Natural Language Processing

Chapter 2

Regular Expressions, Text Normalization, Edit Distance

Department of Computer Science and Engineering,
Shahjalal University of Science and Technology, Sylhet

Regular Expression (RE)

- A language for specifying text search strings.

Regular Expression (RE)

- A language for specifying text search strings.
- Formally,
  - an algebraic notation for characterizing a set of strings.

Basic RE Patterns: Simplest Kind

- The simplest kind of RE is
  - a sequence of simple characters.

Basic RE Patterns: Simplest Kind

- The simplest kind of RE is
  - a sequence of simple characters.

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/!/	"You've left the burglar behind again!" said Nori

Basic RE Patterns: Simplest Kind

- The simplest kind of RE is
  - a sequence of simple characters.

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/!/	"You've left the burglar behind again!" said Nori

- We'll show REs delimited by slashes (/).

Basic RE Patterns: Simplest Kind

- The simplest kind of RE is
  - a sequence of simple characters.

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/!/	"You've left the burglar behind again!" said Nori

- We'll show REs delimited by slashes (/).
- But slashes (/) are not part of the REs.

Basic RE Patterns: Case Sensitivity

- Regular expressions are **case sensitive**.

Basic RE Patterns: Case Sensitivity

- Regular expressions are **case sensitive**.
- Lower case /s/ is distinct from uppercase /S/.

Basic RE Patterns: Case Sensitivity

- Regular expressions are **case sensitive**.
- Lower case /s/ is distinct from uppercase /S/.
- The pattern /woodchucks/ will not match the string *Woodchucks*.

Basic RE Patterns: Disjunction

- The string of characters inside the square braces [and] specifies
  - a **disjunction of characters** to match.

Basic RE Patterns: Disjunction

- The string of characters inside the square braces [and] specifies
  - a **disjunction of characters** to match.

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

Basic RE Patterns: Range

- Where there is a well-defined sequence associated with a set of characters,
  - the brackets can be used with the dash (-) to specify any one character in a range.

Basic RE Patterns: Range

- Where there is a well-defined sequence associated with a set of characters,
  - the brackets can be used with the dash (-) to specify any one character in a range.

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	"we should call it 'Drenched Blossoms' "
/[a-z]/	a lower case letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

Basic RE Patterns: Negation

- If the caret $\widehat{\ }$ is the first symbol after the open square brace [,
  - the resulting pattern is negated.

Basic RE Patterns: Negation

- If the caret $\hat{}$ is the first symbol after the open square brace [,
  - the resulting pattern is negated.
- If the caret $\hat{}$ occurs anywhere else,
  - it usually stands for a caret $\hat{}$.

Basic RE Patterns: Negation

- If the caret ^ is the first symbol after the open square brace [,
  - the resulting pattern is negated.
- If the caret ^ occurs anywhere else,
  - it usually stands for a caret ^.

RE	Match (single characters)	Example Patterns Matched
/[^A-Z]/	not an upper case letter	"Oyfn pripetchik"
/[^Ss]/	neither 'S' nor 's'	"I have no exquisite reason for't"
/[^.]/	not a period	"our resident Djinn"
/[e^]/	either 'e' or '^'	"look up ^ now"
/a^b/	the pattern 'a^b'	"look up a^ b now"

Basic RE Patterns: Optionality

- How can we talk about optional elements,
  - like an optional *s* in *woodchuck* and *woodchucks*?

Basic RE Patterns: Optionality

- Use the question mark /?/,
  - which means "zero or one instances of the previous character".

Basic RE Patterns: Optionality

- Use the question mark /?/,
  - which means "zero or one instances of the previous character".

RE	Match	Example Patterns Matched
/woodchucks?/	woodchuck or woodchucks	"woodchuck"
/colou?r/	color or colour	"color"

Basic RE Patterns: Problem

- Consider the language of certain sheep, which consists of strings that look like the following:
  baa!
  baaa!
  baaaa!
  baaaaa!
  ...

Basic RE Patterns: Problem

- Consider the language of certain sheep, which consists of strings that look like the following:
  baa!
  baaa!
  baaaa!
  baaaaa!
  ...
- How can we represent the language using RE?

Basic RE Patterns: Kleene *

- Use the *

Basic RE Patterns: Kleene *

- Use the *
- Pronounced “cleany star”.

Basic RE Patterns: Kleene *

- Use the *
- Pronounced "cleany star".
- Means "zero or more occurrences of the immediately previous character or regular expression".

Basic RE Patterns: Kleene *

- An integer (a string of digits) is thus /[0-9][0-9]*/.

Basic RE Patterns: Kleene *

- An integer (a string of digits) is thus /[0-9][0-9]*/.
- Why isn't it just /[0-9]*/?

Basic RE Patterns: Kleene +

- Have to write the regular expression for digits twice,
  - so there is a shorter way to specify “at least one” of some character.

Basic RE Patterns: Kleene +

- Have to write the regular expression for digits twice,
  - so there is a shorter way to specify "at least one" of some character.
- Kleene + means "one or more occurrences of the immediately preceding character or regular expression".

