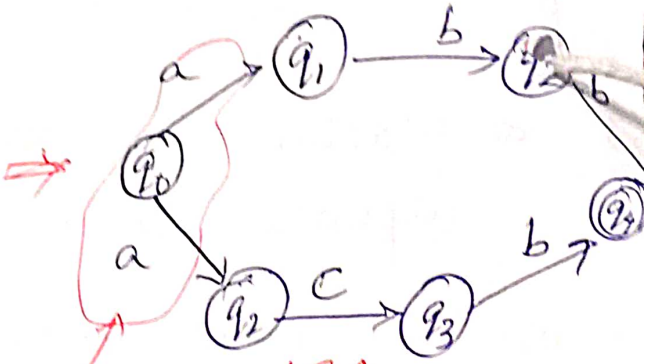
DFA : Only one transition possible from a state on one symbol.

NFA : for each state there can be more than one transition on a given symbol, each leading to a different state.

Ex:

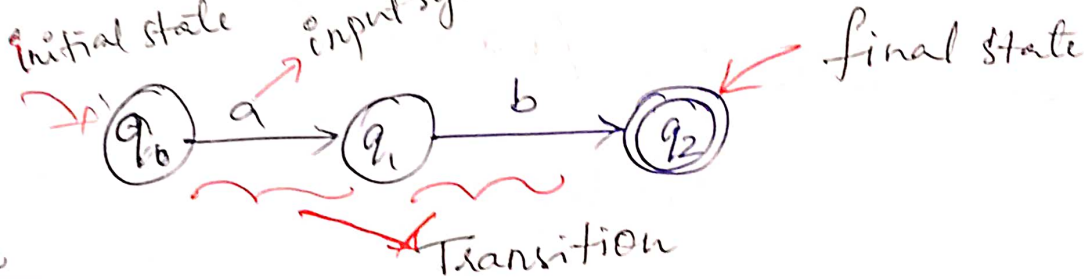


DFA



2 possible transition NFA
from q_0 on i/p symbol 'a'

Ex for FSA:



5-tuple

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

q_0 = initial state

$F = \{q_2\}$

$\delta(q_0, a) \Rightarrow q_1$, $\delta(q_0, b) = q_2$.

State-transition table: list of transitions in table form

where → row → states

Columns → input

Entries in table → transition corresponds to input

→ ϕ → Missing transition.

Ex:

	inputs	
State ↓	a	b
start: q_0	q_1	ϕ
q_1	ϕ	q_2
final: q_2	ϕ	ϕ

Morphological parsing

(taking some i/p & produce some sort of structures for it)

(5)

* "Morphology" is the study of word structure & the formation of words from the smaller units (morphem

* Morphemes → Smallest meaning-bearing units in langua

Ex: eggs → has 2 morphemes.

egg + s

Morphological parser will tell us that 'eggs' is the plural form of noun stem egg.

2 ~~typ~~ classes of Morphemes: (Stem or lemma)

↳ base word
↳ Stem → Main morpheme. (central meaning)

↳ Affixes → Modifies the stem meaning.

- → Prefix: before stem
- → Suffix: end of stem
- → Infix: inside a stem
- → Circumfix: ~~eng~~ Either end of the Stem

3 ways of word formation:

Run - Running

↳ Inflection: root word is combined ~~to~~ to yield same class of stem

↳ Derivation: Combines a word stem with morpheme yield different class of word.

Eg: Computing noun from verb.

Noun → Computation from Verb → Compute

↳ Compounding:

process of merging 2 or more words to form a new word.

Eg: PersonalComputer, Desktop, Overlook. ...

* Morphological parsing takes an input word in a text, as o/p it produces the parsed form consisting of 'canonical form' (or lemma) of the word & a 'set of tags'.

Ex: Part of speech, inflectional properties (gender, number, person, tense ...).

Requirements for morphological parser:

1) Lexicon
2) Morpholatics
3) Orthographic rules.

