Definition - What does Tokenization mean?

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining.

Tokenization is used in computer science, where it plays a large part in the process of lexical analysis.

Tokenization relies mostly on simple heuristics in order to separate tokens by following a few steps:

- Tokens or words are separated by whitespace, punctuation marks or line breaks
- White space or punctuation marks may or may not be included depending on the need
- All characters within contiguous strings are part of the token. Tokens can be made up of all alpha characters, alphanumeric characters or numeric characters only.

Tokens themselves can also be separators. For example, in most programming languages, identifiers can be placed together with arithmetic operators without white spaces. Although it seems that this would appear as a single word or token, the grammar of the language actually considers the mathematical operator (a token) as a separator, so even when multiple tokens are bunched up together, they can still be separated via the mathematical operator.

Tokenization

Where Are Terms From?

- How can we derive terms from documents to be indexed?
  - Collect the documents to be indexed
  - Tokenize the text
  - Linguistic preprocessing of tokens
- Language-dependent
  - Use heuristic methods, user selection, metadata, or machine learning methods to determine the language of the document
  - Some language (e.g., Arabic) may need special sequencing preprocessing

Choosing a Proper Document Unit

- Many possible choices
  - Each file in a folder as a document
  - In an mbox-format UNIX email file, each email within the large file is treated as a document
  - Within an email, each attachment may be treated as a document
- Why does indexing granularity matter?
  - A tradeoff between precision and recall
  - Big granularity often leads to low accuracy – searching for "Kung Fu Panda" may return a book containing "Kung Fu" at the beginning and "Panda" at the end
  - Very small granularity often leads to low recall – searching "Beijing Olympic" may miss the two sentences "I went to Beijing to join my friends. We watched the Olympic games together."

Tokenization

- Given a character sequence and a defined document unit, **tokenization** is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation

Input: Friends, Romans, Countrymen, lend me your ears;

Output: | Friends | Romans | Countrymen | lend | me | your | ears |

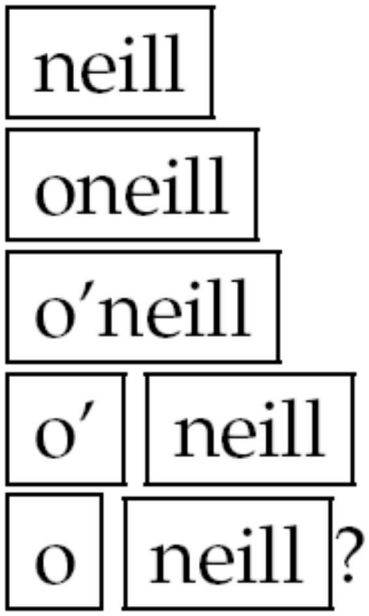
Tokens, Types, and Terms

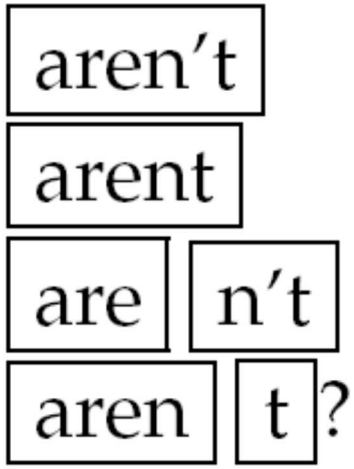
- Text: “to sleep perchance to dream”
- A **token** is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing
  - Examples: “to”, “sleep”, “perchance”, “to”, “dream”
- A **type** is the class of all tokens containing the same character sequence
  - Examples: “to”, “sleep”, “perchance”, “dream”
- A **term** is a (perhaps normalized) type that is included in the IR system’s dictionary
  - Example: “sleep”, “perchance”, “dream”

Apostrophes

- Used for possession and contractions

  Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.





Specific Tokens

- C++, C#
- B-52, B777
- M*A*S*H
- Email addresses (cmpt456@sfu.ca)
- Web URLs (http://www.cs.sfu.ca)
- IP addresses (142.32.48.231)
- Phone number (778-782-3054)
- City names (San Francisco, New York)

Hyphens

- Hyphenation is used in English for
  - Splitting up vowels in words (co-education)
  - Joining nouns as names (Hewlett-Packard)
  - Showing word grouping (the hold-him-back-and-drag-him-away maneuver)
  - Special usage (San Francisco-Los Angeles)
- Splitting on white space may not always be desirable
  - "New York University" should not be returned for query "York University"
  - "lowercase", "lower-case", and "lower case" are equivalent

Word Segmentation

- In some languages (e.g., Chinese), text is written without any spaces between words

