DATA SCIENCE WITH PYTHON

CS 498 (SPECIAL TOPICS IN COMPUTER SCIENCE)

TEXT MINING & NLTK

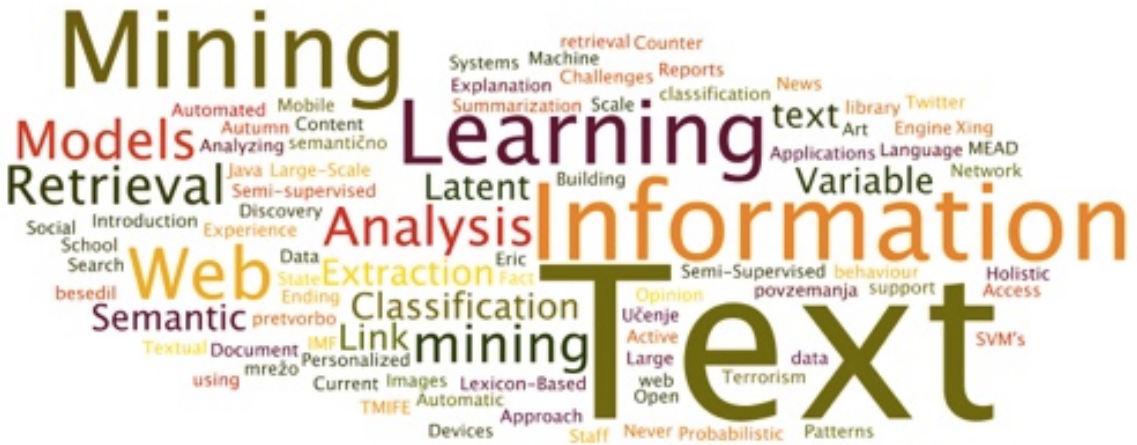
Dr. Malak Abdullah

Computer Science Department

Jordan University of Science and Technology

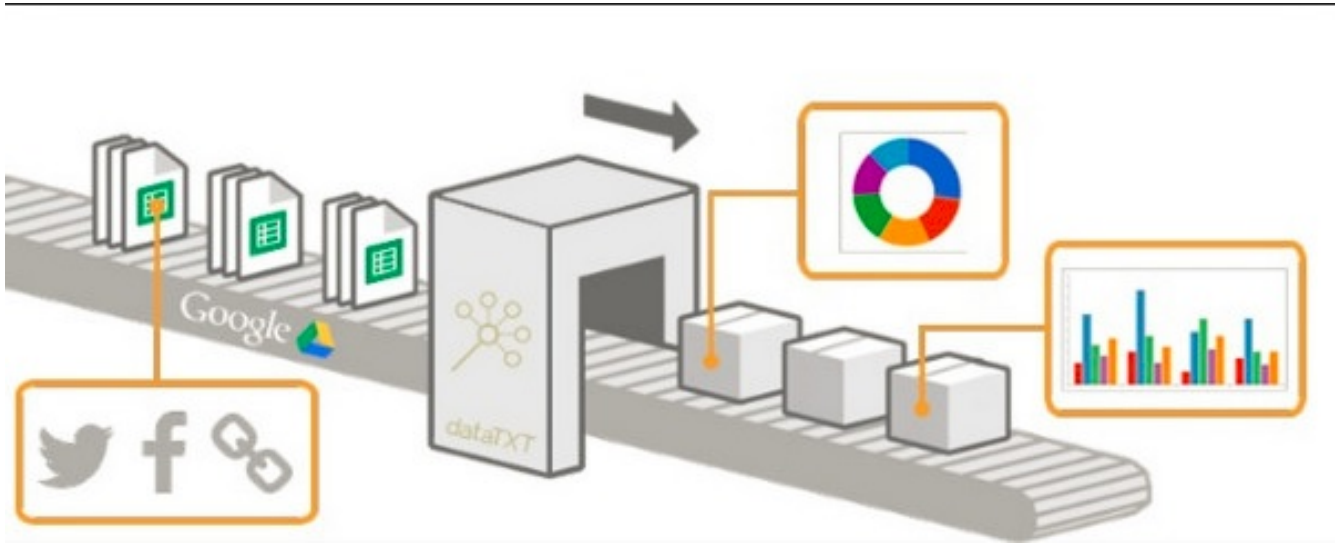
What Is Text Mining?

- Also known as Text Data Mining
- Process of examining large collections of *unstructured* textual resources in order to generate new information, typically using specialized computer software



Why Do We Use Text Mining?