Lexicon: lists stem & affixes together with basic information about them.

doing → do + V + PP (present participle)
stem Morphological infor

books → book + N + PL (plural noun)

Morphotatics: ordering of morphemes. (or) describes the way morphemes are arranged.

(rest-less-ness) ← valid English word.

Ex: rest - ness - less ← Invalid.

Orthographic rules: Spelling rules that specify the changes that occur when 2 morphemes are combined.

Ex: y → ier , ier → y , (empty) ε → s

(easy → easier) (earlier - early) (bird → birds)

tri-gram

* **two-level morphological model,**

In this model, a word is represented as a correspondence between its lexical level form & its surface level form.

* **Surface level** → represents actual spelling of the word.

* **lexical level** → represents the concatenation of its constituent morphemes.

* Morphological parsing is viewed as a mapping from the surface level into morpheme & feature sequences on the lexical level.

* **Ex:** Surface form represented for 'playing' is

Surface level | P | L | a | y | i | N | g |

Lexical level | P | l | a | y | +V | PL |

Similarly for "books" → Surface form.

"book +N +PL" → Lexical form.

* This Model is implemented with kind of FSA called "Finite-State Transducer (FST)".

* Transducer maps set of symbol to another.

* FST passes over the i/p string & through o/p strings in the form of symbol.

* FST is a 6-tuple $(\Sigma_1, \Sigma_2, Q, \delta, S, F)$

$Q \rightarrow$ set of states

$S \rightarrow$ initial state

$F \rightarrow$ set of final states $(F \subseteq Q)$

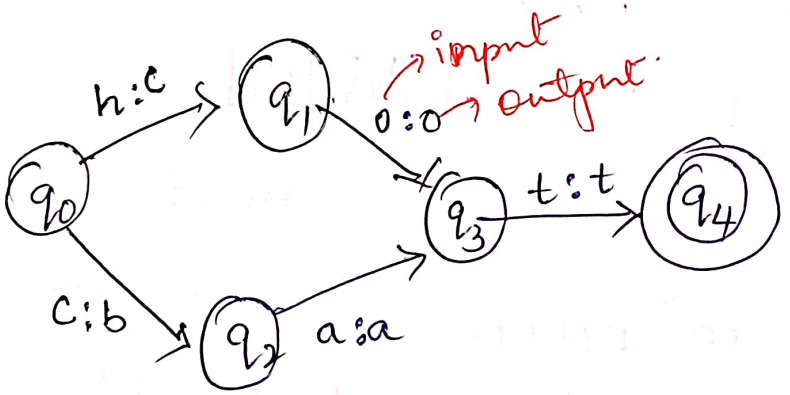
$\Sigma_1 \rightarrow$ input alphabet

$\Sigma_2 \rightarrow$ output alphabet

$\delta \rightarrow$ Function mapping, $Q \times (\Sigma_1 \cup \{\varepsilon\}) \times (\Sigma_2^* \cup \{\varepsilon\})$ to subset of power set of Q.

* FST transitions labelled with symbol from $\Sigma_1 \times \Sigma_2$ (input & output alphabets).

Ex: FST ~~for~~ that accepts 2 i/p strings, hot & cat maps them onto cat & bat respectively.



* 2-level morphology using FST.

~~Scanned~~ (P.T.O)

