Module-2

Word-Level Analysis of Natural Language

- Regular Expression(Regex):
- It is a language for specifying text search strings
- It is a pattern matching standard for string parsing, searching and replacement
- Regular expressions are used to parse dates, urls,email,addresses,log files etc
- It is used for searching texts in Unix tool –gerp.

Examples:

Sl. No	Regular Expression	Match	Example Patterns
1.	/a/	Any string containing 'a'	India is great.
2.	/book/	Any string containing 'book' as a substring	I have a collection of some interesting books.
3.	/old␣books/	Any string containing 'old books' as a substring	A good collection of old books
4.	/[wW]oodchuck/	Woodchuck or woodchuck	Woodchuck
5.	/[abc]/	'a' or 'b' or 'c'	abstract
6.	/[A-Z]/	An uppercase letter	Amazon Incorporation

6.	/[A-Z]/	An uppercase letter	Amazon Incorporation
7.	/[a-z]/	An lower case letter	Silicon
8.	/[0-9]/	A single digit	OR-02
9.	/[^a]/	Any single character except 'a'	ajit
10.	/[^A-Z]/	Any character other than an uppercase letter	TREC Conference
11.	/[^Bb]/	Neither 'B' nor 'b'	Break
12.	/[^abc]/	Any character other than a, b or C	gmail
13.	/[+*?.]/	Match any of +,	5+3=8

14. /[a^]/ Match a or ^ ^ has different uses

15. /[sS]ana/ Match the string 'sana' or 'Sana'

16. /[sS]upernova[sS]/ Match the string supernovas, Supernovas, supernovaS, SupernovaS

17. /supernovas?/ Match supernova and supernovas

18 /a*/ Any string of zero or more a's a, aaa, hello

19. /aa*/ Any string of one or more a's a, aa, aaa, ...

20. /[ab]*/ Any string of zero or more a's or b's aaaa, aaab, abab.

21. /[0-9][0-9]*/ To accept any +ve integers — 231, 2417129, 80, ...

or

/[0-9]+/

22. /^The/ — Matches the word 'The' only at the start of a line

23. /␣$/ — Matches a space at the end of a line

24. /beg.n/ — Matches any character between beg and n. — begin, beg'n, begun

25. /^The dog\.$/ — Matches for a line containing only the phrase 'The dog.'

26. /.at/ — Matches cat, rat, mat etc.

27. /.*/ — For multiple characters

28. /\b[Tt]he\b/ — Matches 'The' or 'the' but not 'other'

28. /\b[Tt]he\b/ — Matches 'The' or 'the' but not 'other'

29. /cat|dog/ — Matches either the string cat or dog.

30. /gupp(y|ies)/ — Match guppy & guppies

31. /(Column⊔[0-9]+⊔*)*/ — Match Column1 Column2 Column 3 ...

Note

1) Characters are grouped by putting them between square brackets. The use of brackets specifies a disjunction of characters.

2) A dash is used to specify a range.

3) Regular expression specify what a single character cannot be by the use of caret (^) at the begining

4) Regular expressions are case sensitive

5) The use of question mark makes the preceding character optional.

6) The * operator, called Kleene * (cleany star) specify ~~limited~~ repeated occurrences of a character.

7) The kleene + provides a shorter notation to specify one or more of the previous character

8) The caret (^) is also used as an anchor to specify a match at the begining of a line

9) The dollar sign ($) is used to specify a match at the end of a line

10) The dot matches any single character (except a carriage return)

11) The anchor \b matches a word boundary.

12) We use disjunction operator (pipe symbol) to match string 1 or string 2.

13) Enclosing a pattern in parentheses makes it act like a single character for the purpose of neighbouring operators like the pipe and a kleene *

Operator precedence hierarchy for regular expressions

The following table gives the order of RE operator precedence from highest to lowest:

Parentheses	()
Counters	* + ? {}
Sequences & anchors	the ^my end$
Disjunction	|

- Write RE for the following:

1. to accept strings book or books
2. To accept colour and color.
3. To accept all variations of MHz,Mhz,mHz,mhz,MegaHertz,Megahertz,megaHertz,megahertz
4. To accept any +ve integer with an optional decimal point
5. To check a string is an email address or not.