  信息检索和Web搜索是一门很有意思的课程。

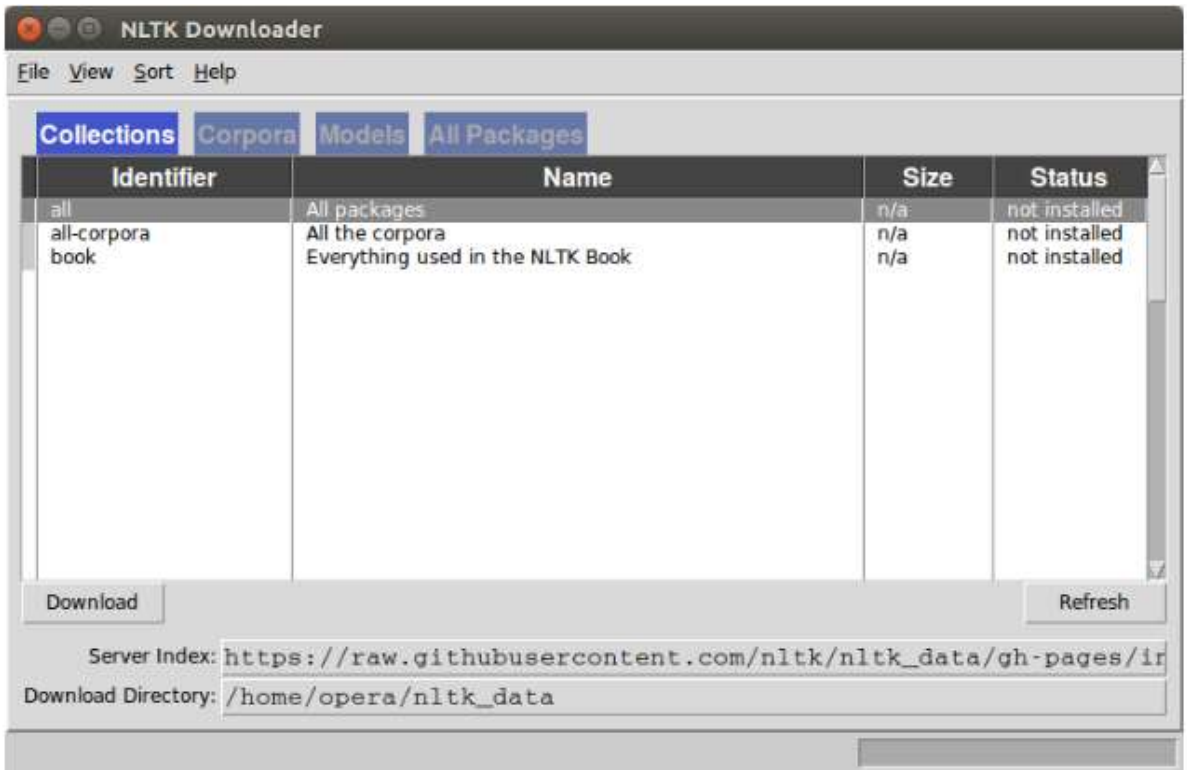
- Word segmentation methods
  - Use a large vocabulary and take the longest vocabulary match
  - Machine learning sequence models (e.g., Markov models, conditional random fields)
  - Character k-grams

*****The following code will help you to tokenize sentences*****

```
import nltk
nltk.download()


from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack a dull boy, all work and no play"
print(word_tokenize(data))
```



Click all and then click download. It will download all the required packages which may take a while, the bar on the bottom shows the progress.

A sentence or data can be split into words using the method **word_tokenize():**

```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack a dull boy, all work and no play"
print(word_tokenize(data))
```

Tokenize words

A sentence or data can be split into words using the method word_tokenize():

```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack a dull boy, all work and no play"
print(word_tokenize(data))
```

This will output:

```
['All', 'work', 'and', 'no', 'play', 'makes', 'jack', 'dull', 'boy', ',', 'all', 'work', 'and', 'no', 'play']
```

All of them are words except the comma. Special characters are treated as separate tokens.

Tokenizing sentences

The same principle can be applied to sentences. Simply change the to sent_tokenize()
We have added two sentences to the variable data:

```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack dull boy. All work and no play makes jack a dull boy."
print(sent_tokenize(data))
```

Outputs:

```
['All work and no play makes jack dull boy.', 'All work and no play makes jack a dull boy.']
```

NLTK and arrays

If you wish to you can store the words and sentences in arrays:

```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack dull boy. All work and no play makes jack a
dull boy."

phrases = sent_tokenize(data)
words = word_tokenize(data)

print(phrases)
print(words)
```