Morphology

Dereje T. Abzaw

Outline

- Regular Expressions
- Operators
- Tokenization
- Segmentation
- Normalization , Lemmatization and Stemming

Regular Expression - RegEx

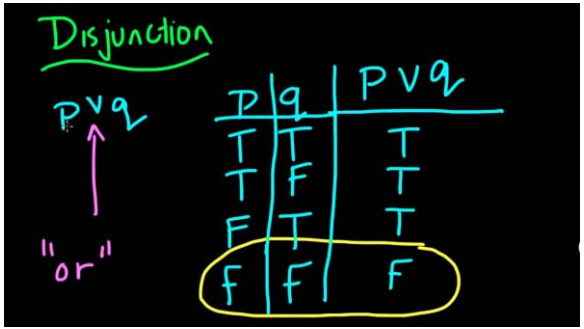
- One of the unsung successes in standardization in computer science has been the regular expression (often shortened to **regex**), a language for specifying text search strings.
- This practical language is used in every computer language, word processor, and text processing tools.
- Formally, a **regular expression** is an algebraic notation for characterizing a set of strings.
- Regular expressions are particularly useful for **searching in texts**, when **we have a pattern** to search for and a corpus of texts, be it single document or a collection, to search through.
- A regular expression search function will search through the corpus, returning all texts that match the pattern.
- For example, the Unix command-line tool **grep , Findstr**
- What does a **grep** do?
    - Takes a regular expression and returns every line of the input document that matches the expression.

Basic Regular Expression Patterns

- The simplest kind of regular expression is a sequence of simple characters; putting characters in sequence is called **concatenation**.
- Regular expressions are **case sensitive**; lower case is distinct from upper case.
- **Substring Search:** /Woodchucks/

Regex	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

Contd...(Regex)



- To solve capitalization differences, we use a **square braces [and].**
  - **Disjunction of Characters (**OR**)**

Regex	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"

- /[1234567890]/ **- Single digit**
- /[ABCDEFGHIJKLMNOPQRSTUVWXYZ]/ **- Any Capital Letter**

Contd...(Regex)

- **Range** - Usage of hyphens (-)
  - /[2-5]/ - Numbers between 2 and 5
  - *What is the verdict on the borders **2** and **5**?, Do you think it includes them?*
- **Negation** - Usage of caret (^)
  - /[^a]/ - Not small letter “a”
  - is the first symbol after the open square brace [

Regex	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”

Contd ... (Regex)

- **Question Mark (?)** - It makes the preceding token in the regular expression optional. This means that the pattern can occur zero or one time to match.
- **Kleene * or Asterisk (Cleany star) (*)** - It allows the preceding token to appear zero or more times. So, it matches the preceding character (or group of characters) an arbitrary number of times, including none.
- **Kleene (+)** - Similar to the asterisk, but the preceding token must appear one or more times. It ensures that the character (or group of characters) before it appears at least once.
- **Period (.)** - It is a wildcard that matches any single character except newline characters.
- **Dollar Sign ($)** - It signifies the end of a string or line. Using it in a pattern specifies that the preceding characters must be at the end of the string.
- **Backslash b (\b)** - Represents a word boundary. This means it matches positions where one side is a word character (like letters or digits) and the other side is not a word character (for instance, a space or the start/end of a string).
- **Backslash B (\B)** - The opposite of `\b`. It matches in positions where both sides are word characters or both sides are not word characters. It's useful for finding matches of characters or patterns that are not at the boundaries of words

Contd … (Regex)

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

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

Regex	Match
^	start of line
$	end of line
\b	word boundary
\B	non-word boundary

Contd … (Regex)…More Operators

Regex	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)	in_Concord
\S	[^\s]	Non-whitespace	in␣Concord

Regex	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	zero or one occurrence of the previous char or expression
{n}	exactly *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

Contd … More Operators

Regex	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	

Reading Assignment

- What counts as word,
- Difference between Word and Corpora
- Meanings of
  - **Utterance**
  - **Disfluency**
  - **Fragments**, **Fillers** or **Filled pauses.**
  - **Code Switching**
  - **Word Instances**

Tokenization

- Word tokenization is a fundamental step where a piece of text is divided into smaller units called **tokens**.
- *These tokens can be **words**, **numbers**, or **punctuation marks**.
- The main goal of tokenization is to break down text into pieces that can be analyzed or processed individually, making it easier for algorithms to understand and work with the text.
- The techniques we use are easily extended to any other units of meaning contained in a sequence of characters, like **ASCII emoticons**, **Unicode emojis**, **mathematical symbols**, and so on.
- Will tokenization be easy for logographic writing systems too?

日文　　章魚　　怎麼 說?
Japanese octopus how say
How to say octopus in Japanese?

日　　文章 魚 怎麼 說?
Japan essay fish how say

Top Down Tokenization

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a “.”
  - Decides End Of Sentence/Not End Of Sentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

Bottom Up Tokenization - Word Segmentation

- The character is too small a unit, and so algorithms for **word segmentation** are required.
- These can also be useful for word segmentation in the rare situations where word rather than character boundaries are required.
- The standard segmentation algorithms for these languages use **neural sequence models** trained via **supervised machine learning** on **hand-segmented training set.**

Word Normalization , Lemmatization, & Stemming

Word Normalization

- **Word normalization** is the task of putting words/tokens in a standard format.
- **Case folding** - Mapping everything to lowercase folding
  - “Woodchuck” and “woodchuck” are represented **identically**
- Very helpful for generalization in tasks that are not case sensitive
  - **Information Retrieval** or **Speech recognition**.
- Not well for tasks that are case sensitive
  - Machine Translation, Sentiment Analysis
- Example: “US” the country and “us” the pronoun can outweigh the advantage in generalization that case folding would have provided for other words

Stemming

- Normalization technique is to eliminate the small meaning differences of pluralization or possessive endings of words, or even various verb forms.
- This normalization, identifying a common stem among various forms of a word, is called **stemming**.
  - For example, the words **housing** and **houses** share the same stem, house.
- Stemming removes suffixes from words in an attempt to combine words with similar meanings together under their common stem.
- A stem **isn't** required to be a **properly spelled word**, but merely a token, or label, representing several possible spellings of a word.

Lemmatization

- **Lemmatization** is the task of determining that two words have the same root, despite their surface differences.
  - The words **am**, **are**, and **is** have the shared **lemma** *be*
  - The words **dinner** and **dinners** both have the **lemma** *dinner*.
- Lemmatizing each of these forms to the same lemma will let us find all mentions of words
- Lemmatization is a potentially more accurate way to normalize a word than stemming or normalization because it takes into account a word's meaning.
- A lemmatizer uses a knowledge base of word synonyms and word endings to ensure that only words that mean similar things are consolidated into a single token.