UKDW

Universitas Kristen Duta Wacana

Dr. Lucia D. Krisnawati

NLP Basics:

Words, Regular Expressions, and Automata

29 Agustus 2017

Outline

Outline

- Basic concept & terminologies
- Regular Expression
- Finite State Automata

Basic Terminologies

Type, Token and Term

- “Rose is a rose is a rose is a rose”.
- How many words does the sentence above have?
- Token:
  - An individual occurrence of a symbol or string (‘word’)
  - An instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing (Manning & Schuetze, 2009)
- Type:
  - A class of all tokens containing the same character sequences
- Term:
  - A normalized type that is included in the IR’s system, indexing, and any other NLP processes.

Basic Terminologies

Type, Token, Terms

But what are thoughts? Well, we all have them. They are variously described as ideas, notions, concepts, impressions, perceptions, views, beliefs, opinions, values, and so on. At times they are brief, coming and going in an instant. On other occasions they seem to endure and we can mull them over again and again in the act we call thinking. We can put them aside, fall asleep, and then return to them later. We refer to them as things we can handle. However, this is just a metaphor.

Basic Terminologies

Type, Token, Terms

rank	word	freq	rank	word	freq
1	we	6	6	they	3
2	and	5	7	To	3
3	them	5	8	again	2
4	are	3	9	as	2
5	can	3	10	in	2

Basic Terminologies

Stopwords are:

- Function words or common words words whose frequency are really high in a document collection, therefore they play no roles in searching, eg.
  - English: words in the former table
  - Indonesian: maka, dia, dan, atau, mereka, yang, dst
- Common treatment to stopwords:
  - They are often discarded in indexing or excluded from the vocabulary

Basic Concept

Preprocessing

- Tokenization:
  - The task of chopping a given text into pieces called *token*.
  - eg. But what are thoughts? Well, we all have them.
    - But | what | are | thoughts | well | we | are | all | have | them
- Text normalization:
  - A task of transforming a text into a normal form to make it consistent by:
    - replacing multiple white spaces into a single one
    - Lower cases
    - Removing punctuations and non-readable characters
    - Removing numbers or transforming them into words, eg 1 → one
    - Transforming acronym into their normal forms, etc.

Basic Concepts

Points to Ponder:

- Is tokenization a trivial task?
  - Discuss it with your friends where you see the difficulties of tokenization
- What do you think the challenges of text normalization?
- Which will be processed first: tokenization or text normalization?

Basics Concepts

Preprocessing

- Stemming: a crude heuristic process that chops off the end of words in the hope of achieving the base form, and often includes the removal of derivational affixes eg:
  - Coming, came, come, comes, cometh → come
  - menyapu, disapu, tersapu, sapuan → sapu
- Lemmatization: the process of removing inflectional ending properly with the use of a vocabulary and morphological analysis of words, normaly aiming to remove inflectional endings and to return the base or dictionary form of a word (Manning & Schuetze, 2009), eg
  - Coming, came, come, comes, cometh → come
  - menyapu, disapu, tersapu, sapuan → sapu
- Stemming & Lemmatization have the same aim but they differ in their flavor!!!

Basic Concepts

Information:

- Classified as Western Austronesian Language, Indonesian inherits the agglutinative characteristic, which process of word building is done mainly by affixation and reduplication
- Modern Indonesian is included into a partly isolated language where the number of loan words from foreign languages is quite a lot

Points to Ponder

- Which task fits to preprocess Indonesian text?
    - Provide reasons for your arguments.

Regular Expression

Regular Expression (RE):

- Is a special language that is used for specifying simple classes of string for text searching.
- RE search requires:
  - Pattern to search for
  - A corpus → a collection of documents
- RE characteristics: case sensitive

Regular Expression

Formal Definition:

Let Σ be an alphabet. The regular Expression over the alphabet Σ is defined recursively as follows:

- Ø and ε are regular expressions
- A is a regular expression for each a ∈ Σ,
- If r and s are regular expreessions, then so are r ∪ s , r · s, and r*
- No other sequences of symbols are regular expressions

(Kelly, D, 1995)

Regular Expression

Basic RE Patterns

- The search string can consist of a single letter, /!/ or a sequence of letters, /song/