1. Write a RE to accept strings book or books

/book|books/ or /books?/ or /[Bb]ooks?/ most appropriate

2) Write a RE to accept strings colour and color

/colou?r/

3) Write a RE to accept all variations of ie

MHz, Mhz, mHz, mhz, MegaHertz, Megahertz, megaHertz, megahertz

/[Mm][Hh]z|[Mm]ega[Hh]ertz/

4) Write a RE which will accept any +ve integer with an optional decimal point

/[0-9]+(\.[0-9]+)?/

3) Write a RE to accept all variations of ie
MHz, Mhz, mHz, mhz, MegaHertz, Megahertz, megaHertz, megahertz

`/[Mm][Hh]z|[Mm]ega[Hh]ertz/`

4) Write a RE which will accept any +ve integer with an optional decimal point

`/[0-9]+(\.[0-9]+)?/`

5) Write a RE to check a string is an email address or not.

`/^[A-Za-z0-9_\.-]+@[a-z\.]$/`

Advanced Operators

- The RE /{3}/ means exactly 3 occurrences of the previous characters. For example, /a\.{5}b/ will match 'a' followed by five dots followed by 'b'.

- Some of the similar operators for counting are summarized below:

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	zero or one occurrences of the previous char or expression
{n}	n occurrences of the previous char or expression
{n, m}	from n to m occurrences of the previous char or expression
{n, }	at least n occurrences of the previous char or expression

The following are the some aliases for common ranges which can be used mainly to save typing

RE	Expansion	Match	Examples
\d	[0-9]	any digit	Party of 5
\D	[^0-9]	any non-digit	Blue diamond
\w	[a-zA-Z0-9]	any alphanumeric/ underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	non-whitespace	in depth

There are some special characters which are referred t
by special notation based on the backslash (\)

RE	Match	Examples
*	an asterisk	KA*PA*A
\.	a period	Dr. Das
\?	a question mark	How are you?
\n	a newline	
\t	a tab	

Finite State Automata (FSA)

- A finite state automata is defined by the following five parameters

$Q = q_0 q_1 q_2 \cdots q_{n-1}$: a finite set of n states

Σ = a finite input alphabet of symbols

q_0 = the start state

F = the set of final states, $F \subseteq Q$

$\delta(q, i)$ = transition function

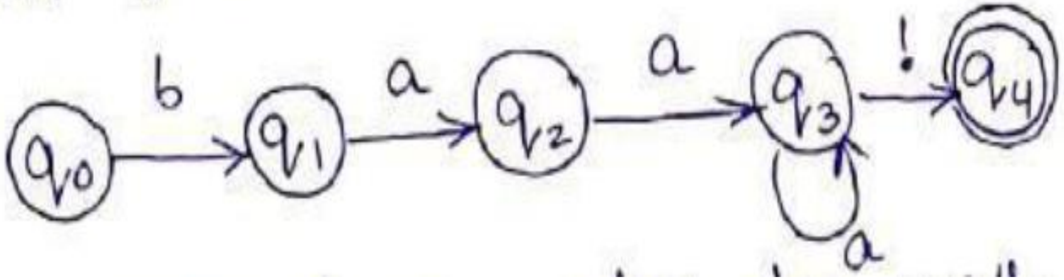
Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q, i)$ returns a new state $q' \in Q$. q' maybe same as q.

- A regular expression is one way of describing FSA.
- Regular expression, FSA and regular grammar are used to characterize a particular kind of formal language called regular language.

- Use of an FSA to recognize sheeptalk
  * We define the sheep language as any string from the following (infinite) set:

    baa!, baaa!, baaaa!, baaaaa!, ...

