In [1]:

```
%%html
<style>
table {display: block;}
td {
  font-size: 20px
}
.rendered_html { font-size: 20px; }
*{ line-height: 200%; }
</style>
```

Welcome to the Natural Language Processing and the Web WS23/24 course

Dr. Seid Muhie Yimam

- Email: seid.muhi.yimam@uni-hamburg.de (mailto:seid.muhi.yimam@uni-hamburg.de)
- Office: Informatikum, F-426
- Tel: 040 / 42883 - 2383

Saba Anwar

- Email saba.anwar@uni-hamburg.de (mailto:saba.anwar@uni-hamburg.de)

- Office: Informatikum, F-415
- Tel: 040 / 42883 - 2418

Practice Class

When:

- Group-A: Wednesday 14:15 -- 15:45 => **F-534**
- Group-B: Wednesday 14:15 -- 15:45 => **C-221**
- Group-C: Wednesday 16:15 -- 17:45 => **D-125/129**

Who:

1. Dr. Seid Muhie Yimam
2. Saba Anwar

Exercises

- Exercises (can be done in groups of 2-3) should be submitted before the next tutorial
- Machine learning and final projects will have separate deadlines
- Machine learning projects and final projects can be done in groups

---

To pass the class:

- Obtain at least 50% in the exercise points and Machine learning projects **AND**
- Obtain at least 50% in the final project points

- You must use your own computer to do the exercises and projects
- All exercises will use Python as programming language with different NLP libraries(example NLTK, spaCy), visualization (example mtplotlib or seaborn) , and machine learning (example scikit learn, fast.ai and PyTorch) frameworks

Course Outline

Week	Topic	Lecturer	When
1 - 3	NLTK and different NLP processing components	Seid/Saba	Oct 18 -- Nov 1
4	Introductions to ML using Scikit learn and Feature engineering for ML projects	Seid/Saba	Nov 8, (Distribute ML Project)
5	Deep learning using PyTorch	Seid/Saba	Nov 15
6-7	Large Language Models using Transformers	Seid/Saba	Nov 22-29
8	ML project presentation	Seid/Saba	Dec 6
9	Distribute Project idea	Seid/Saba	Dec 13

Week	Topic	Lecturer	When
10	Project proposal presentation/discussion	Seid/Saba	Dec 20
10 -	Project development consultation	Seid/Saba	Dec 20 -- Jan 24

How to install Python

The Practice class will be conducted using Python 3. We will rely on Anaconda3 (https://www.anaconda.com/distribution/ (https://www.anaconda.com/distribution/)), specifically version 3.7+, a Python (and R) distribution which include most of the NLP and machine learning components. It also includes Anaconda Navigator, which is desktop graphical user interface that allows to launch different applications (Jupyter notebook, Spider GUI, GraphViz...) easily.

How to install Anaconda

Choose Python 3.7+ and install it to a location you prefer. Currently, Anconda installs around 200 packages, which include NLTK and the scikit learn machine learning packages. See here (http://docs.anaconda.com/anaconda) for the list of included packages.

Accessing Jupyter Notebook

For most of the practice classes, we will use the Jupyter Notebook web application to run and demonstrate examples. Except in the case of developing complex assignments and projects, you will also use Jupyter notebook to submit your assignments and in-class exercises.

How to open Jupyter Notebook:

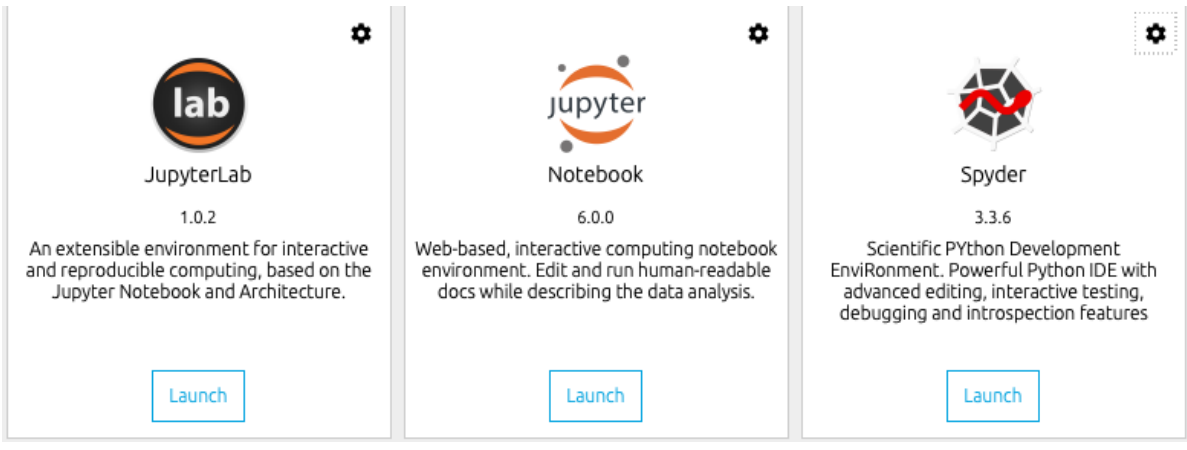
Using Anaconda Navigation

To open Jupyter Notebook using Anaconda navigator:

Once Anaconda is installed, open Anaconda Navigator from your program/application lists for MacOS and Windows.

