Lab 3: Sentence Segmentation and Word Tokenization

**References:**

1. Introduction to Information Retrieval, by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Cambridge University Press. 2008.
2. Natural Language Processing with Python, by Steven Bird, Ewan Klein and Edward Loper, 2014.

Quick Review

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. Here is an example of tokenization:

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

Output: \framebox{Friends\weestrut} \framebox{Romans\weestrut} \framebox{Countrymen\weestrut} \framebox{lend\weestrut} \framebox{me\weestrut} \framebox{your\weestrut} \framebox{ears\weestrut}

These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. 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. A type is the class of all tokens containing the same character sequence. A term is a (perhaps normalized) type that is included in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in a taxonomy, but in practice in modern IR systems, they are strongly related to the tokens in the document. However, rather than being exactly the tokens that appear in the document, they are usually derived from them by various normalization processes.

Practices

* 1. Sentence Segmentation

Write the following code in Python, run it and discuss the result. Replace the sample text with your own choice and re-run the programme.

from nltk.tokenize import sent\_tokenize

sampleText = "This programme is designed to provide students with knowledge and applied skills in data science, big data analytics and business intelligence. It aims to develop analytical and investigative knowledge and skills using data science tools and techniques, and to enhance data science knowledge and critical interpretation skills. Students will understand the impact of data science upon modern processes and businesses, be able to identify, and implement specific tools, practices, features and techniques to enhance the analysis of data."

Sentences = sent\_tokenize(sampleText)

print("There are ", len(Sentences), "sentences in this text\n")

counter = 0

for sent in Sentences:

counter+=1

print(counter,".",sent,"\n")

* 1. Word tokenization

Write the following code in Python, run it and discuss the result. Replace the sample text with your own choice and re-run the programme.

from nltk.tokenize import word\_tokenize

sampleText = "This programme is designed to provide students with knowledge and applied skills in data science, big data analytics and business intelligence. It aims to develop analytical and investigative knowledge and skills using data science tools and techniques, and to enhance data science knowledge and critical interpretation skills. Students will understand the impact of data science upon modern processes and businesses, be able to identify, and implement specific tools, practices, features and techniques to enhance the analysis of data."

Tokens = word\_tokenize(sampleText)

print("There are ", len(Tokens), "tokens in this text\n")

counter = 0

for w in Tokens:

counter+=1

print(counter,".",w)

*# To print all the tokens*

print(Tokens)

print()

Use nltk.Text() to create a text list from the tokens list.

import nltk

Tokenstext = nltk.Text(Tokens)

print(Tokenstext[0:len(Tokenstext)])

* 1. Accessing Text from the Web

You may be interested in analysing other texts from Project Gutenberg. You can browse the catalogue of 25,000 free online books at <http://www.gutenberg.org/catalog/>, and obtain a URL to an ASCII text file. Although 90% of the texts in Project Gutenberg are in English, it includes material in over 50 other languages, including Catalan, Chinese, Dutch, Finnish, French, German, Italian, Portuguese and Spanish (with more than 100 texts each).

Text number 2554 is an English translation of Crime and Punishment, and we can access it as follows:

from urllib.request import urlopen

from nltk.tokenize import sent\_tokenize, word\_tokenize

url = "http://www.gutenberg.org/files/2554/2554-0.txt"

sampleText = urlopen(url).read().decode('utf8')

len(sampleText)

Repeat Sentence Segmentation and Word tokenization for the above corpora.

* 1. Dealing with HTML

If you can process text on the web in form of HTML document, you can get Python to do the work directly. The first step is the same as before, using urlopen. We pick a BBC News story called “Heart risk link to big families”:

url = "http://news.bbc.co.uk/2/hi/health/2284783.stm"

sampleText = urlopen(url).read().decode('utf8')

Check and discuss the result.

To get text out of HTML we will use a Python library called *BeautifulSoup*, available from the website below:

<http://www.crummy.com/software/BeautifulSoup/>

from bs4 import BeautifulSoup

textTokenx = BeautifulSoup(sampleText).get\_text()

Compare the result with the earlier, and discuss.

* 1. Reading Local Files

In order to read a local file, we need to use Python's built-in open() function, followed by the read() method. Suppose you have a file sample01.txt, you can load its contents like this:

fileHandler = open('sample01.txt')

sampleText = fileHandler.read()

Create a text file using notepad and use the above commands to read the file and process it.

* 1. Removing Stop Words

You may want to remove stop words from the list of tokens. The following code helps to get rid of stop words.

from nltk.tokenize import sent\_tokenize, word\_tokenize

from nltk.corpus import stopwords

fileHandler = open('sample01.txt')

sampleText = fileHandler.read()

Tokens = word\_tokenize(sampleText)

print("There are ", len(Tokens), "words in this text\n")

stopTokens = stopwords.words("english")

filteredTokens = []

for w in Tokens:

if w not in stopTokens :

filteredTokens.append(w)

print("There are ", len(filteredTokens), "words in this text after removing stop words\n")

print(filteredTokens)

Most probably you would also like to strip off punctuation, you can use string.punctuation:

import string

...

stopTokens = stopwords.words("english") + list(string.punctuation)

...

Stop words can be seen via the following codes for all the space delimited languages.

from nltk.corpus import stopwords

stop\_words\_english = set(stopwords.words('english'))

print(stop\_words\_english)

print()

stop\_words\_spanish = set(stopwords.words('spanish'))

print(stop\_words\_spanish)

You can print the entire list of stop words available in ENGLISH as the natural language. Also, can print the frequency of each words in the corpus and possible to print the frequency distribution.

Word cloud can be generated using the wordcloud package from python.

import wordcloud

print("\n", stopTokens)

# word count / bag of words

print(wordtokens.count("skills"))

fdist1 = nltk.FreqDist(filteredTokens)

print("\n",fdist1.most\_common(10))

# Display the generated word cloud

word\_cloud = wordcloud.WordCloud(background\_color='white').generate(text)

plt.figure(figsize = (15, 8), facecolor = None)

plt.imshow(word\_cloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

**Advantages of Word Clouds:**

1. Analysing customer and employee feedback.
2. Identifying new SEO keywords to target.

**Drawbacks of Word Clouds:**

1. Word Clouds are not perfect for every situation.
2. Data should be optimized for context.