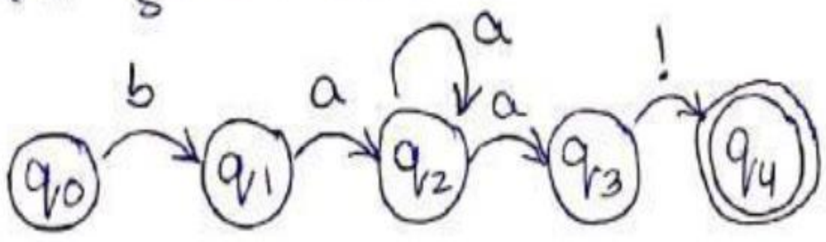
  * The Regular expression for this kind of sheeptalk is
    /baa+!/
  * The FSA for the same sheeptalk is as follows:

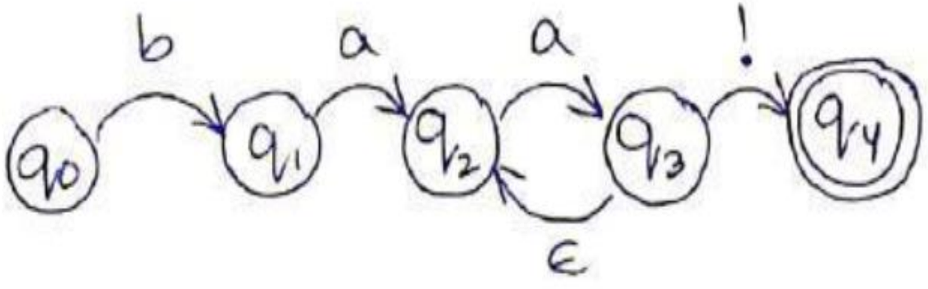


- Non-deterministic FSA (NFSA)

* Automata with decision points are called non-deterministic FSA

* Two variations of NFSA for the sheep language is given below





Morphological Parsing

- Morphology studies word structure and the formation of words from smaller meaning-bearing units called morphemes.
- The goal of morphological parsing is to discover the morphemes that build a given word.

  Example: cats = cat + s
  (stem) (affix)

(stem) (affix)

- There are two broad classes of morphemes: stems & affixes
  - ✓ stem is the main morpheme that contains the central meaning
  - ✓ Affixes modify the meaning given by the stem.

  Affixes can be any one of the following 4 types:

  a) suffix: morphemes applied to the end of stem

     Eg: reading, quickly

  b) prefix: morphemes which appear before a stem

     Eg: unhappy, unlock, preexisting

  c) infix: morphemes that appear inside the stems

     Eg: speedometer

  ... that appear on both

d) circumfix: morphemes that appear on both sides of the stem

Eg: unrecognized, unhappiness, rewrites

of word formation:

3 main ways of word formation:

1. Inflection: a root is combined with a grammatical morpheme(number, tense, case, gender) to yield a word of the same class as the original stem

- Brought-bring

2. Derivation: Creates a new word by changing part-of-speech

It combines a word stem with a grammatical morpheme to yield a word belonging to a different class

Ex: 1. computation from compute

    2. teacher from teach

- 3. Compounding: It is the process of merging two or more words to form a new word

Ex:desktop,overlook

Finite-State Morphological Parsing

- The problem of parsing morphology takes any word as input and outputs the stem with its additional features.
- Some of the sample input/output is shown below:

Input	Output (Morphological Parse)
cats	cat + N + PL
cat	cat + N + SG
cities	city + N + PL
geese	goose + N + PL
goose	goose + N + SG
goose	goose + V
merging	merge + V + PresPart

How to build a morphological parser?

1. Lexicaon: the list of stems and affixes, together with basic information about them
2. Morphotactics: The model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word

Ex: rest-less-ness is a valid word and not rest-ness-less

3. Orthographic rules: These spelling rules are used to model the changes that occur in the word

Ex: the y---- ier spelling rule changes easy to easier and not easyer

- Morphological analysis can be avoided if an exhaustive lexicon is available that lists features for all the word-forms of all the roots
- Given a word, we simply consult the lexicon to get its feature values
- This has several limitations:

1. It puts heavy demand on memory
2. Fail to show the relationship between different roots having similar word forms
3. It is not practical to list all possible word-forms in a language

- The simplest morphological systems are stemmer that converts morphological variations of a given word to its stem
- Stemmers don’t require a lexicon, instead it uses a set of rewrite rules of the form:

ex : ier---y (earlier--→ early)

ing --- nothing or epsilon (playing ---→ play)

- Stemming algorithm works in 2 steps:

1. Suffix removal: this step removes predefined ending from words
2. Recoding: this step adds predefined ending to the output of the first step.