For Linux, open a terminal and type `anaconda-navigator`

Then you can choose either JupyterLab or Jupyter Notebook to open Jupyter notebook. You can also open it via Spyder but you need to install a notebook plugin



Open Juypter using command line

From the command line, type:

```
jupyter notebook
```

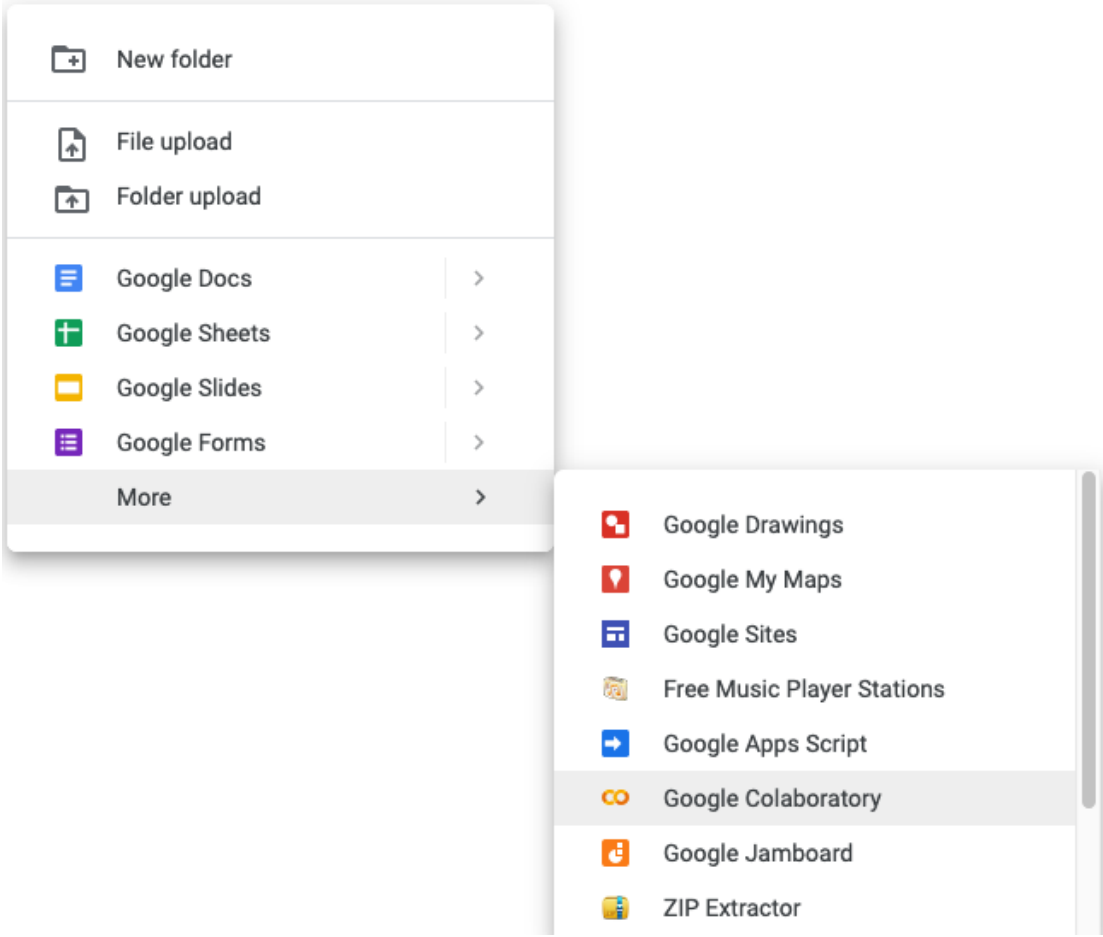
This will open the Jupyter Notebook in the

Google Colab

Google provides a free cloud service based on Jupyter Notebooks that support free GPU, allows developing deep learning applications using PyTorch, TensorFlow, Keras, and so on. Using Colab, you can share your Jupyter notebooks, mount your Google Drive, etc.

Creating a folder for your notebooks

Create a folder in your Google Drive and create Google Colaboratory



Or you can directly go to Colab: https://colab.research.google.com/ (https://colab.research.google.com/) and create a new notebook.

From there you can create a new notebook and set up your environment, such as selecting a GPU.

Mounting your Google Drive

Mounting your Google Drive allows you to persist models, files, and logs on your Google Drive folder as the Colab session is limited.