RE	Example
/woodschucks/	“interesting links to woodchucks and lemurs”
/a/	“Mary Ann stopped at Mona’s”

- A square bracket [ ] is used to specify a disjunction operation , with dash (-) bracket signifies a range

RE	Match	Example Pattern
/[Ww]oodchuck/	Woodchuck or woodchuck	“Woodchuck”
/[0-9]/	Any digit	“plenty of 7 and 9”

Wildcards in RE

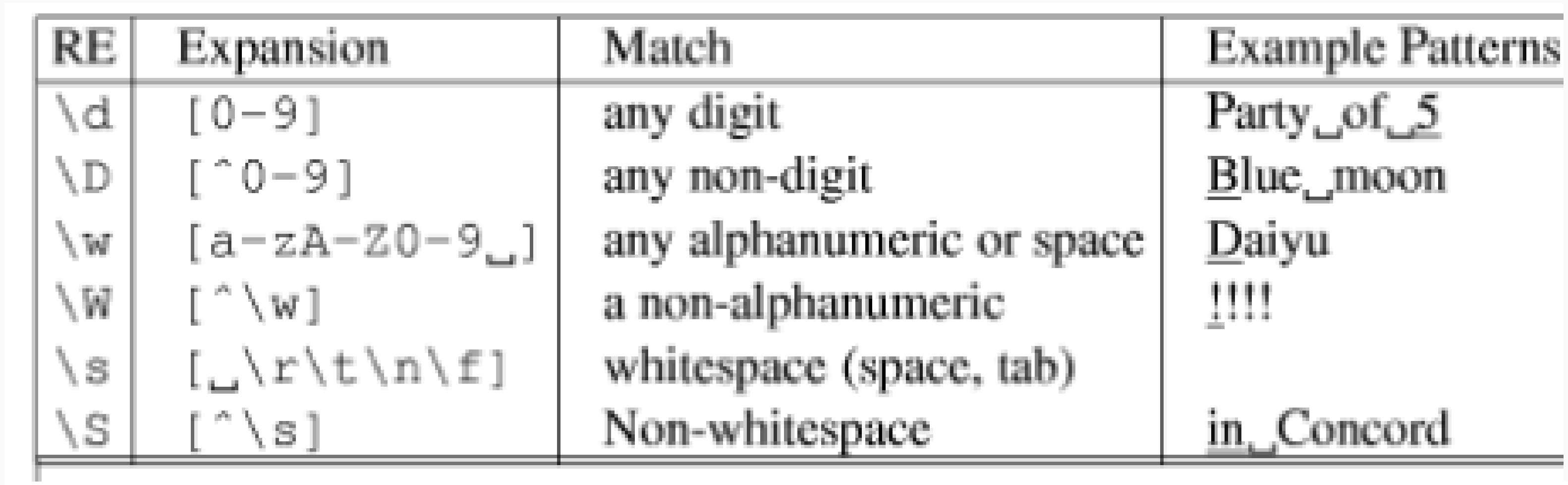
Symbol/ usage	RE	Match	examples
Caret: For negation or ^	/[^A-Z]/	Not an upper case	“HUT Indonesia Raya”
Question mark: Optionality/previous expression	/Colou?r/	Color or colour	“The color of Indonesian flag..”
Kleene Star: Zero or more occurrences	/[ab]*/	Zero or more ‘a’s or ‘b’s	Ø, a, b, aaaa, ababab, bbbb
Kleene plus: 1 or more occurrences	/[ab]+/	One or more ‘a’s or ‘b’s	a, b, aaaa, ababab, bbbb
Period: Any characters	/beg.n/	Any character between ‘beg’ and ‘n’	Begin, began, begun