Basic RE Patterns: Kleene +

- Have to write the regular expression for digits twice,
  - so there is a shorter way to specify "at least one" of some character.
- Kleene + means "one or more occurrences of the immediately preceding character or regular expression".
- An integer (a string of digits) is thus /[0-9]+/.

Basic RE Patterns: Kleene +

- Have to write the regular expression for digits twice,
  - so there is a shorter way to specify "at least one" of some character.
- Kleene + means "one or more occurrences of the immediately preceding character or regular expression".
- An integer (a string of digits) is thus /[0-9]+/.
- Two ways to specify the sheep language: /baaa*!/ or /baa+!/.

Basic RE Patterns: Wildcard

- The period (/./),
  - a wildcard expression that matches any single character (except a carriage return).

Basic RE Patterns: Wildcard

- The period (/./),
  - a wildcard expression that matches any single character (except a carriage return).

RE	Match	Example Matches
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Basic RE Patterns: Wildcard

- The wildcard (.) is often used together with the Kleene *
  - to mean “any string of characters”.

Basic RE Patterns: Wildcard

- The wildcard (.) is often used together with the Kleene *
  - to mean “any string of characters”.
- Suppose we want to find any line in which a particular word, for example, aardvark, appears twice.

Basic RE Patterns: Wildcard

- The wildcard (.) is often used together with the Kleene *
  - to mean "any string of characters".
- Suppose we want to find any line in which a particular word, for example, aardvark, appears twice.
- We can specify this with the regular expression /aardvark.*aardvark/.

Basic RE Patterns: Anchors

- Special characters that anchor regular expressions to particular places in a string.

Basic RE Patterns: Anchors

- Special characters that anchor regular expressions to particular places in a string.
- The most common anchors are
  - the caret $\hat{}$ and the dollar sign \$.

Basic RE Patterns: Anchor caret ^

- The caret ^
  - matches the start of a line.

Basic RE Patterns: Anchor caret ^

- The caret ^
  - matches the start of a line.
- The pattern /^The/ matches
  - the word *The* only at the start of a line.

Basic RE Patterns: The caret ^

- The caret ^ has three uses:

Basic RE Patterns: The caret ^

- The caret ^ has three uses:
  - To match the start of a line,

Basic RE Patterns: The caret ^

- The caret ^ has three uses:
  - To match the start of a line,
  - To indicate a negation inside of square brackets,

Basic RE Patterns: The caret ^

- The caret ^ has three uses:
  - To match the start of a line,
  - To indicate a negation inside of square brackets,
  - Just to mean a caret ^.

Basic RE Patterns: Anchor dollar sign $

- The dollar sign $ matches
  - the end of a line.

Basic RE Patterns: Anchor dollar sign $

- The dollar sign $ matches
  - the end of a line.
- ␣$
  - a useful pattern for matching a space at the end of a line.

Basic RE Patterns: Anchor dollar sign $

- The dollar sign $ matches
  - the end of a line.
- ␣$
  - a useful pattern for matching a space at the end of a line.
- /^The dog\.$/
  - matches a line that contains only the phrase *The dog.*

Basic RE Patterns: Anchor dollar sign $

- The dollar sign $ matches
  - the end of a line.
- ␣$
  - a useful pattern for matching a space at the end of a line.
- /^The dog\.$/
  - matches a line that contains only the phrase *The dog.*
- We have to use the backslash (\) here
  - since we want the . to mean "period" and not the wildcard.

Basic RE Patterns: Anchors more

- Two other anchors:
  - \b matches a word boundary
  - \B matches a non-boundary.

Basic RE Patterns: Anchors more

- Two other anchors:
  - \b matches a word boundary
  - \B matches a non-boundary.
- /\bthe\b/
  - matches the word *the* but not the word *other*.

Basic RE Patterns: “word” definition

- Based on the definition of “words” in programming languages.

Basic RE Patterns: “word” definition

- Based on the definition of “words” in programming languages.
- Technically, a “word” for the purposes of a regular expression is defined as
  - any sequence of digits, underscores, or letters.

Basic RE Patterns: “word” definition

- Based on the definition of “words” in programming languages.
- Technically, a “word” for the purposes of a regular expression is defined as

  - any sequence of digits, underscores, or letters.
- /\b99\b/

Basic RE Patterns: “word” definition

- Based on the definition of “words” in programming languages.
- Technically, a “word” for the purposes of a regular expression is defined as
  - any sequence of digits, underscores, or letters.
- /\b99\b/
  - will match the string 99 in *There are 99 bottles of beer on the wall* (because 99 follows a space)

Basic RE Patterns: “word” definition

- Based on the definition of “words” in programming languages.
- Technically, a “word” for the purposes of a regular expression is defined as

  - any sequence of digits, underscores, or letters.

- /\b99\b/
  - will match the string 99 in *There are 99 bottles of beer on the wall* (because 99 follows a space)
  - but not 99 in *There are 299 bottles of beer on the wall* (since 99 follows a number).

Basic RE Patterns: "word" definition

- Based on the definition of "words" in programming languages.
- Technically, a "word" for the purposes of a regular expression is defined as

  - any sequence of digits, underscores, or letters.