Once in Colab, you can run the following code to mount to your google drive

```
from google.colab import drive
drive.mount('/content/gdrive')
```

How to open the associated notebooks

1. From Moodle, **unzip** the attachment ch1.zip
2. Once Jupyter notebook is open, navigate to the extracted folder and open the file ch1.ipynb

Using UHH JupyterHub

One more option to use is UHH server with pre-installed Python, jupyter notebook and many more languages

use your stine ID to login

https://code.min.uni-hamburg.de/hub (https://code.min.uni-hamburg.de/hub)

Notebook cheatsheet

See here (https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf_bw/) to get the basic shortcuts and commands to execute a Jupyter notebook cell or display information

Example: Inside a cell, The following are important shortcuts:

1. `shift-Tab-Tab` -- Show documentation of methods or classes
2. `shift-Enter` --- Execute cell and move/create the next cell
3. `Ctrl-Enter` -- Execute a cell and remain the focus on the same cell

Introduction *To* Python

Python preprocessing and NLP pipeline

In this practice class, we will install and discuss two packages/libraries that are important for preprocessing and NLP pipeline, namely NLTK and spaCy. We will use these two libraries extensively for the practice classes and assignments throughout the course.

Installing NLTK components

NLTK is installed automatically with the Anaconda distribution. However, different modules, datasets, and packages are left for the user to download on demand. Try running the following code to install some or all the required resources. If you already have

the required resources, the program should run without any problem. Otherwise, it will give an error with hints on how to install the missing resources.

Example of installing missed resources:

```
import nltk
```

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

In [2]:
```
1 import nltk
2 nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /Users/anwar/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[2]: True

If you want to choose from a graphical user interface, run the following from the command line

```
nlt.download()
```

Then you can choose what you like to download. You can download everything but it will take more time and storage.

To install all the packages/resources, run the following

```
nltk.download('all')
```

In [3]:

```
import nltk
#Download specific resources
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
# If you want to download everything from nltk, uncomment the follow
# nltk.download('all')
```

```
[nltk_data] Downloading package gutenberg to /Users/anwar/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to /Users/anwar/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to /Users/anwar/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package nps_chat to /Users/anwar/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package webtext to /Users/anwar/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package treebank to /Users/anwar/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
```

Out[3]:

```
True
```

The NLTK resource contains dataset that are annotated for different linguistic annotation tasks such as POS tagging, dependency parsing, NER...

In [4]:

```
# Show all books from nltk
from nltk.book import *
#the books are available as text1---text9
print (text1.name)
print("----")
print (text1.concordance("monstrous"))
print ("----")
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
Moby Dick by Herman Melville 1851
----
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towa
rds us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork
we have r
ll over with a heathenish array of monstrous clubs and spears . Some w
ere thick
d as you gazed , and wondered what monstrous cannibal and savage could
ever hav
that has survived the flood ; most monstrous and most mountainous ! Th
at Himmal
they might scout at Moby Dick as a monstrous fable , or still worse an
d more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I sh
all ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am
strongly
ere to enter upon those still more monstrous stories of them which are
to be fo
ght have been rummaged out of this monstrous cabinet there is no telli
ng . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast
up dead u
None
----
```

Individual items can be downloaded from NLTK corpus library

For example, the brown corpus can be downloaded and imported as follows

```
nltk.download('brown')
```

```
from nltk.corpus import brown
```

The brown corpus includes different genres of texts such as fictions, reviews, hobbies, news, and so on.

To view the categories:

```
brown.categories()
```

In [5]:
```
1 nltk.download('brown')
2 from nltk.corpus import brown
3 brown.categories()
```

```
[nltk_data] Downloading package brown to /Users/anwar/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

Out[5]:
```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

In [6]:

```
#List the first 10 words in the corpus
print(brown.words(categories='fiction')[:10])
# How many words are there in the corpus
print("All:", len(brown.words()))
# How many words in the adventure category
print("adventure:",len(brown.words(categories='adventure')))
# Print the last 10 sentences from the romance categories
print("sents from romance:",brown.sents(categories="romance")[-10:])
#How many sentences are there in total
print("All Sentences:", len(brown.sents()))
```