Regular Expressions

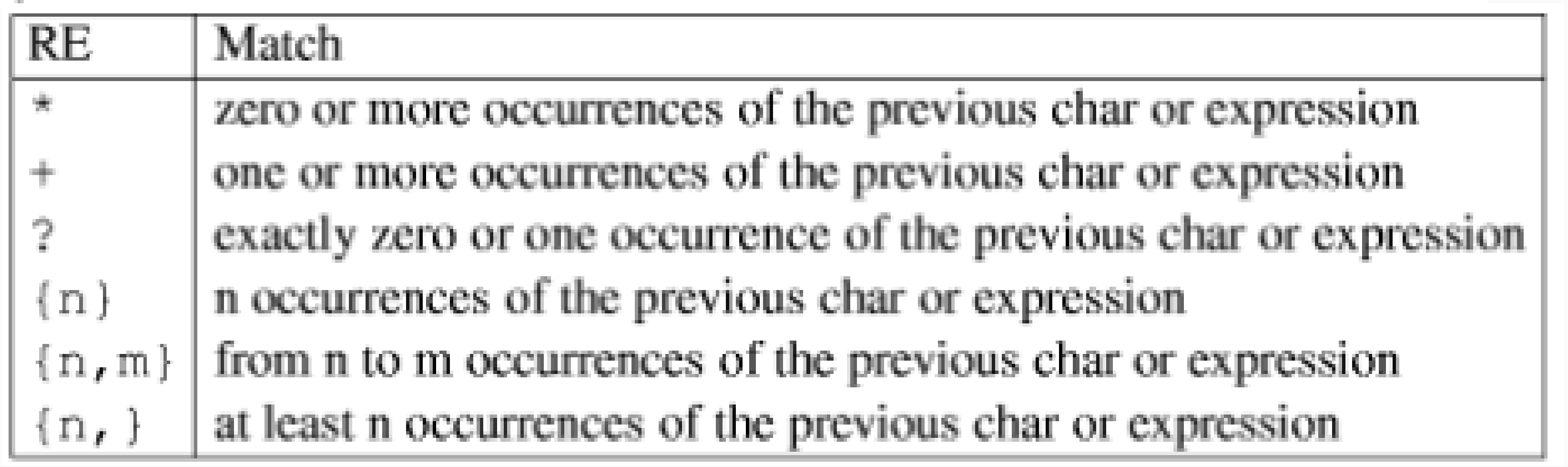
Metacharacters in RE

Symbol/usage	RE	Match	Example
Pipe " | ": disjuction	/cat|dog/ /gupp(y|ies)/	cat or dog Guppy or guppies	"Some cats and dogs are..." "The guppy also known as..."
Paranthesis ( ) Grouping, precedence	/[0-9]+(\.[0-9][0-9])? /	Any integers or decimal points	"155 people have approximately income of $ 150.85 "
\b Word boundary	/\b[tT]he\b/	Match 'the' or 'The' but not "Theater"	"The a team has been released..."
Caret ^ : Beginning of line	/(^|[^a-b])/	Beginning of a line or non-alphabetic characters	"125 juta penduduk Indonesia….."
Dollar $: End of line	/end$/	Match any characters 'end' at the end of line	"……… They depend on the road"

Advanced Operators



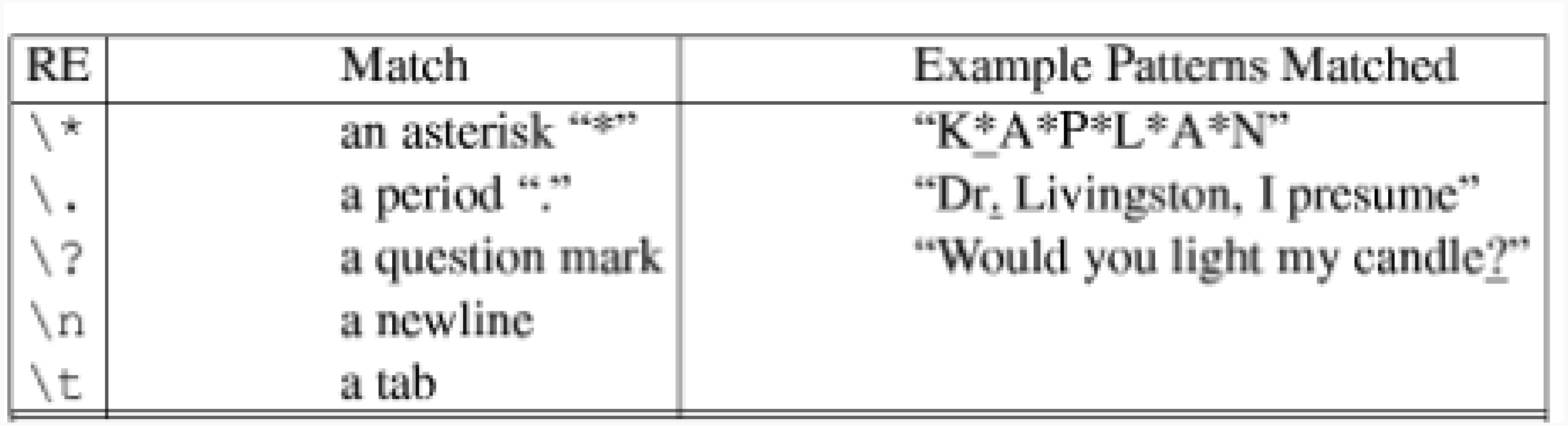
RE	Expansion	Match	Example Patterns
`\d`	`[0-9]`	any digit	Party␣of␣5
`\D`	`[^0-9]`	any non-digit	Blue␣moon
`\w`	`[a-zA-Z0-9␣]`	any alphanumeric or space	Daiyu
`\W`	`[^\w]`	a non-alphanumeric	!!!!
`\s`	`[␣\r\t\n\f]`	whitespace (space, tab)	
`\S`	`[^\s]`	Non-whitespace	in␣Concord