Ex : easier -→ easy

Drawbacks:

Stemmers are not perfect for morphologically rich languages

- Different levels of morphological model:

1. Surface level: represents the actual spelling of the word
2. Lexical level: represents the concatenation of its constituents morphemes

- In this model , a word is represented as a correspondence between its lexical level form and its surface level form
- ex   books
-         book +N+PL

- This model is usually implemented with FST(Finite State Transducer)

Finite State Transducer(FST)

- It is a type of finite automation which maps between two set of symbols
- It passes over the input string by consuming the input symbols from the input tape and produces the output in the form of symbols

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

Q: set of states

S: initial state

F: set of final states, $F \subseteq Q$

Σ_1: input alphabet

Σ_2: output alphabet

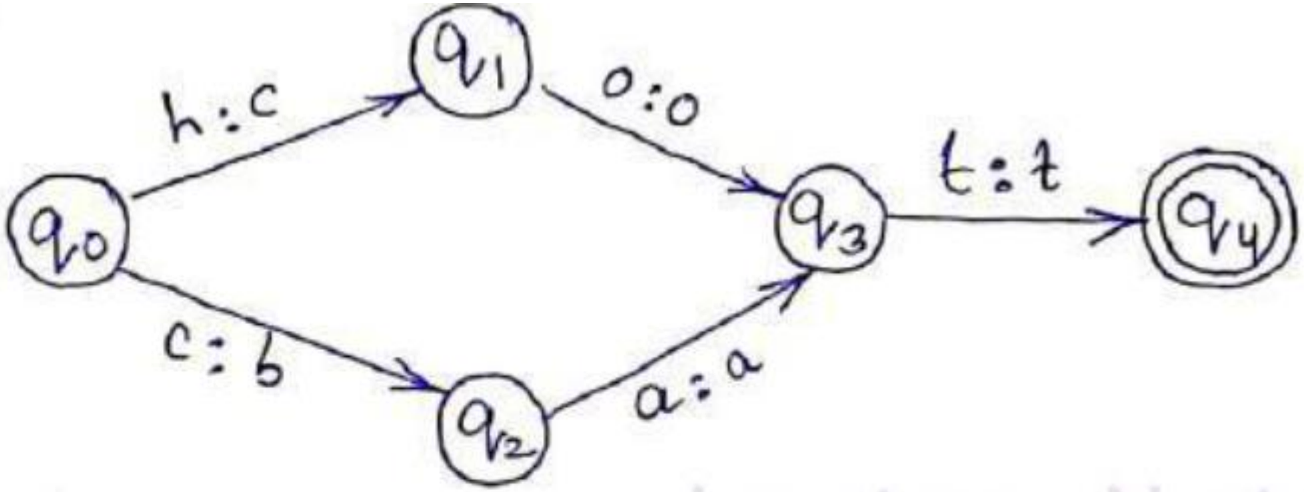
δ : function mapping

$Q \times (\Sigma_1 \cup \{\epsilon\}) \times (\Sigma_2 \cup \{\epsilon\})$

to a subset of power set of

- The transducers can be seen as automata with ttransitions labelled with symbols from $\Sigma_1 \times \Sigma_2$

- The following figure shows a simple transducer that accepts two input strings hot and cat and maps them onto cot and bat



Spelling Error Detection and Correction

- 80% of the typing errors and misspellings are:

1. Substitution of a single letter
2. Omission of a single letter
3. Insertion of a single letter
4. Transposition of two adjacent letters

- Shafer and Hardwick(1968) found that the common errors in decreasing order of occurrence are substitution, omission, insertion

- Optical Character Recognition (OCR) and other automatic reading devices introduce errors of substitution, deletion and insertion
- Unlike typing errors , spelling errors are mainly phonetic where the misspell word is pronounced in the same way as the correct word
- Spelling errors belong to two distinct categories:

1. Non-word errors
2. Real-word errors
3. Non word error: when an error results in a word that does not appear in a given lexicon or is not a valid orthographic word form

ex: leve------→leave

- 2. Real word error: when errors result in a word is another actual word of the language
- It may cause local syntactic error, global syntactic error .