```
['Thirty-three', 'Scotty', 'did', 'not', 'go', 'back', 'to', 'school',
'.', 'His']
All: 1161192
adventure: 69342
sents from romance: [["I'm", 'not', 'giving', 'you', 'a', 'chance',
',', 'Bill', ',', 'but', 'availing', 'myself', 'of', 'your', 'generou
s', 'offer', 'of', 'assistance', '.'], ['Good', 'luck', 'to', 'you',
"''", '.'], ['``', 'All', 'the', 'in-laws', 'have', 'got', 'to', 'hav
e', 'their', 'day', "''", ',', 'Adam', 'said', ',', 'and', 'glared',
'at', 'William', 'and', 'Freddy', 'in', 'turn', '.'], ['Sweat', 'start
ed', 'out', 'on', "William's", 'forehead', ',', 'whether', 'from', 're
lief', 'or', 'disquietude', 'he', 'could', 'not', 'tell', '.'], ['Acro
ss', 'the', 'table', ',', 'Hamrick', 'saluted', 'him', 'jubilantly',
'with', 'an', 'encircled', 'thumb', 'and', 'forefinger', '.'], ['Nobod
y', 'else', 'showed', 'pleasure', '.'], ['Spike-haired', ',', 'burly',
',', 'red-faced', ',', 'decked', 'with', 'horn-rimmed', 'glasses', 'an
d', 'an', 'Ivy', 'League', 'suit', ',', 'Jack', 'Hamrick', 'awaited',
'William', 'at', 'the', "officers'", 'club', '.'], ['``', 'Hello',
',', 'boss', "''", ',', 'he', 'said', ',', 'and', 'grinned', '.'], ['`
`', 'I', 'suppose', 'I', 'can', 'never', 'expect', 'to', 'call', 'yo
u', "'", 'General', "'", 'after', 'that', 'Washington', 'episode',
"''", '.'], ['``', "I'm", 'afraid', 'not', "''", '.']]
All Sentences: 57340
```

NLTK also includes Web and chat texts from `Firefox discussion forum`, the `movie script` from `Pirates of the Caribbean`, `wine reviews`, ...

To import the Web and chat corpus:

```
from nltk.corpus import webtext
```

In [7]:

```
from nltk.corpus import webtext

for id in webtext.fileids():
    #display some parts of the text in the document
    print(id, webtext.raw(id)[:30],"...")
```

```
firefox.txt Cookie Manager: "Don't allow s ...
grail.txt SCENE 1: [wind] [clop clop clo ...
overheard.txt White guy: So, do you have any ...
pirates.txt PIRATES OF THE CARRIBEAN: DEAD ...
singles.txt 25 SEXY MALE, seeks attrac old ...
wine.txt Lovely delicate, fragrant Rhon ...
```

NLTK also include `inaugural speeches` of US presidents.

You can import these corpora as follows

```
from nltk.corpus import inaugural
```

In [8]:

```
from nltk.corpus import inaugural
# print all inaugural
print("Inaugral Texts:",inaugural.fileids())
inaugural.raw("2017-Trump.txt")[:1000]
```

```
Inaugral Texts: ['1789-Washington.txt', '1793-Washington.txt', '1797-A
dams.txt', '1801-Jefferson.txt', '1805-Jefferson.txt', '1809-Madison.t
xt', '1813-Madison.txt', '1817-Monroe.txt', '1821-Monroe.txt', '1825-A
dams.txt', '1829-Jackson.txt', '1833-Jackson.txt', '1837-VanBuren.tx
t', '1841-Harrison.txt', '1845-Polk.txt', '1849-Taylor.txt', '1853-Pie
rce.txt', '1857-Buchanan.txt', '1861-Lincoln.txt', '1865-Lincoln.txt',
'1869-Grant.txt', '1873-Grant.txt', '1877-Hayes.txt', '1881-Garfield.t
xt', '1885-Cleveland.txt', '1889-Harrison.txt', '1893-Cleveland.txt',
'1897-McKinley.txt', '1901-McKinley.txt', '1905-Roosevelt.txt', '1909-
Taft.txt', '1913-Wilson.txt', '1917-Wilson.txt', '1921-Harding.txt',
'1925-Coolidge.txt', '1929-Hoover.txt', '1933-Roosevelt.txt', '1937-Ro
osevelt.txt', '1941-Roosevelt.txt', '1945-Roosevelt.txt', '1949-Truma
n.txt', '1953-Eisenhower.txt', '1957-Eisenhower.txt', '1961-Kennedy.tx
t', '1965-Johnson.txt', '1969-Nixon.txt', '1973-Nixon.txt', '1977-Cart
er.txt', '1981-Reagan.txt', '1985-Reagan.txt', '1989-Bush.txt', '1993-
Clinton.txt', '1997-Clinton.txt', '2001-Bush.txt', '2005-Bush.txt', '2
009-Obama.txt', '2013-Obama.txt', '2017-Trump.txt', '2021-Biden.txt']
```

Out[8]:

```
"Chief Justice Roberts, President Carter, President Clinton, President
Bush, President Obama, fellow Americans, and people of the world: Than
k you.\n\nWe, the citizens of America, are now joined in a great natio
nal effort to rebuild our country and restore its promise for all of o
ur people. Together, we will determine the course of America and the w
orld for many, many years to come. We will face challenges, we will co
nfront hardships, but we will get the job done.\n\nEvery 4 years, we g
ather on these steps to carry out the orderly and peaceful transfer of
power, and we are grateful to President Obama and First Lady Michelle
Obama for their gracious aid throughout this transition. They have bee
n magnificent. Thank you.\n\nToday's ceremony, however, has very speci
al meaning. Because today we are not merely transferring power from on
e administration to another or from one party to another, but we are t
ransferring power from Washington, DC, and giving it back to you, the
people.\n\nFor too long, a "
```

NLTK contains tagged and parsed corpora that we will discuss in the upcoming practice classes

In [9]:

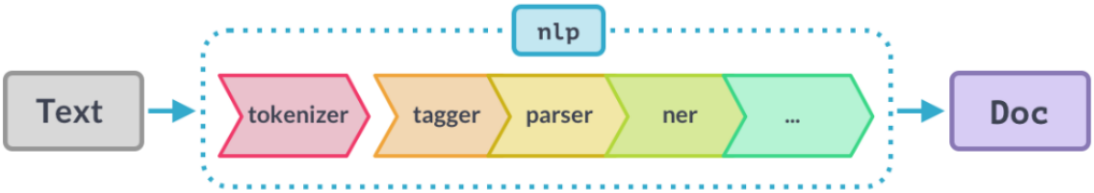
```
import nltk
nltk.download('maxent_ne_chunker')
nltk.download('averaged_perceptron_tagger')
nltk.download('words')

texts = [
    """
   ONCE UPON A TIME a girl named Cinderella lived with her stepmothe
   Poor Cinderella had to work hard all day long so the others could
   It was she who had to wake up each morning when it was still dark
   It was she who cooked the meals. It was she who kept the fire goi
   The poor girl could not stay clean, from all the ashes and cinder
    """
]
for text in texts:
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        tagged_words = nltk.pos_tag(words)
        ne_tagged_words = nltk.ne_chunk(tagged_words)
        print (ne_tagged_words)
```

```
(S
  ONCE/RB
  UPON/IN
  A/NNP
  TIME/NNP
  a/DT
  girl/NN
  named/VBN
  (PERSON Cinderella/NNP)
  lived/VBD
  with/IN
  her/PRP$
  stepmother/NN
  and/CC
  two/CD
  stepsisters/NNS
  ./.)
(S
  (PERSON Poor/NNP)
  (PERSON Cinderella/NNP)
  had/VBD
  to/TO
  work/VB
  hard/RB
  all/DT
  day/NN
  long/RB
  so/RB
  the/DT
  others/NNS
  could/MD
  rest/VB
  ./.)
(S
  It/PRP
  was/VBD
  she/PRP
  who/WP
  had/VBD
  to/TO
  wake/VB
  up/RP
  each/DT
  morning/NN
  when/WRB
  it/PRP
  was/VBD
  still/RB
  dark/JJ
  and/CC
  cold/JJ
  to/TO
  start/VB
  the/DT
  fire/NN
  ./.)
(S It/PRP was/VBD she/PRP who/WP cooked/VBD the/DT meals/NNS ./.)
(S
  It/PRP
  was/VBD
  she/PRP
```

```
  who/WP
  kept/VBD
  the/DT
  fire/NN
  going/VBG
  ./.)
(S
  The/DT
  poor/JJ
  girl/NN
  could/MD
  not/RB
  stay/VB
  clean/JJ
  ,/,
  from/IN
  all/PDT
  the/DT
  ashes/NNS
  and/CC
  cinders/NNS
  by/IN
  the/DT
  fire/NN
  ./.)
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /Users/anwar/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/anwar/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package words to /Users/anwar/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

spaCy

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. It is a package, with a support for a number of nlp tasks such as tokenization, lemmatisation, part-of-speech tagging, entity recognition and so on.



How to Install spaCy

To install spaCy, visit the documentation page [here (https://spacy.io/usage)](https://spacy.io/usage)

Using pip

```
pip install -U spacy
```

If you want to install it from the notebook, append the ! before the command

```
!pip install -U spacy
```

To install some models, run the following from the command line (make sure to use the correct python version, that is python3)

```
python3 -m spacy download
en_core_web_sm
```

or

```
python3 -m spacy download en
```

`en_core_web_sm` : (11 MB) English multi-task CNN trained on OntoNotes. Assigns context-specific token vectors, POS tags, dependency parse and named entities.

`en_core_web_md` : (91 MB) English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl. Assigns word vectors, context-specific token vectors, POS tags, dependency parse and named entities.

`en_core_web_lg` : (789 MB) English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl.