- Turn text into data for analysis
- Generate new information

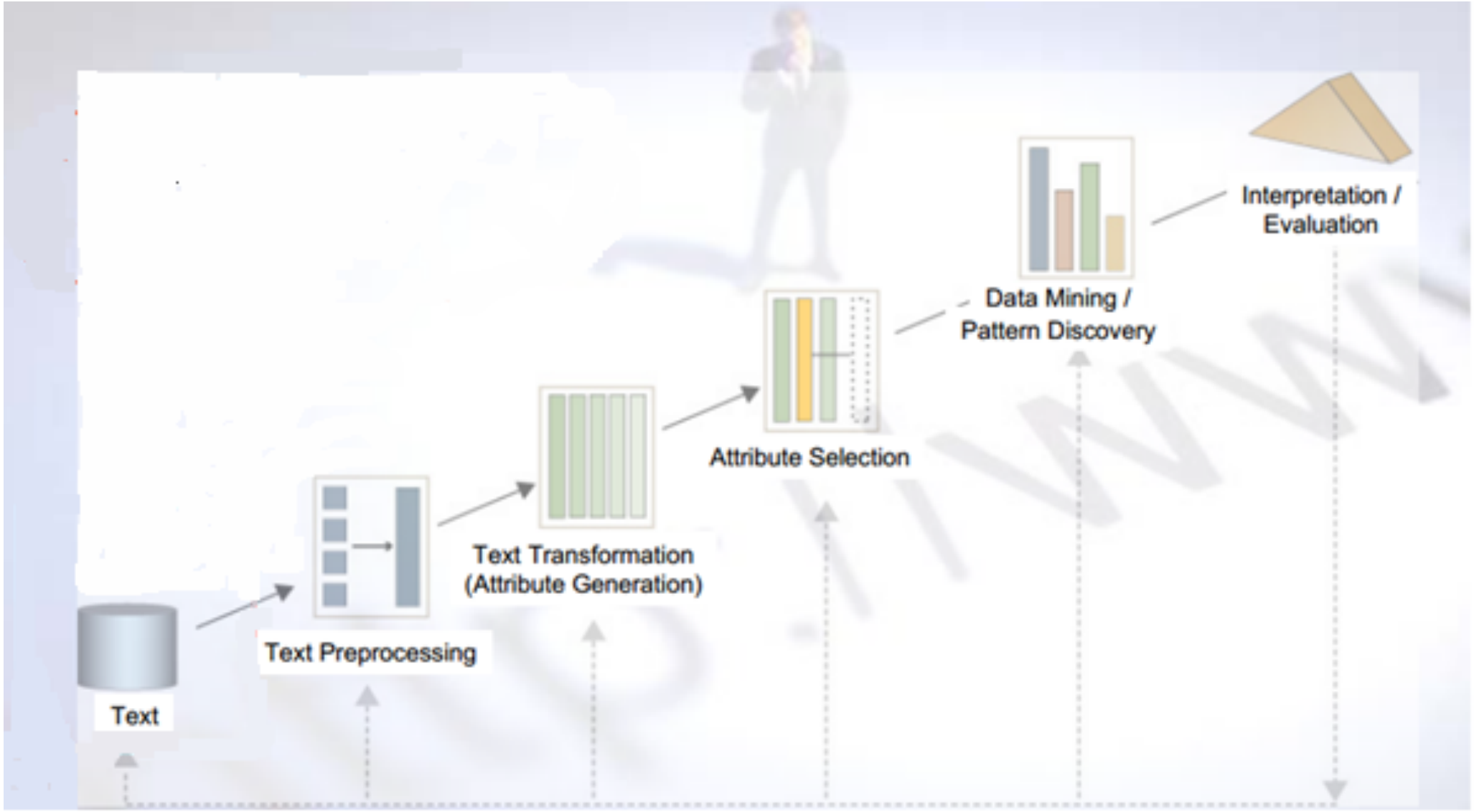


Where Do We See It Being Used?

Applications

- Enterprise Business Intelligence
- Healthcare/Medical Records
- National Security
- Scientific Discovery
- Sentiment Analysis Tools
- Natural Language Service
- Publishing
- Automated Ad Placement
- Information Access
- Social Media Monitoring

Text Mining Process



Text

- Collect large volume of textual data
- Text Characteristics:
  - High dimensionality w/ tens of thousands of words
  - Noisy data
    - Erroneous data or misleading data
  - Unstructured text
    - Written resources, chat room conversations, or normal speech
  - Ambiguity
    - Word ambiguity or sentence ambiguity

Text Preprocessing

- Text Cleanup
  - Normalize texts converted from binary formats (programs, media, images, and most compressed files)
  - Deal with tables, figures, and formulas
- Tokenization
  - Process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens

Text Mining In Social Media

- People use social media to communicate
- Social media provides rich information of human interaction and collective behavior
- Traditional Media vs. Modern Social Media
- Information in most social media sites are stored in text format
- Text Mining can help deal with textual data in social media for research

Natural Language Toolkit (NLTK)

- A suite of Python libraries for symbolic and statistical natural language programming
  - Developed at the University of Pennsylvania
- Developed to be a teaching tool and a platform for research NLP prototypes
  - Data types are packaged as classes
  - Goal of code is to be clear, rather than fastest performance
- Online book: http://www.nltk.org/book/
  - Authors: Edward Loper, Ewan Kline and Steven Bird

Introduction to NLTK

- NLTK provides:
  - Basic classes for representing data relevant to Natural Language Processing.
  - Standard interfaces for performing NLP tasks such as tokenization, tagging and parsing
  - Standard implementation of each task which can be combined to solve complex problems

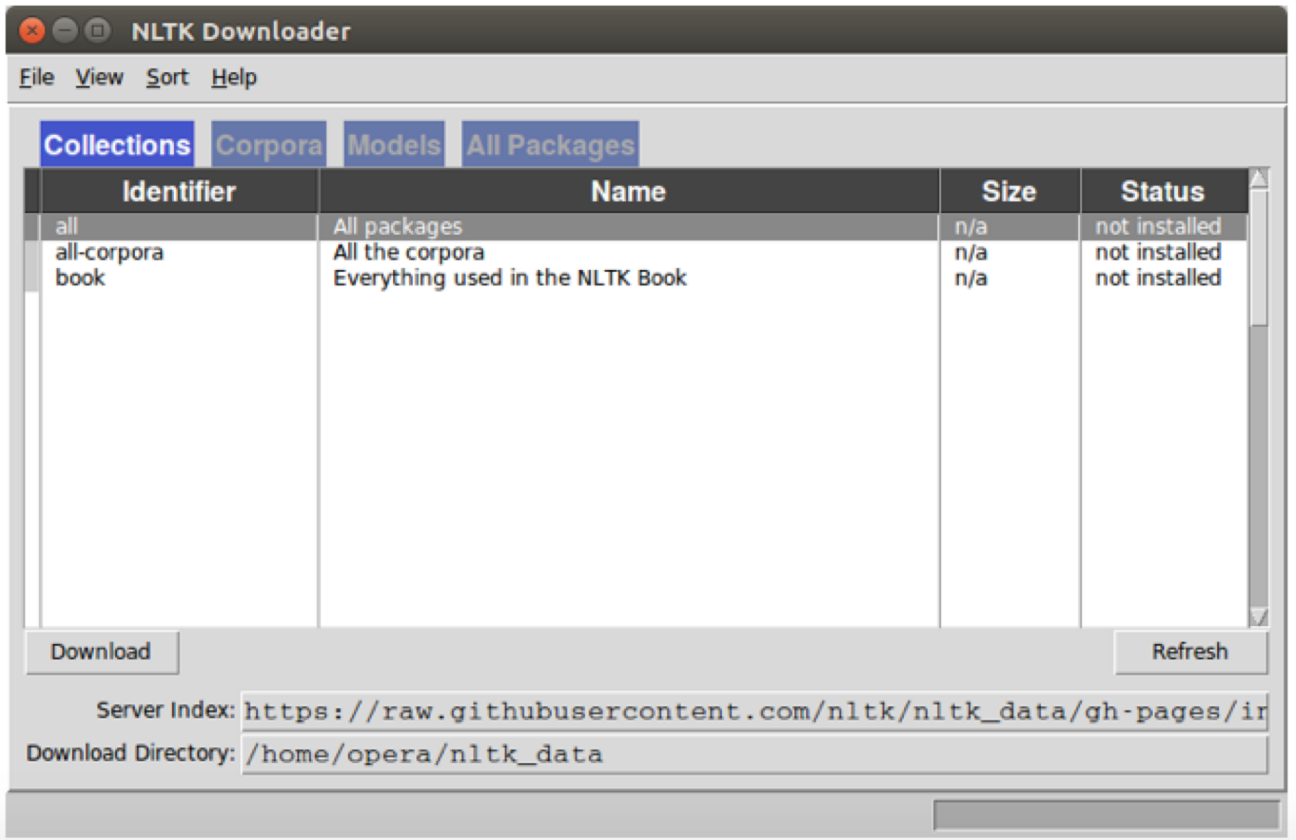
Install NLTK with Python 3.x using:

```
sudo pip3 install nltk
```

Installation is not complete after these commands. Open python and type:

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

A graphical interface will be presented:



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', '
```

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

TOKENIZING SENTENCES

propython.py

```
1
2  import nltk
3
4
5  sentence = "At eight o'clock on Thursday morning Ali didn't feel very good. He bought a panadol with $5.28."
6  tokens=nltk.word_tokenize(sentence)
7  print (tokens)
8
9  print ("*"*30)
10
11 from nltk.tokenize import sent_tokenize, word_tokenize
12 print (sent_tokenize(sentence))
13 print ("*"*30)
14 x= [word_tokenize(t) for t in sent_tokenize(sentence)]
15 print (x)
16
```

```
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Ali', 'did', "n't", 'feel', 'very', 'good', '.', 'He', 'bought', 'a', 'panadol',
'with', '$', '5.28', '.']
******************************
["At eight o'clock on Thursday morning Ali didn't feel very good.", 'He bought a panadol with $5.28.']
******************************
[['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Ali', 'did', "n't", 'feel', 'very', 'good', '.'], ['He', 'bought', 'a',
'panadol', 'with', '$', '5.28', '.']]
******************************
```

REMOVING STOP WORDS

propython.py

```
1
2  import nltk
3
4  sentence = "At eight o'clock on Thursday morning Ali didn't feel very good. He bought a panadol with $5.28."
5  tokens=nltk.word_tokenize(sentence)
6  print (tokens)
7  print ("*"*30)
8
9  from nltk.corpus import stopwords
10 stopWords = set(stopwords.words('english'))
11 wordsFiltered = []
12
13 for w in tokens:
14     if w not in stopWords:
15         wordsFiltered.append(w)
16
17 print(wordsFiltered)
18
```

```
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Ali', 'did', "n't", 'feel', 'very', 'good', '.', 'He', 'bought', 'a',
'panadol', 'with', '$', '5.28', '.']
******************************
['At', 'eight', "o'clock", 'Thursday', 'morning', 'Ali', "n't", 'feel', 'good', '.', 'He', 'bought', 'panadol', '$', '5.28',
'.']
[Finished in 1.6s]
```

STEMMING

- A word stem is part of a word. It is sort of a normalization idea, but linguistic.
  For example, the stem of the word waiting is wait.

NLTK – stemming

Start by defining some words:

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

We import the module:

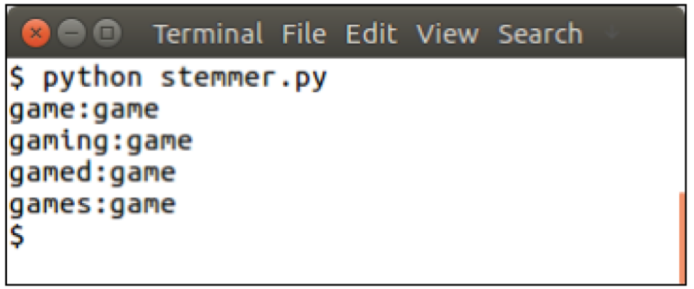
```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
```

And stem the words in the list using:

```
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))
```



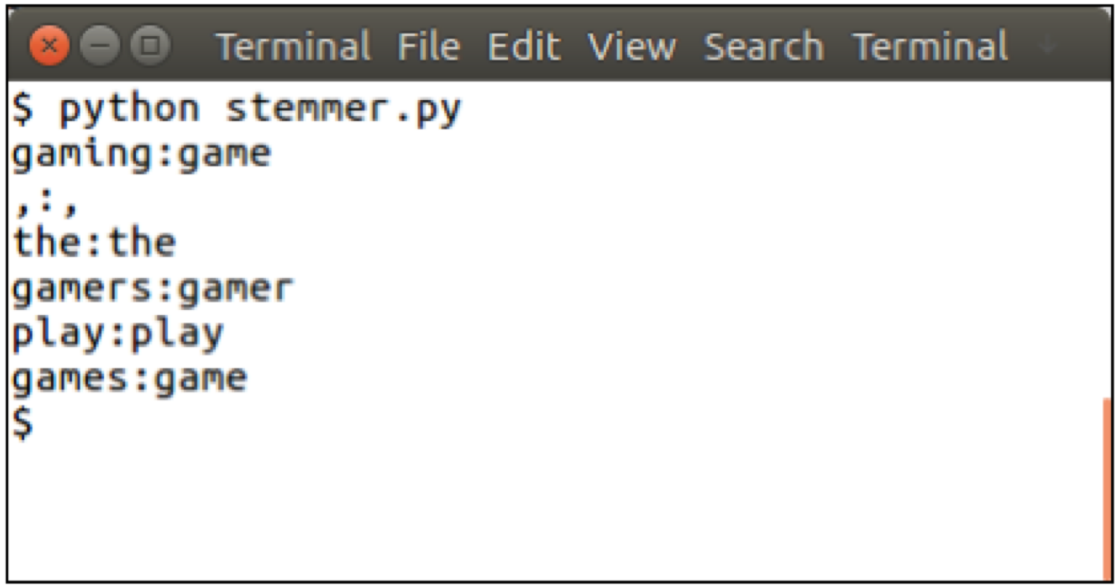
nltk word stem example

