ITCS159 Software Lab for Basic Scientific Problem Solving
Lab 13: Introduction to text processing using NLTK

November 8, 2021

Objective

In today's area of Internet and online services, data is generating at incredible speed and amount. Generally, Data analyst, engineer, and scientists are handling relational or tabular data. These tabular data columns have either numerical or categorical data. However, generated data has a variety of structures such as text, image, audio, and video. Online activities such as articles, website text, blog posts, social media posts. Those are unstructured data. Corporate and business need to analyze textual data to understand customer activities, opinion, and feedback to successfully derive their business. To compete with big textual data, text analytics is evolving at a faster rate than ever before.

Text Analytics has lots of applications in today's online world. By analyzing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. Youtube can also analyze and understand peoples viewpoints on a video.

In this tutorial, you are going to cover the following topics:

- Text Analysis Operations using NLTK
- Tokenization
- Stopwords
- Lexicon Normalization such as Stemming and Lemmatization
- POS Tagging

Those are common practices in Natural Language Processing (NLP). NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction.

The Natural Language Toolkit (NLTK) is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, open source,

easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

Note that:

- Each lab consists of a number of practical exercises, all of which are required for marks to be attained. In particular, you should try to accomplish the 3 labelled milestones.
- Attendance is compulsory and will be monitored on Mycourse. If you are unable to attend a session you must inform a tutor in advance.
- You can finish all milestones before calling for marking.

During the marking, your understandings will be tested, and the tutor will not give the marks if you do not display an understanding of the problem and its solution. The tutors are there to help you think through any issues – but they won't write the programs for you!

NLTK installation

The first step is to install NLTK. You can run the Anaconda Prompt (remember to run it as the administrator). To install NLTK on python, at the command prompt, type

```
conda install nltk
```

Before start coding:

Once the installation is completed, open jupyter notebook from command prompt by typing

```
jupyter notebook
```

NLTK can be imported using

```
import nltk
```

Introduction to NLTK

After you imported NLTK, you need to download the required data (e.g., text processing functions, text corpus). It can be done by using the following code:

```
import nltk
nltk.download()
```

However, if you found errors on SSL, please try the following code:

```
import nltk
import ssl

try:
        _create_unverified_https_context = ssl.
            _create_unverified_context
        except AttributeError:
                pass
else:
        ssl._create_default_https_context =
            _create_unverified_https_context

nltk.download()
```

For the third option, we use a proxy server. You should specify the proxy address as follows.

```
nltk.set_proxy('http://proxy.madhidol:8080', ('USERNAME', '
    PASSWORD'))
nltk.download()
```

A new window should open, showing the NLTK Downloader. You don't have to download all packages. That would take few hours. You need to download `Popular packages` by downloading `popular` package in the `Collections` tab.

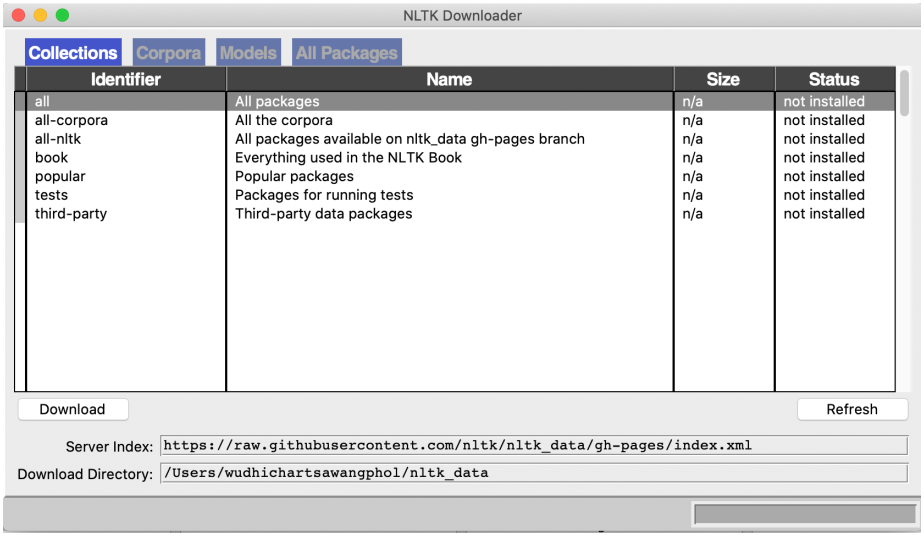

Figure 1: NLTK Downloader

However, if you have a problem with downloading packages using `NLTK downloader`, you can manually install those packages for example:

```
nltk.download('punkt')
```

If you find errors from running the code in the lab practices, please check the recommendation at the end of error messages. They inform what packages you need to install as shown in the Figure 1. You just install using the above command.

```
LookupError:
**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')
```

Figure 2: Error Example

Milestone 1: You have now completed the Milestone 1. Keep going!

Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences. Use the following code to use NLTK for sentence tokenizing.

```
from nltk.tokenize import sent_tokenize

text="""Dogs ,Canis lupus familiaris, are domesticated mammals,
     not natural wild animals. They were originally bred from
    wolves. They have been bred by humans for a long time, and
    were the first animals ever to be domesticated.

Today, some dogs are used as pets, others are used to help
    humans do their work. They are a popular pet because they
    are usually playful, friendly, loyal and listen to humans.
    Thirty million dogs in the United States are registered as
    pets. Dogs eat both meat and vegetables, often mixed
    together and sold in stores as dog food. Dogs often have
    jobs, including as police dogs, army dogs, assistance dogs,
    fire dogs, messenger dogs, hunting dogs, herding dogs, or
    rescue dogs.

They are sometimes called "canines" from the Latin word for dog
     - canis. Sometimes people also use "dog" to describe other
    canids, such as wolves. A baby dog is called a pup or puppy.
     A dog is called a puppy until it is about one year old.

Dogs are sometimes referred to as "man's best friend" because
    they are kept as domestic pets and are usually loyal and
    like being around humans."""

tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

How many sentences in the paragraph?

What is the third sentences in the paragraph?

Word Tokenization

Word tokenizer breaks text paragraph into words. Use the following code to use NLTK for word tokenizing.

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

How many words in the paragraph?

How many words in the first sentence?

How many words in the last sentence?

Do you find repeat words?

Frequency Distribution

Let's calculate the frequency distribution of those words using Python NLTK. There is a function in NLTK called `FreqDist()` that does the job. Try the following code:

```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

What is the output from `print(fdist)`?

Let's try to show `fdist` on the notebook as the following and try to interpret the result.

```
fdist
```

How about the most common words? Use the following command to show the common words

```
fdist.most_common(10)
```

What is the top ten most common word?

Stopwords

Any piece of text which is not relevant to the context of the data and the end-output can be specified as the noise. For example – language stopwords (commonly used words of a language – is, am, the, of, in etc), URLs or links,

social media entities (mentions, hashtags), punctuations and industry specific words. This step deals with removal of all types of noisy entities present in the text.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

Let create our stopwords as the following:

```
my_stopwords = ["is", "a", "this", ","]
```

However, NLTK also provide a set of common stopwords. Let explore what those stopwords are.

```
from nltk.corpus import stopwords

nltk_stop_words=set(stopwords.words("english"))
print(nltk_stop_words)
```

What are those stopwords provided by NLTK?

Removing Stopwords

Next, you are going to use the stopwords that you prepared to remove noise from our text. Let try the following code:

```
filtered_words=[]
for w in tokenized_word:
        if w not in my_stopwords:
                filtered_words.append(w)

print("Tokenized Sentence:",tokenized_word)
print("\n")
print("Filterd Sentence:",filtered_words)
```

Can you apply the stopwords provided by NLTK (`nltk_stop_words`) to your text?

Milestone 2: You have now completed the Milestone 2. Keep going!

Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

The most common lexicon normalization practices are

- Stemming
- Lemmatization

Stemming

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes ("ing", "ly", "es", "s" etc). For example, connection, connected, connecting word reduce to a common word "connect".

```
from nltk.stem import PorterStemmer
stem = PorterStemmer()

word = "connection"
stem.stem(word)
```

What is the output? Can you try some other words such as derivational, multiplying?

Now let apply stemming on our text using the following code

```
from nltk.stem import PorterStemmer

stem = PorterStemmer()

stemmed_words=[]
for w in filtered_words:
        stemmed_words.append(stem.stem(w))

print("Filtered Words:",filtered_words)
print("\n")
print("Stemmed Words:",stemmed_words)
```

What is the output?

Lemmatization

Lemmatization, on the other hand, is an organized and step by step procedure of obtaining the root form of the word, it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations). Let's try the code below:

```
from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()
word = "connecting"

lem.lemmatize(word, pos="v")
```

Note that `pos` is a part of speech parameter. Let's change `pos` to see the result.

```
from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()
word = "flying"

print(lem.lemmatize(word, pos="v"))
print(lem.lemmatize(word, pos="n"))
```

What is the different between using v and n?

POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

Let's explore how POS tagging work using the code below

```
sent = "I am studying at ICT"

tokens=nltk.word_tokenize(sent)
print(tokens)

nltk.pos_tag(tokens)
```

Can you try to interpret the result?

Milestone 3: You have now completed the Milestone 3. Raise YOUR HAND NOW!