In [10]:

```
!pip install -U spacy
```

```
Requirement already satisfied: spacy in /Users/anwar/miniconda3/lib/py
thon3.8/site-packages (3.7.2)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy) (2.4.
2)
Requirement already satisfied: setuptools in /Users/anwar/miniconda3/l
ib/python3.8/site-packages (from spacy) (61.2.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from spacy) (4.63.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from spacy) (2.27.1)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /Users/anwar/mi
niconda3/lib/python3.8/site-packages (from spacy) (1.1.2)
Requirement already satisfied: numpy>=1.15.0 in /Users/anwar/miniconda
3/lib/python3.8/site-packages (from spacy) (1.24.1)
Requirement already satisfied: packaging>=20.0 in /Users/anwar/minicon
da3/lib/python3.8/site-packages (from spacy) (22.0)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from spacy) (2.0.10)
Requirement already satisfied: thinc<8.3.0,>=8.1.8 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from spacy) (8.2.1)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /Users/anwar/mi
niconda3/lib/python3.8/site-packages (from spacy) (0.3.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from spacy) (3.3.0)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /Users/a
nwar/miniconda3/lib/python3.8/site-packages (from spacy) (1.0.5)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /Users/a
nwar/miniconda3/lib/python3.8/site-packages (from spacy) (3.0.12)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /Users/anwar/mi
niconda3/lib/python3.8/site-packages (from spacy) (0.9.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from spacy) (2.4.8)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from spacy) (3.0.9)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from spacy) (6.3.0)
Requirement already satisfied: jinja2 in /Users/anwar/miniconda3/lib/p
ython3.8/site-packages (from spacy) (3.1.2)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /Users/anw
ar/miniconda3/lib/python3.8/site-packages (from spacy) (1.0.10)
Requirement already satisfied: pydantic-core==2.10.1 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<3.
0.0,>=1.7.4->spacy) (2.10.1)
Requirement already satisfied: typing-extensions>=4.6.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<
3.0.0,>=1.7.4->spacy) (4.7.1)
Requirement already satisfied: annotated-types>=0.4.0 in /Users/anwar/
miniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<3.
0.0,>=1.7.4->spacy) (0.6.0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/anwar/mini
conda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spac
y) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /Users/anwar/miniconda
3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy)
(3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->s
pacy) (1.26.8)
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/anw
ar/miniconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.
0->spacy) (2.0.4)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /Users/anwar/mini
conda3/lib/python3.8/site-packages (from thinc<8.3.0,>=8.1.8->spacy)
(0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from thinc<8.3.0,>=8.1.8->sp
acy) (0.1.3)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from typer<0.10.0,>=0.3.0->spacy)
(8.1.6)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /Users/a
nwar/miniconda3/lib/python3.8/site-packages (from weasel<0.4.0,>=0.1.0
->spacy) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/anwar/minicon
da3/lib/python3.8/site-packages (from jinja2->spacy) (2.1.1)
```

In [11]:

```
!python3 -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.7.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.0/en_core_web_sm-3.7.0-py3-none-any.whl (https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.0/en_core_web_sm-3.7.0-py3-none-any.whl) (12.8 MB)
     |████████████████████████████████| 12.8 MB 3.3 MB/s eta 0:00:01
Requirement already satisfied: spacy<3.8.0,>=3.7.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from en-core-web-sm==3.7.0) (3.7.2)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (1.0.5)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.4.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (4.63.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.27.1)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.4.8)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (1.0.10)
Requirement already satisfied: jinja2 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (3.1.2)
Requirement already satisfied: packaging>=20.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (22.0)
Requirement already satisfied: thinc<8.3.0,>=8.1.8 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (8.2.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.0.8)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (3.0.12)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.3.3)
Requirement already satisfied: numpy>=1.15.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (1.24.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.0.10)
Requirement already satisfied: setuptools in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (61.2.0)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /Users/anwar/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /Users/anwa
```

```
r/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en
-core-web-sm==3.7.0) (6.3.0)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /Users/anwar/mi
niconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-cor
e-web-sm==3.7.0) (1.1.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en-co
re-web-sm==3.7.0) (3.0.9)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from spacy<3.8.0,>=3.7.0->en
-core-web-sm==3.7.0) (3.3.0)
Requirement already satisfied: pydantic-core==2.10.1 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<3.
0.0,>=1.7.4->spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.10.1)
Requirement already satisfied: typing-extensions>=4.6.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<
3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (4.7.1)
Requirement already satisfied: annotated-types>=0.4.0 in /Users/anwar/
miniconda3/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,<3.
0.0,>=1.7.4->spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.6.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/anw
ar/miniconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.
0->spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /Users/anwar/mini
conda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spac
y<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/anwar/m
iniconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->s
pacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (1.26.8)
Requirement already satisfied: idna<4,>=2.5 in /Users/anwar/miniconda
3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.
8.0,>=3.7.0->en-core-web-sm==3.7.0) (3.3)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (from thinc<8.3.0,>=8.1.8->sp
acy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.1.3)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /Users/anwar/mini
conda3/lib/python3.8/site-packages (from thinc<8.3.0,>=8.1.8->spacy<3.
8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.7.11)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /Users/anwar/min
iconda3/lib/python3.8/site-packages (from typer<0.10.0,>=0.3.0->spacy<
3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (8.1.6)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /Users/a
nwar/miniconda3/lib/python3.8/site-packages (from weasel<0.4.0,>=0.1.0
->spacy<3.8.0,>=3.7.0->en-core-web-sm==3.7.0) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/anwar/minicon
da3/lib/python3.8/site-packages (from jinja2->spacy<3.8.0,>=3.7.0->en-
core-web-sm==3.7.0) (2.1.1)
```