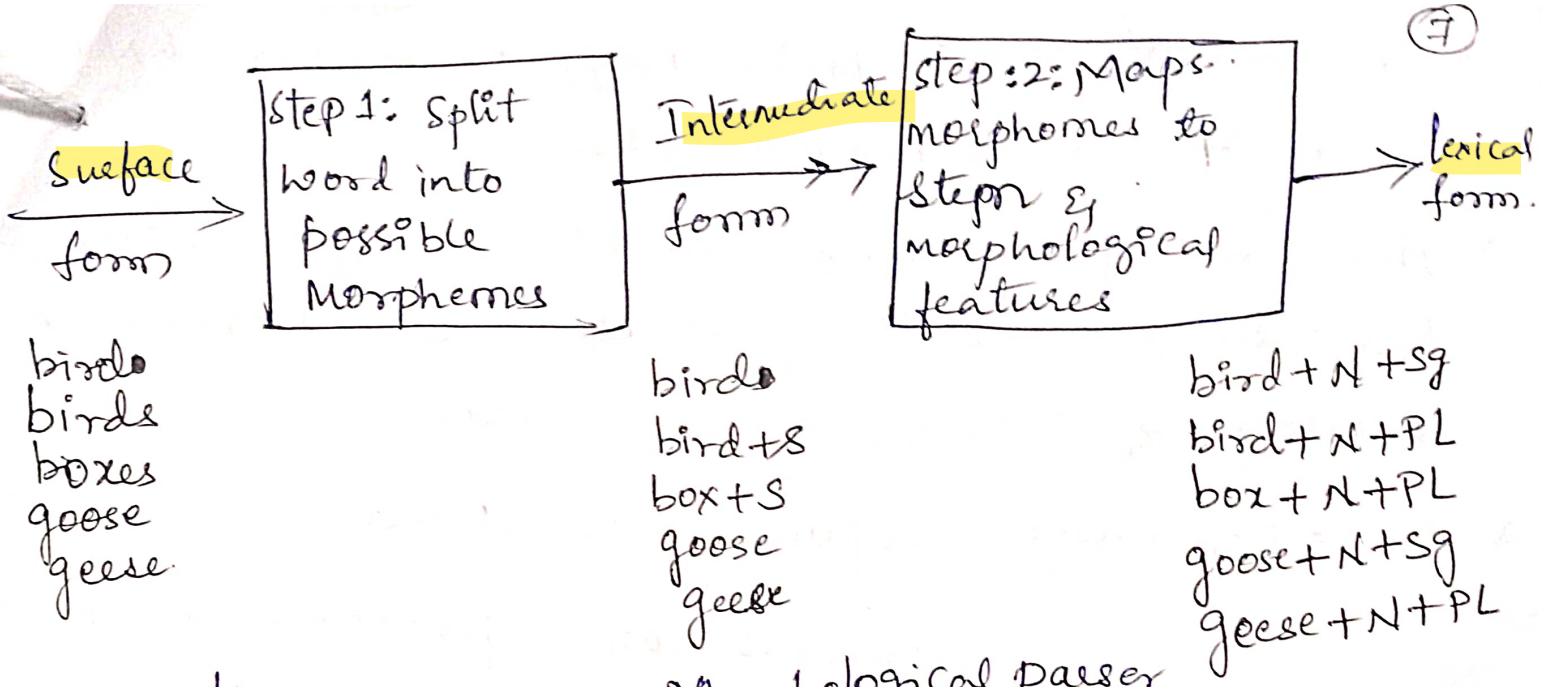


fig: Two-Step-Morphological parser

* first Splits words into its possible components
* Here we also consider spelling rules.

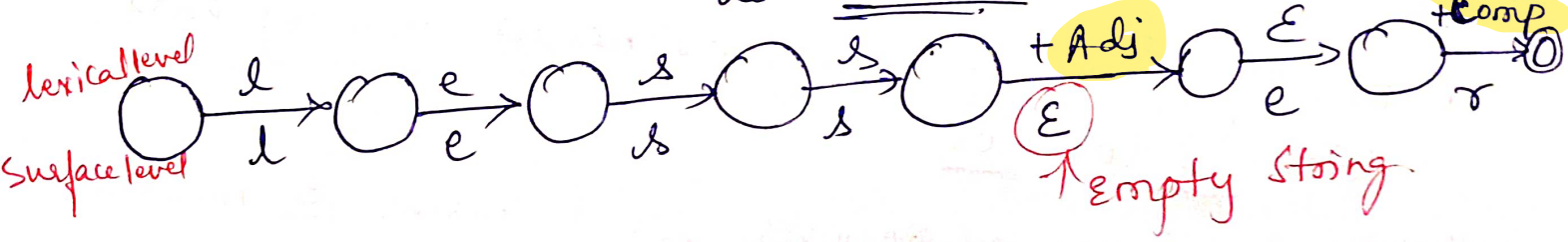
Ex ① birds ↳ bird+s

② boxes ↳ box+s
boxe+s

* The o/p of the 1st step is concatenation of morphemes i.e stems + affixes.