- Spelling correction consists of detecting and correcting errors
- These problems are addressed in two ways:

1. Isolated error detection and correction:
    - Each word is checked separately , independent of its context
    - Simple solution is to look up the word in a lexicon
    - There are some problems associated:
    - it needs the existence of a lexicon containing all correct words which would take a long time to compile and occupy a lot of space

- For some languages it is impossible to list all the correct words
- Strategy fails for real –world errors
- Larger the lexicon , more likely it is that an error goes undetected
- Because the chance of a word being found is greater in a large lexicon

- Context – dependent error detection and correction:
- Utilize the context of a word to detect and correct errors
- Requires grammatical analysis and hence more complex and language dependent
- The list of candidate words must first be obtained using an isolated – word method, before making a selection depending on the context

Spelling correction algorithms are broadly classified as:

- 1. Minimum edit distance:
- The minimum edit distance between 2 strings is the minimum number of operations (insertions,deletions,sustitutions) required to transform on string into another

2. Similarity Key techniques:

In this we change a given string into a key such that similar strings will be changed into the same key

3. n-gram based technique:

- Can be used for both non-word and real-word error detection
- Strings that contain unusual n-grams can be identified as possible spelling errors
- This technique needs a large corpus as training data
- In he case of real world error detection , we calculate the likelihood of one character following another and use this information to find possible correct word candidates

- 4. Neural nets: these have the abilty to do associative recall based on incomplete and noisy data

5. Rule based techniques:

A set of rules (heuristics) derived from knowledge of a common spelling error patterns is used to transform misspelled words into a valid words

Minimum Edit Distance

- It is the number of insertions, deletions and substitutions required to change one string into another

Eg: Minimum edit distance between tutor & tumour is 2
i.e. ed(tutor, tumour) = 2

- For any two strings s and t, $ed(s,t) = ed(t,s)$

Eg:

minimum edit distance between "tutor" and "tumour" is 2.

Source string : t u t o - r

t u m o u r

replace / substitute

insert

minimum 2 edit operations

- $\boxed{ed(s,t) = ed(t,s) = \text{min no. of edit operations}}$

minimum edit distance is symmetric

- Implementation

- It can be implemented using DP (Dynamic programming) paradism.

- The dynammic 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 symbol of target string.

	" "	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

- For any two strings s and t, $ed(s,t) = ed(t,s)$
- Levensthein distance between two sequences is obtained by assigning a unit cost to each operation
- We use Dynamic Programming algorithms for finding minimum edit distance between two sequences
- this method uses an edit distance matrix in which the $(i,j)^{th}$ cell represents the distance between the first i characters of the source and the first j characters of the target string.

- The value in each cell is computed as

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

Algorithm

Input: Two strings X and Y

Output: The minimum edit distance between X and Y

1. $m \leftarrow \text{length}(X)$
2. $n \leftarrow \text{length}(Y)$
3. for $i \leftarrow 0$ to m do
4. $\quad dist[i, 0] \leftarrow i$
5. for $j \leftarrow 0$ to n do
6. $\quad dist[0, j] \leftarrow j$

```
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(X_i, Y_j),
                          dist[i, j-1] + delet_cost }
```

	#	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

Words and Word Classes

- Words are classified into categories called part-of-speech or word classes or lexical categories
- These lexical categories are defined by their syntactic and morphological behaviours
- Some of the word classes in English Language are:

NN	noun	student, chair, proof
VB	verb	read, walk, study
ADJ	adjective	large, high, tall, less
		carefully, slowly, uniformly

and morphological [illegible]

- Some of the word classes in English Language are:

NN	noun	student, chair, proof
VB	verb	read, walk, study
ADJ	adjective	large, high, tall, less
JJ	adverb	carefully, slowly, uniformly
IN	preposition	in, on, to, of
PRP	pronoun	I, me, they
DET	determiner	the, a, an, this, those