Reg Ex CheatSheet Cheat Sheet

by datamansam via cheatography.com/139410/cs/29846/

Regex

(*) indicates that the preceding character can occur 0 or more times.	meo*w	mew, meow, meooow, and meooooooooooooow
? - character can appear either 0 or 1 time	humou?r	humour humor
. and it can match any single character (letter, number, symbol or whitespace) in a piece of text	....	any 9-character text
[] will match any of the characters included within the brackets	con[sc]en[sc]us	consensus, concensus, consencus, and concencus
{} contains the exact quantity	roa{3}r	roaaar
{}n. the quantity range of characters to be matched	roa{3,6}r	roaaar, roaaaar, roaaaaar, or roaaaaaar
|, allows for the matching of either of two subexpressions.	baboons|gorillas	will match the text baboons as well as the text gorillas.
Anchors (hat ^ and dollar sign $) are used in regular expressions to match text at the start and end of a string, respectively.	^Monkeys: my mortal enemy$	will completely match the text Monkeys: my mortal enemy but not match Spider Monkeys: my mortal enemy or Monkeys: my mortal enemy in the wild
[letter-letter] or [n-n]	a range of characters that can be matched	[A-Z]. : match any uppercase letter [a-z]. : match any lowercase letter [0-9]. : match any digit [A-Za-z] : match any uppercase or lowercase letter
Shorthand character classes simplify writing regular expressions	\w represents the regex range [A-Za-z0-9_], \d represents [0-9],	\W represents [$^{\text{A-Za-z0-9\_]}}$ matching any character not included by \w, \D represents $^{[}$0-9] matching any character not included by \d
Negated character set	[^cdh]are	will match the m in mare.
+ ndicates that the preceding character can occur 1 or more times	meo+w	will match meow, meooow, and meooooooooooow, but not match mew

Text Preprocessing

Noise removal	import re result = re.sub(r'[\.\?-\!\,\:\;\"]', '', text)	Removes Punctuation
Tokenization is the text preprocessing task of breaking up text into smaller components of text	from nltk.tokenize import word_tokenize text = "This is a text to tokenize" tokenized = word_tokenize(text)	print(tokenized) # ["This", "is", "a", "text", "to", "tokenize"]
In natural language processing, normalization encompasses many text preprocessing tasks including	stemming, lemmatization,	upper or lowercasing, and stopwords removal.

Text Preprocessing (cont)

Stemming In natural language processing, stemming is the text preprocessing normalization task concerned with bluntly removing word affixes (prefixes and suffixes).	from nltk.stem import Porter-Stemmer tokenized = ["So", "many", "-squids", "are", "-jumping"] stemmer = PorterStemmer() stemmed = [stemmer.stem(t-oken) for token in tokenized]	# ['So', 'mani', 'squid', 'are', 'jump']
Lemmat-ization In natural language processing, lemmatization is the text preprocessing normalization task concerned with bringing words down to their root forms.	from nltk.stem import WordNe-tLemmatizer tokenized = ["So", "many", "-squids", "are", "-jumping"] lemmatizer = WordNetLemma-tizer() lemmatized = [lemmatizer.lem-matize(token) for token in tokenized]	['So', 'many', 'squid', 'be', 'jump']

Text Preprocessing (cont)

stopword removal is the process of removing words from a string that don't provide any inform-ation about the tone of a statement.	from nltk.c-orpus import stopwords # define set of English stopwords stop_words = set(st-opwords.w-ords('eng-lish'))	# remove stopwords from tokens in dataset statement-_no_stop = [word for word in word_tokens if word not in stop_words]
parser. chunk.Reg-expParser	Uses a set of regular expression patterns to specify the behavior of the parser	{<DT|JJ>} # chunk determiners and adjectives

Token = Smaller Component of Text
Stem = Remove prefix and suffix
Lemmatization = Bring down to root
Stopword = Remove meaningless

Lists and Strings

z = 'Natural Language Processing'	z.repl-ace(' ', '\n')	'Natural\nLa-nguage\nProc-essing'
	list(z)	Split text into character tokens
	set(z)	Unique tokens
x = ['Natural', 'Language', 'Toolkit']	x.inse-rt(0, 'Python')	['Language', 'Natural', 'Python', 'Toolkit']