✔ Download and installation successful

You can now load the package via spacy.load('en_core_web_sm')

In [12]:

```
1 # might be neede!pip install typing_extensions==4.7.1 --upgrade
```

```
Requirement already satisfied: typing_extensions==4.7.1 in /Users/anwa
r/miniconda3/lib/python3.8/site-packages (4.7.1)
```

In [13]:

```
# start spaCy's pipeline, which can load differnt spacy models
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("The big grey dog ate all of the chocolate, but fortunatel
#Tokenization
print("Using split", doc.text.split())
#Using spaCy, only orth_ method
print("Using spaCy",[token.orth_ for token in doc])
# using spacy, including the integer representation of tokens
print("spaCy integer",[(token, token.orth_, token.orth) for token in
```

```
Using split ['The', 'big', 'grey', 'dog', 'ate', 'all', 'of', 'the',
'chocolate,', 'but', 'fortunately', 'the', 'he', "wasn't", 'sick!']
Using spaCy ['The', 'big', 'grey', 'dog', 'ate', 'all', 'of', 'the',
'chocolate', ',', 'but', 'fortunately', 'the', 'he', 'was', "n't", 'si
ck', '!']
spaCy integer [(The, 'The', 5059648917813135842), (big, 'big', 1551163
2813958231649), (grey, 'grey', 10475807793332549289), (dog, 'dog', 756
2983679033046312), (ate, 'ate', 10806788082624814911), (all, 'all', 13
409319323822384369), (of, 'of', 886050111519832510), (the, 'the', 7425
985699627899538), (chocolate, 'chocolate', 10946593968795032542), (,,
',', 2593208677638477497), (but, 'but', 14560795576765492085), (fortun
ately, 'fortunately', 13851269277375979931), (the, 'the', 742598569962
7899538), (he, 'he', 1655312771067108281), (was, 'was', 99216865133789
12864), (n't, "n't", 2043519015752540944), (sick, 'sick', 148415976098
57081305), (!, '!', 17494803046312582752)]
```

Important functions for language processing

1. Referencing, Equality, Conditionals, Sequences, Operations on sequences, Rearranging, Zipping, enumerating

In [14]:
```
a = 'English'
b = a
a = 'French'
#  b is Still English
print (b)
print("---")
```

```
English
---
```

In [15]:
```
a = ["English","Spanish"]
b = a
a[0] ="French"
# Now b is changed ==> access by reference
print (b)
print("---")
```

```
['French', 'Spanish']
---
```

In [16]:

```
# the same reference for all 5 entries
a = ["German"] *5
print ([a[i] for i,_ in enumerate(a)])
# Are they the same? use the 'is' keyword
print (a[0] is a[1] is a[2] is a[3] is a[4])
print("---")
```

```
['German', 'German', 'German', 'German', 'German']
True
---
```

In [17]:

```
# change value of one of the entry (the forth item)
a[3] = "French"
print ([a[i] for i,_ in enumerate(a)])
print (a[0] is a[1] is a[2] is a[3] is a[4])
print("---")
```

```
['German', 'German', 'German', 'French', 'German']
False
---
```

In [18]:

```
# conditioning
words = ["the", "quick", "brown", "fox",",", "jumps", "over", "the",
punct ="!.,"
for word in words:
    if word in punct:
        print (word + "is a punctuation", end="; ")# print single li

```

```
,is a punctuation; !is a punctuation;
```

In [19]:
```
# Zip (concatenate lists). Example words, frequencies, POS Tag
print()
print("---")
words = ['I', 'turned', 'off', 'the', 'spectroroute']
freq = [15, 3,6,23,1]
tags = ['noun', 'verb', 'prep', 'det', 'noun']
print ("Zipped:",  list(zip(words, tags, freq)))
print("---")
```

```
---
Zipped: [('I', 'noun', 15), ('turned', 'verb', 3), ('off', 'prep', 6),
('the', 'det', 23), ('spectroroute', 'noun', 1)]
---
```

In [20]:
```
# Sorting, reversing, differences
words_on = ['I', 'turned', 'on', 'the', 'light']
print ("sorted:", sorted(words))
print ("reveresed:",list(reversed(words)))
print ("difference:", set(words).difference(words_on))
print ("difference:",set(words_on).difference(words))
print("---")
```

```
sorted: ['I', 'off', 'spectroroute', 'the', 'turned']
reveresed: ['spectroroute', 'the', 'off', 'turned', 'I']
difference: {'off', 'spectroroute'}
difference: {'light', 'on'}
---
```