```python
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))
```



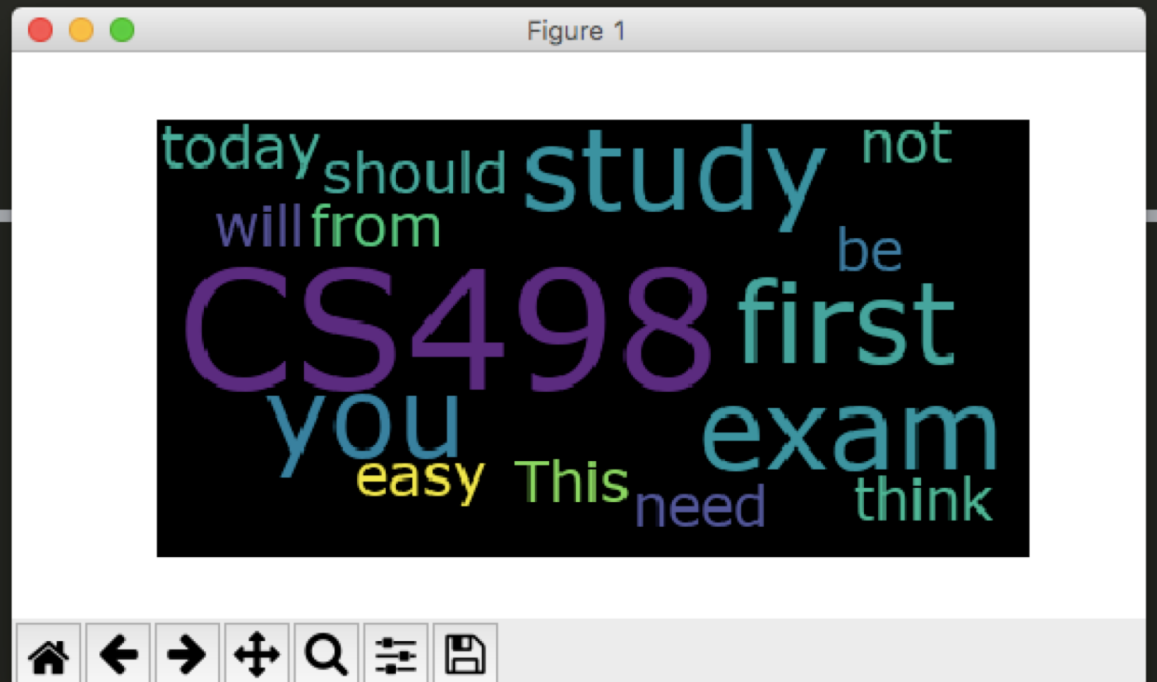
Stemming with NLTK

There are more stemming algorithms, but Porter (PorterStemer) is the most popular.

WORDCLOUD

propython.py

```
1
2  import matplotlib.pyplot as plt
3  from wordcloud import WordCloud, STOPWORDS
4
5  text = 'This is CS498, I think you should study for your first exam, CS498 first exam will not be easy, you need to study CS498 from today'
6
7  def generate_wordcloud(text): # optionally add: stopwords=STOPWORDS and change the arg below
8      wordcloud = WordCloud(font_path='/Library/Fonts/Verdana.ttf',
9                            relative_scaling = 1.0,
10                           stopwords = {'to', 'of' , 'for', 'your', 'I', 'is'} # set or space-separated string
11                           ).generate(text)
12     plt.imshow(wordcloud)
13     plt.axis("off")
14     plt.show()
15
16 generate_wordcloud(text)
```



ASSIGNMENT 3 (DUE DATE 29-10-2018 AT 23:55)

- Talk about JUST university and your favorite course/s with more than 500 words!
- Write a python code that reads this file and build your own wordcloud!
- Upload the text file, the wordcloud and the python code.
- I will collect all your files and build a big cloud to see which course is your favorite course! ☺
- Your files will not be published, but I will publish the big wordcloud!
- Partial credit for your creativity!

- Hint:
  - Remove the stop words
  - Try to use the abbreviation for the course (ex. CS101, CS112…etc)