There can be more than one representation for a word.

* FST for adjective less → ~~lesser~~. Comparative form is "lesser"



* In second step, birds mapped to bird+N+PL, & boxes to → box+N+PL.

* Now we found 'boxe' is not a legal stem. This tells splitting boxes into boxe+s is incorrect & therefore discarded.

* In other cases, spouses (or) parses.
  ↓ ↓
  spouse+s (or) parse+s is correct.

* Use Orthographic rules to handle spelling variations
  Ex: add e after -s, -z, -x, -ch, -sh before s.
  [dish → dishes, box → boxes].

* Each of these steps can be implemented with the help of transducer.

* Thus, we need to build 2 transducers,
  ↳ 1. To maps the surface form to intermediate form
  2. To maps intermediate form to lexical form

* FST based Morphological parser for singular & plural noun in English:

* plural form of regular nouns usually ends with "s" (or) "es"

* However, word ending 's' neednot necessarily be the plural form of a word.
  Ex: miss, ass → singular form.

* deletion of 'e' ~~with~~ when introducing a morpheme boundary.
* deletion is usually requires for words ending in xes, ses, zes (Eg. suffixes & boxes). is shown below.

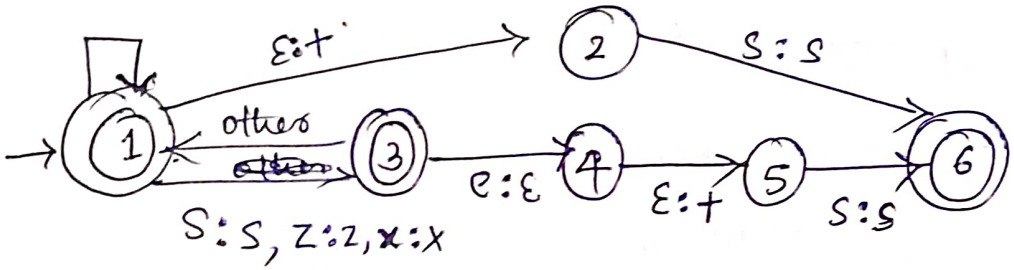
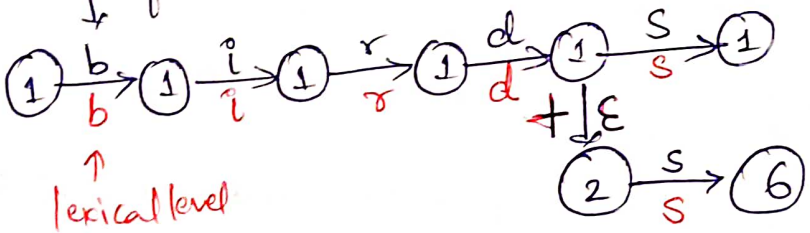


fig:, Simplified FST, mapping English nouns to the intermediate form.

* Possible state sequences for the given surface form birds ~~be boxes~~ ~~ese ifs~~ input is.

surface level.



lexical level

O/P: birds.