RE	Match
`*`	zero or more occurrences of the previous char or expression
`+`	one or more occurrences of the previous char or expression
`?`	exactly zero or one occurrence 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

Regular Expressions

Advanced Operators



RE	Match	Example Patterns Matched
`\*`	an asterisk "*"	"K*A*P*L*A*N"
`\.`	a period "."	"Dr. Livingston, I presume"
`\?`	a question mark	"Would you light my candle?"
`\n`	a newline	
`\t`	a tab	

RE Applications

RE Substitution, Memory , and Eliza

- A simple use of RE is in substitution
  - s/colour/color/
  - S/([0-9]+)/\1/
- Substitution using memory → Eliza
  - Demo on Eliza http://www.manifestation.com/neurotoys/eliza.php3
  - Eliza works by having a cascade of RE substitution
  - s/.* YOU ARE (depressed|sad) .* / I AM SORRY TO HEAR YOU ARE \1 /
  - s/.* YOU ARE (depressed|sad) .* / WHY DO YOU THINK YOU ARE \1 /
  - S/.* all .*/ IN WHAT WAY/

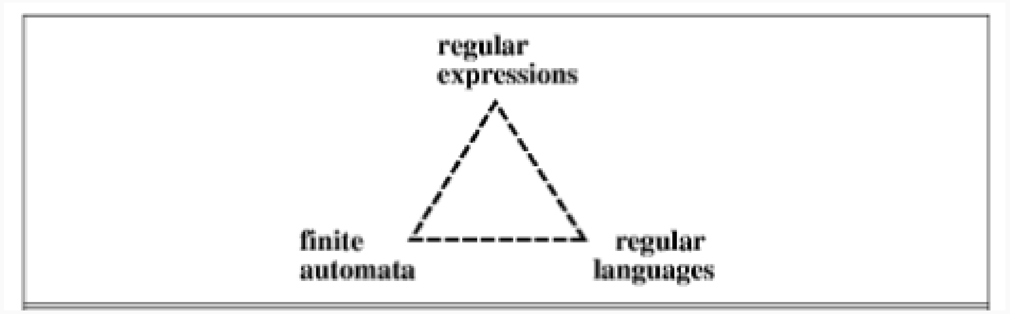
Exercises

Write regular expressions for:

- finding multiple white spaces and changing them into a single white spaces
- Matching a set of all lower case alphabetic strings ending in ‘ng’
- Matching a set of all strings in lower case with 2 consecutive repeated words, eg anak-anak, etc
- finding all kinds of punctuation and removing them from the text
- Finding all strings separated by a hyphen located at the end of a line and be continued on the beginning of a line.

