CSCI 4152/6509
Natural Language Processing

Lab 5:

Python NLTK Tutorial 1

Lab Instructor: Sigma Jahan and Tymon Wranik-Lohrenz

Faculty of Computer Science

Dalhousie University

Lab Overview

- Introduction to Natural Language Toolkit (NLTK)
- Python quick overview;
- Lexical analysis: Word and text tokenizer;
- n-gram and collocations;
- NLTK corpora;
- Naïve Bayes classifier with NLTK.

Python Overview

- Basic syntax: Identifiers
- Lines and Indentation:

Indentation used indicate blocks of code

- Quotation: single (’), double (") and triple (’’’ or """) quotes Example:

```
word = ’word’
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
               made up of multiple lines and sentences."""
```

- Data types, assiging and deleting values

Lists

```
print(len([1, 2, 3]))        # 3 - length
print([1, 2, 3] + [4, 5, 6]) # [1, 2, 3, 4, 5, 6] - concatenation
print(['Hi!'] * 4)           # ['Hi!', 'Hi!', 'Hi!', 'Hi!']
                             #  - repetition
print(3 in [1, 2, 3])        # True - checks membership
for x in [1, 2, 3]: print(x) # 1 2 3 - iteration
```

- Some useful built-in functions for lists: `max`, `min`, `cmp`, `len`, `list` (converts tuple to list), etc.
- Some of the list-specific functions are `list.append`, `list.extend`, `list.count`, etc.

Tuples

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7);
print(tup1[0])      # prints: physics
print(tup2[1:5])    # prints: [2, 3, 4, 5]
```

Basic tuple operations are same as with lists: length, concatenation, repetition, membership and iteration.

Dictionaries

```
dict = {'Name':'Zara', 'Age':7, 'Class':'First'}
dict['Age'] = 8                  # update existing entry
dict['School'] = "DPS School" # Add new entry
del dict['School]                # Delete existing entry
```

List comprehension.

- Building sequences from other sequences
- Examples:

```
a_list = [1, 2, 9, 3, 0, 4]
squared_ints = [e**2 for e in a_list]

print(squared_ints)         # [ 1, 4, 81, 9, 0, 16 ]
```

This is same as:

```
a_list = [1, 2, 9, 3, 0, 4]
squared_ints = []
for e in a_list:
    squared_ints.append(e**2)

print(squared_ints)         # [ 1, 4, 81, 9, 0, 16 ]
```

Now, let us see an example with the ‘if’ statement. The example shows how to filter out non integer types from mixed list and apply operations.

```
a_list = [1, ’4’, 9, ’a’, 0, 4]
squared_ints = [ e**2 for e in a_list if type(e) is int ]

print(squared_ints)        # [ 1, 81, 0, 16 ]
```

However, if you want to include an ‘if-else’ statement, the arrangement looks a bit different.

```
a_list = [1, ’4’, 9, ’a’, 0, 4]
squared_ints = [ e**2 if type(e) is int else ’x’ for e in a_list]

print(squared_ints)        # [1, ’x’, 81, ’x’, 0, 16]
```

You can also generate dictionary using list comprehension:

```
a_list = ["I", "am", "a", "data", "scientist"]
science_list = { e:i for i, e in enumerate(a_list) }

print(science_list)   # {'I': 0, 'am': 1, 'a': 2, 'data': 3,
                      #  'scientist': 4}
```

… or list of tuples:

```
a_list = ["I", "am", "a", "data", "scientist"]
science_list = [ (e,i) for i, e in enumerate(a_list) ]

print(science_list)   # [('I', 0), ('am', 1), ('a', 2),
                      # ('data', 3), ('scientist', 4)]
```

String Handling

Examples with string operations:

```
str = ’Hello World!’
print(str)          # Prints complete string
print(str[0])       # Prints first character of the string
print(str[2:5])     # Prints characters starting from 3rd to 5th
print(str[2:])      # Prints string starting from 3rd character
print(str*2)        # Prints string two times
print(str + "TEST") # Prints concatenated string
```

Other useful functions include `join`, `split`, `count`, `capitalize`, `strip`, `upper`, `lower`, etc.

Example of string formatting:

```
print("My name is %s and age is %d!" % (’Zara’,21))
```

IO Handling

- Python 2 uses the built-in function `raw_input` to read the standard input
- In Python 3 this function is renamed to `input`
- We will use Python 3 in this lab

```
str = input("Enter your input: ")
print("Received input is : ", str)
```

File Opening

To handle files in Python, you can use function `open`. Syntax:

```
file object = open(file_name [, access_mode][, buffering])
```

One of the useful packages for handling tsv and csv files is `csv` library.

Functions

An example how to define a function in Python:

```
def functionname(parameters):
  "function_docstring"
  function_suite
  return [expression]
```

Running your code on `timberlea`

- One way: `python mypscript.py`
- or: `./mypyscript.py`

where `mypscript.py` looks like:

```
#!/local/bin/python

print("Hello World!")
```

Step 1. Logging in to server `timberlea`

- Login to server `timberlea`
- Change directory to `csci4152` or `csci6509`
- Create directory `lab5` and `cd` to it:

```
mkdir lab5
cd lab5
```

Step 2: Python list, tuple and dictionary example

- Create a file called `lab5-list_merge.py` following instructions in the notes
- Submit the file `lab5-list_merge.py` using the `submit-nlp` command

Step 3: Lexical Analysis: tokenization

- Word tokenization: using method `word_tokenize`
- Sentence tokenization: using method `sent_tokenize`
- Storing words and sentences in lists

Step 4. Stop-word Removal

```
#!/local/bin/python
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords  # We imported auxiliary corpus
                                   # provided with NLTK

data = ("All work and no play makes jack dull boy.\n"+
        "All work and no play makes jack a dull boy.")

stopWords = set(stopwords.words(’english’))  # a set of English
words = word_tokenize(data.lower())          #        stopwords
wordsFiltered = []

for w in words:
    if w not in stopWords:
        wordsFiltered.append(w)

print(len(stopWords))  # Print the number of stopwords
print(stopWords)       # Print the stopwords
print(wordsFiltered)   # Print the filtered text
```

Submit `stop_word_removal.py`

- **Note:** If you get an error message, you may need to download the resource `stopwords`
- Submit the previous code as the file `lab5-stop_word_removal.py` using the `submit-nlp` command

Step 5. Stemming
To write an example of a program using stemming, we start by defining some words:

```
words = ["game","gaming","gamed","games"]
```

We import the Porter stemmer module:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
```

and stem the words in the list as follows, where we put all components together:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

words = ["game","gaming","gamed","games"]
ps = PorterStemmer()

for word in words:
    print(ps.stem(word))
```

You can do word stemming for sentences too; we just need to tokenize them first:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

sentence = "gaming, the gamers play games"
words = word_tokenize(sentence)

for word in words:
    print(word + ":" + ps.stem(word))
```

There are more stemming algorithms, but the Porter stemmer is the most popular.

Step 6. N-grams

Word n-grams

```
from nltk import ngrams
sentence = "This is my sentence and I want to ngramize it."
n = 6
w_6grams = ngrams(sentence.split(), n)
for grams in w_6grams:
  print(grams)
```

Character n-grams

```
from nltk import ngrams
sentence = "This is my sentence and I want to ngramize it."
n = 6
c_6grams = ngrams(sentence, n)
for grams in c_6grams:
  print(’’.join(grams))
```

Step 7. Exploring Corpora

- Let us explore some text stats

```
#!/local/bin/python

from nltk import FreqDist
from nltk.tokenize import word_tokenize

data = ("All work and no play makes jack dull boy.\n"+
        "All work and no play makes jack a dull boy.")
words = word_tokenize(data)

fdist1 = FreqDist(words)

print(fdist1.most_common(2)) # Prints two most common tokens
print(fdist1.hapaxes())      # Prints tokens with frequency 1
```

Fill in the comments with answers:

```
# lab5-explore_corpus.py
from nltk.corpus import gutenberg
from nltk import FreqDist

# Count each token in austen-persuasion.txt of the Gutenberg collection
list_of_words = gutenberg.words("austen-persuasion.txt")
fd = FreqDist(list_of_words) # Frequency distribution object

print("Total number of tokens: " + str(fd.N()))  # <insert_comment_how_many>
print("Number of unique tokens: " + str(fd.B())) # <insert_comment_how_many>
print("Top 10 tokens:")                          # <insert_comment_which_is_3rd>
for token, freq in fd.most_common(10):
  print(token + "\t" + str(freq))
```

To find out more about `FreqDist` refer to `http://www.nltk.org/book/ch01.html` section 3.1.

- Submit `lab5-explore_corpus.py`

Step 8. Document Classification

```
#!/local/bin/python

from nltk import FreqDist, NaiveBayesClassifier
from nltk.corpus import movie_reviews
from nltk.classify import accuracy
import random

documents = [(list(movie_reviews.words(fileid)), category)
              for category in movie_reviews.categories()
              for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)     # This line shuffles the order of the documents

all_words = FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features[’contains({})’.format(word)] = (word in document_words)
    return features

featuresets = [(document_features(d), c) for (d,c) in documents]
```

```
train_set, test_set = featuresets[100:], featuresets[:100] # Split
                                               # data to train and test set
classifier = NaiveBayesClassifier.train(train_set)

print(accuracy(classifier, test_set))

# <answer_area>
# <answer_area>
# <answer_area>
```

- Submit the file `lab5-movie_rev_classifier.py` using the `submit-nlp` command

This is the end of Lab 5.