o/p : bird+s

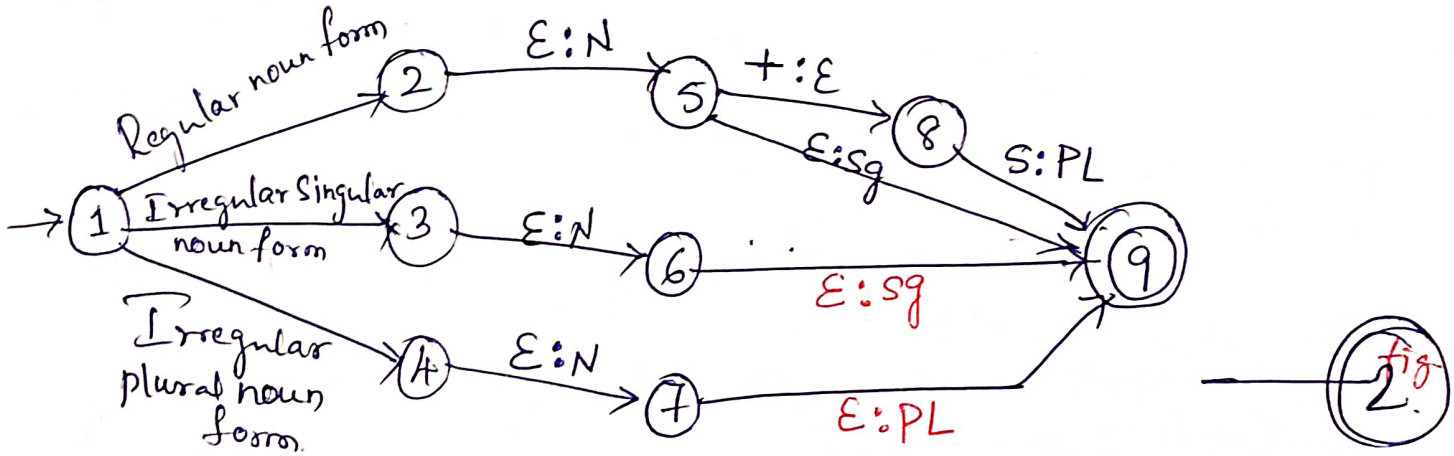


fig: Transducer for Step 2.

* The next step is to develop a transducer that maps from intermediate level to lexical level.

The i/p to the 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)

* The general structure of this transducer shown in fig ②
* Mapping of surface form geese [irregular noun] to its correct stem goose as:

g	o	o	s	e
g	e	e	s	e

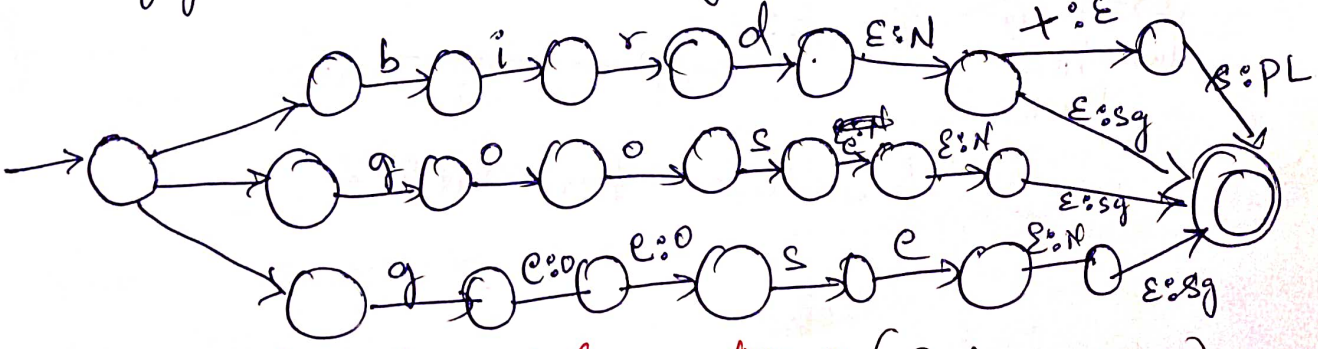
g:g e:o e:o s:s e:e

* bird → b:b i:i r:r + d:d
* Surface word "bird" maps to morphological "bird +N +PL" as follows:

b:b i:i r:r d:d + ε:N +S:PL

[Each letter maps to itself but 'ε' maps to # +N & 's' maps to pl.

* Below fig is the resulting Composed transducer.



Spelling error detection & Correction: (Refer PPT)