Finite State Automata (FSA)

RE, FSA, and Regular Languages



Finite state automata:

a.k.a. Finite Automaton, Finite State Machine, FSA or FSM

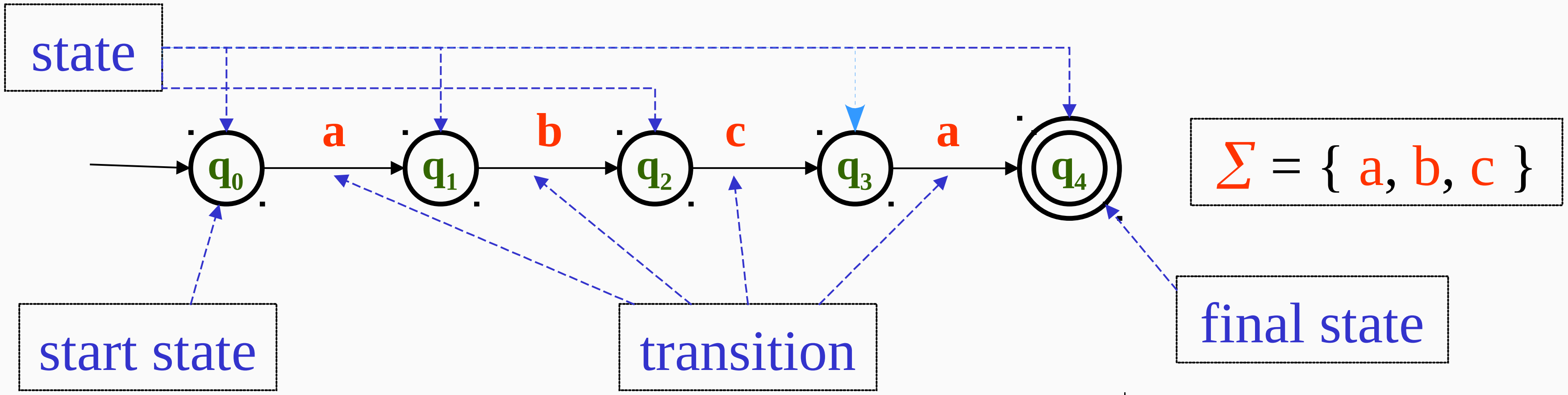
- An abstract machine which can be used to implement regular expressions (etc.).
- Has a finite number of states, and a finite amount of memory (i.e., the current state).
- Can be represented by directed graphs or transition tables

Finite State Automata

Representation:

- An FSA may be represented as a directed graph; each node (or vertex) represents a state, and the edges (or arcs) connecting the nodes represent transitions.
- Each state is labeled.
- Each transition is labeled with a symbol from the alphabet over which the regular language represented by the FSA is defined, or with ε, the empty string.
- Among the FSA’s states, there is a start state and at least one final state (or accepting state).

Finite State Automata



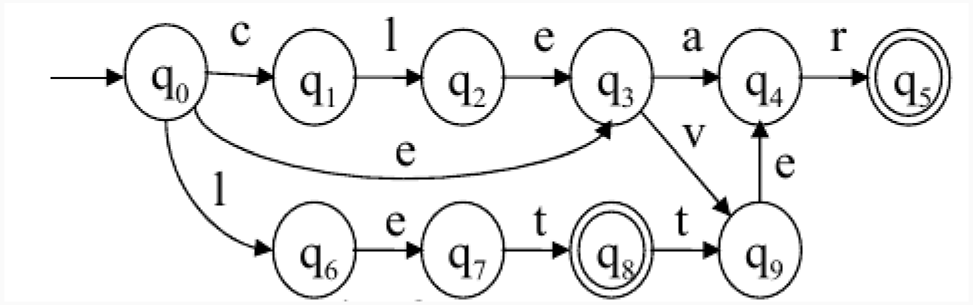
Representation

An FSA may also be represented with a state-transition table. The table for the above FSA

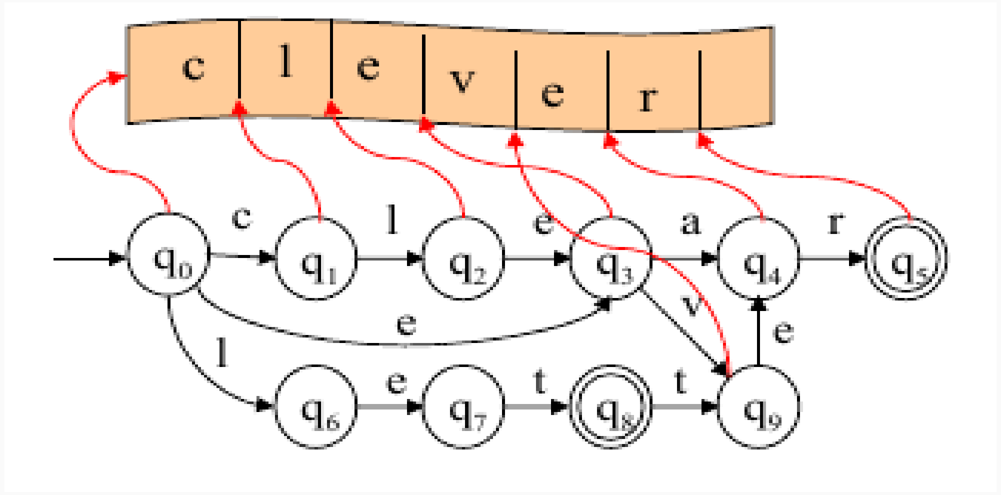
	Input		
State	a	b	c
0	1	∅	∅
1	∅	2	∅
2	∅	∅	3
3	4	∅	∅
4	∅	∅	∅

