**NLP Documentation.**

A *regular expression* is a special sequence of characters that describe a pattern of text that should be found, or matched, in a string or document. By matching text, we can identify how often and where certain pieces of text occur, as well as have the opportunity to replace or update these pieces of text if needed.

Regular Expressions have a variety of use cases including:

* validating user input in HTML forms
* verifying and parsing text in files, code and applications
* examining test results
* finding keywords in emails and web pages

The simplest text we can match with regular expressions are ***literals***. This is where our regular expression contains the exact text that we want to match.

Do you love baboons and gorillas? You can find either of them with the same regular expression using ***alternation!*** Alternation, performed in regular expressions with the pipe symbol, |, allows us to match either the characters preceding the | OR the characters after the |

***Character sets***, denoted by a pair of brackets [], let us match one character from a series of characters, allowing for matches with incorrect or different spellings.

Cleaning and preparation are crucial for many tasks, and NLP is no exception. ***Text preprocessing*** is usually the first step you’ll take when faced with an NLP task.

There is a LOT you can do here, depending on the formatting you need. Lucky for you, [Regex](https://en.wikipedia.org/wiki/Regular_expression) and NLTK will do most of it for you! Common tasks include:

**Noise removal** — stripping text of formatting (e.g., HTML tags).

**Tokenization** — breaking text into individual words.

**Normalization** — cleaning text data in any other way:

* **Stemming** is a blunt axe to chop off word prefixes and suffixes. “booing” and “booed” become “boo”, but “sing” may become “s” and “sung” would remain “sung.”
* **Lemmatization** is a scalpel to bring words down to their root forms. For example, NLTK’s savvy lemmatizer knows “am” and “are” are related to “be.”
* Other common tasks include lowercasing, [stopwords](https://en.wikipedia.org/wiki/Stop_words" \t "_blank) removal, spelling correction, etc.

***Parsing*** is a stage of NLP concerned with segmenting text based on syntax.

***Part-of-speech tagging (POS tagging)*** identifies parts of speech (verbs, nouns, adjectives, etc.). NLTK can do it faster (and maybe more accurately) than your grammar teacher.

***Named entity recognition (NER)*** helps identify the proper nouns (e.g., “Natalia” or “Berlin”) in a text. This can be a clue as to the topic of the text and NLTK captures many for you.

***Dependency grammar*** trees help you understand the relationship between the words in a sentence. It can be a tedious task for a human, so the Python library spaCy is at your service, even if it isn’t always perfect.

***Language models*** are probabilistic computer models of language.

We build and use these models to figure out the likelihood that a given sound, letter, word, or phrase will be used

One of the most common language models is the unigram model, a statistical language model commonly known as ***bag-of-words***. As its name suggests, bag-of-words does not have much order to its chaos!

Bag-of-words can be an excellent way of looking at language when you want to make predictions concerning topic or sentiment of a text.