- /\b99\b/
  - will match the string 99 in *There are 99 bottles of beer on the wall* (because 99 follows a space)
  - but not 99 in *There are 299 bottles of beer on the wall* (since 99 follows a number).
  - But it will match 99 in $99 (since 99 follows a dollar sign ($), which is not a digit, underscore, or letter).

Break

Basic RE Patterns: Disjunction operator

- Search for either the string *cat* or the string *dog*.

Basic RE Patterns: Disjunction operator

- Search for either the string *cat* or the string *dog*.
- Why can't we say /[catdog]/?

Basic RE Patterns: Disjunction operator

- Search for either the string *cat* or the string *dog*.
- Why can't we say /[catdog]/?
- Use the disjunction operator, also called the pipe symbol |.

Basic RE Patterns: Disjunction operator

- Search for either the string *cat* or the string *dog*.
- Why can't we say /[catdog]/?
- Use the disjunction operator, also called the pipe symbol |.
- The pattern /cat|dog/ matches
  - either the string *cat* or the string *dog*.

Basic RE Patterns: Precedence

- How can I specify both *guppy* and *guppies*?

Basic RE Patterns: Precedence

- How can I specify both *guppy* and *guppies*?
- We cannot simply say /guppy|ies/,
  - because that would match only the strings *guppy* and *ies*.

Basic RE Patterns: Precedence

- How can I specify both *guppy* and *guppies*?
- We cannot simply say /guppy|ies/,
  - because that would match only the strings *guppy* and *ies*.
- Use the parenthesis operators (and).

Basic RE Patterns: Precedence

- How can I specify both *guppy* and *guppies*?
- We cannot simply say /guppy|ies/,
  - because that would match only the strings *guppy* and *ies*.
- Use the parenthesis operators (and).
- The pattern /gupp(y|ies)/ would specify that
  - we meant the disjunction only to apply to the suffixes y and ies.

Basic RE Patterns: Operator precedence hierarchy

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

- Counters have a higher precedence than sequences,
  - /the*/ matches *theeeee* but not *thethe*.

Basic RE Patterns: Operator precedence hierarchy

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

- Counters have a higher precedence than sequences,
  - /the*/ matches *theeeee* but not *thethe*.
- Sequences have a higher precedence than disjunction,
  - /the|any/ matches *the* or *any* but not *thany* or *theny*.

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)
- Solution: `/\b[tT]he\b/`

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)
- Solution: `/\b[tT]he\b/`
- Might want to find *the* in some context where it might also have underlines or numbers nearby (*the_* or *the25*)

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)
- Solution: `/\b[tT]he\b/`
- Might want to find *the* in some context where it might also have underlines or numbers nearby (*the_* or *the25*)
- Solution: `/[^a-zA-Z][tT]he[^a-zA-Z]/`

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)
- Solution: `/\b[tT]he\b/`
- Might want to find *the* in some context where it might also have underlines or numbers nearby (*the_* or *the25*)
- Solution: `/[^a-zA-Z][tT]he[^a-zA-Z]/`
- Won't find the word *the* when it begins a line.

Basic RE Patterns: A Simple Example

- Write a RE to find cases of the English article *the*.
- Solution: `/the/`
- Miss the word when it begins a sentence and hence is capitalized (i.e., *The*)
- Solution: `/[tT]he/`
- Still incorrectly return texts with the embedded in other words (e.g., *other* or *theology*)
- Solution: `/\b[tT]he\b/`
- Might want to find *the* in some context where it might also have underlines or numbers nearby (*the_* or *the25*)
- Solution: `/[^a-zA-Z][tT]he[^a-zA-Z]/`
- Won't find the word *the* when it begins a line.
- Solution: `/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/`

NLP Errors

Two kind of Errors:

- false positives:
  - strings that we incorrectly matched like *other* or *there*

NLP Errors

Two kind of Errors:

- false positives:
  - strings that we incorrectly matched like *other* or *there*
- false negatives:
  - strings that we incorrectly missed, like *The*.

NLP Errors: Two efforts

Two kind of Efforts:

- Increasing **precision** (minimizing false positives)

NLP Errors: Two efforts

Two kind of Efforts:

- Increasing **precision** (minimizing false positives)
- Increasing **recall** (minimizing false negatives)

A More Complex Example

- Write a RE for

  - any machine with at least 6 GHz and 500 GB of disk space for less than $1000.

A More Complex Example

- Write a RE for
  - any machine with at least 6 GHz and 500 GB of disk space for less than $1000.
- HOME WORK: Section 2.1.4

More RE Operators: Aliases

- Aliases for common sets of characters.

More RE Operators: Aliases

- Aliases for common sets of characters.

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\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␣Concord

More RE Operators: Counters

- RE operators for counting.

More RE Operators: Counters

- RE operators for counting.

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
{,m}	up to *m* occurrences of the previous char or expression

More RE Operators: Backslashed

- Some characters that need to be backslashed.

More RE Operators: Backslashed

- Some characters that need to be backslashed.

RE	Match	First Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Why don't they come and lend a hand?"
\n	a newline	
\t	a tab	

THANK YOU