Finite State Automata

Transition Graph



Traversal of FSA



$S = q_0 \quad F = \{q_5, q_8\}$

Transition function $\delta: \Phi \times \Sigma \rightarrow \Phi$

$\delta(q_0,c)=q_1$

$\delta(q_0,e)=q_3$

$\delta(q_0,l)=q_6$

$\delta(q_1,l)=q_2$

$\delta(q_2,e)=q_3$

$\delta(q_3,a)=q_4$

$\delta(q_3,v)=q_9$

$\delta(q_4,r)=q_5$

$\delta(q_6,e)=q_7$

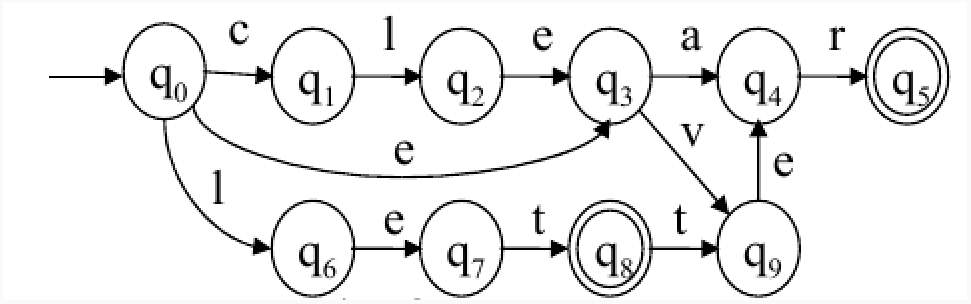
$\delta(q_7,t)=q_8$

$\delta(q_8,t)=q_9$

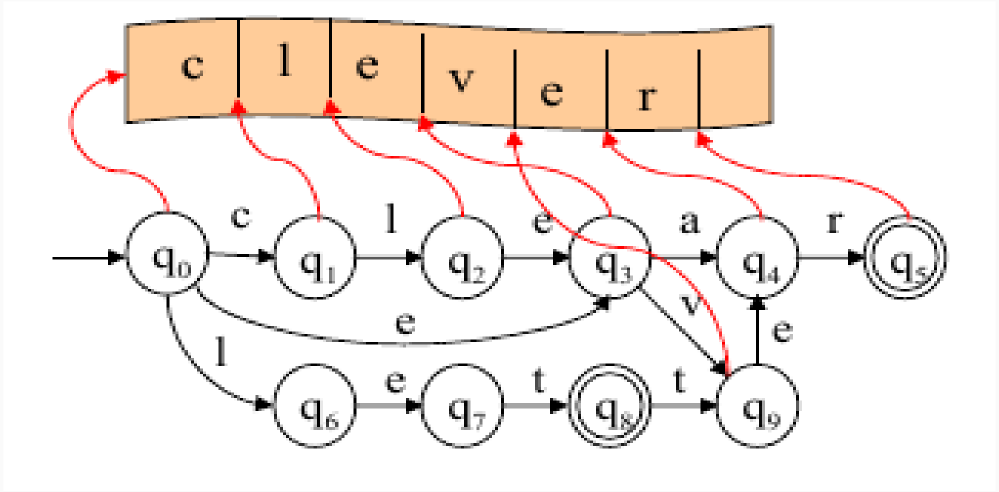
$\delta(q_9,e)=q_4$

Finite State Automata

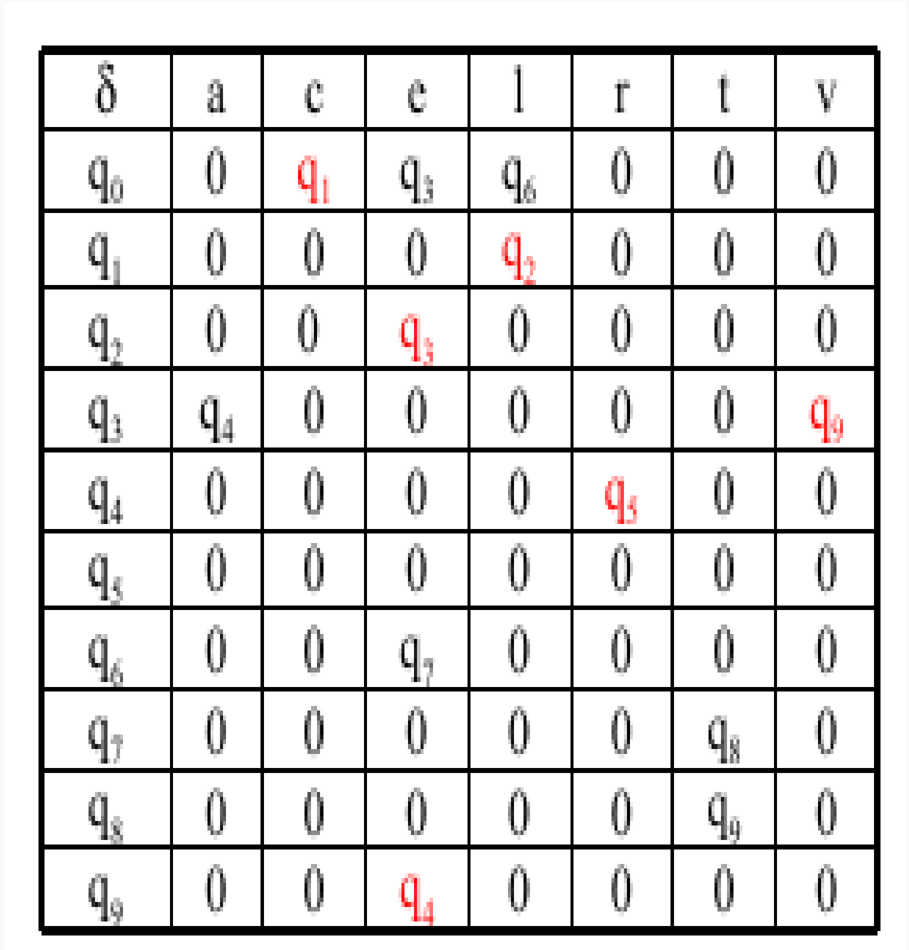
Transition Graph



Traversal of FSA



state diagram



δ	a	c	e	l	r	t	v
q_0	0	q_1	q_3	q_6	0	0	0
q_1	0	0	0	q_2	0	0	0
q_2	0	0	q_3	0	0	0	0
q_3	q_4	0	0	0	0	0	q_9
q_4	0	0	0	0	q_5	0	0
q_5	0	0	0	0	0	0	0
q_6	0	0	q_7	0	0	0	0
q_7	0	0	0	0	0	q_8	0
q_8	0	0	0	0	0	q_9	0
q_9	0	0	q_4	0	0	0	0

Finite State Automata

Traversal/recognition as search:

- Recognition → selection of the correct path of all possible paths
- Search strategy can affect efficiency: in what order should the paths be searched?
  - Depth-first (LIFO [last in, first out]; stack)
  - Breadth-first (FIFO [first in, first out]; queue)
  - Depth-first uses memory more efficiently, but may enter into an infinite loop under some circumstances

Finite State Automata

FSA With Output

- Finite Automata may also have an output alphabet and an action at every state that may output an item from the alphabet
- Useful for lexical analyzers
  - As the FSA recognizes a token, it outputs the characters
  - When the FSA reaches a final state and the token is complete, the lexical analyzer can use
    - Token value – output so far
    - Token type – label of the output state

Exercise

On-Hand exercise:

- Write an FSA for time-of-day expression like ‘eleven o’clock’, twelve-thirty or a quarter to ten