In [21]:
```
# rearrenging lists
words_on[2], words_on[3], words_on[4] = words_on[3],words_on[4],word
print("rearranged:",words_on)
```

```
rearranged: ['I', 'turned', 'the', 'light', 'on']
```

2. Splitting datasets:

We will discuss in upcoming practice class on how to properly split datasets into training, development, and test set. There are

In [22]:

```
text = """
This is a simple training example.
We split by the text into training and test sets.
Most of the data goes to training.
Small parts for testing
"""
split = int(0.9 * len(text.split()))
train, test = text.split()[:split], text.split()[split:]
print ("train set",train)
print("test set", test)
```

```
train set ['This', 'is', 'a', 'simple', 'training', 'example.', 'We',
'split', 'by', 'the', 'text', 'into', 'training', 'and', 'test', 'set
s.', 'Most', 'of', 'the', 'data', 'goes', 'to', 'training.', 'Small']
test set ['parts', 'for', 'testing']
```

3. Generator

Assume you are searching a given word in a corpus

```
def search1(substring, words):
    result = []
    for word in words:
        if substring in word:
            result.append(word)
    return result
```

Here in search1, the list is accumulated and the final result is returned once the search is over. What if the file is very large...?

```
def search2(substring, words):
    for word in words:
        if substring in word:
            yield word
```

Here, search2 uses a yield statement, which is a generator. It will pause when it get the first word and the caller function will do

In [23]:

```
import nltk
def search1(substring, words):
    result = []
    for word in words:
        if substring in word:
            result.append(word)
    return result

def search2(substring, words):
    for word in words:
        if substring in word:
            yield word


for item in search1('zzo', nltk.corpus.brown.words()):
    print(item, end=" ")
print("\n----")

for item in search2('zzo', nltk.corpus.brown.words()):
    print(item, end=" ")
```

```
Palazzo Palazzo Palazzo Palazzo palazzo palazzo Palazzo Palazzo Palazz
o palazzo Palazzo palazzo Piazzo mezzo palazzos
----
Palazzo Palazzo Palazzo Palazzo palazzo palazzo Palazzo Palazzo Palazz
o palazzo Palazzo palazzo Piazzo mezzo palazzos
```

4. Procedural and declarative style

Example, count the average length of a token

In [24]:
```
#Procedural - average word length
tokens = nltk.corpus.brown.words(categories='news')
count = 0
total = 0
for token in tokens:
    count += 1
    total += len(token)
print(total / count)

```

```
4.401545438271973
```

In [25]:
```
#Declarative - average word length
tokens = nltk.corpus.brown.words(categories='news')
print (sum(len(t) for t in tokens)/len(tokens))
```

```
4.401545438271973
```

In [26]:
```
# Procedural - longest word
text = nltk.corpus.gutenberg.words('milton-paradise.txt')
longest = ''
for word in text:
    if len(word) > len(longest):
        longest = word
print(longest) # Print the longest word (ONLY one).
```

```
unextinguishable
```

In [27]:
```
# Procedural - longest word
maxlen = max(len(word) for word in text)# find longest word length
print([word for word in text if len(word) == maxlen]) # print those
```

```
['unextinguishable', 'transubstantiate', 'inextinguishable', 'incompre
hensible']
```

5. Using the map functions

In [28]:
```
# maps - applies a function to every item in a sequence
# Example - average lenghts of a sentence in the news Brown corpus
lengths = list(map(len, nltk.corpus.brown.sents(categories='news')))
print (sum(lengths) / len(lengths))
# the same as the following
lengths = [len(sent) for sent in nltk.corpus.brown.sents(categories=
print(sum(lengths) / len(lengths))

```

```
21.75081116158339
21.75081116158339
```

6. Using lambda functions in maps

In [29]:
```
# count the number of vowels in each word
sent = ['Take', 'care', 'of', 'the', 'sense', ',', 'and', 'the', 'so
list(map(lambda w: len(list(filter(lambda c: c.lower() in "aeiou", w
```

Out[29]:
```
[2, 2, 1, 1, 2, 0, 1, 1, 2, 1, 2, 2, 1, 3, 0]
```

7. Named arguments

When there are a lot of parameters for a function, it is possible to refer them by name, even using default values.

In [30]:

```
def pos_tag(word, tag="UNK"):
    return {word: tag}
print(pos_tag("man", "NN"))
print(pos_tag("mann"))
# *args, **kwargs ==> variable number of unamed and named/keywords a
def freq_words(file, min=1, num=10):
    text = open(file).read()
    tokens = nltk.word_tokenize(text)
    freql = nltk.FreqDist(t for t in tokens if len(t) >= min)
    return list(freql.keys())[:num]
print(freq_words('data/news/news1.txt', 14, 5))
print(freq_words('data/news/news1.txt', min=14, num=5))
print(freq_words('data/news/news1.txt', num=5, min=14))
```

```
{'man': 'NN'}
{'mann': 'UNK'}
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
```

8. Matplotlib

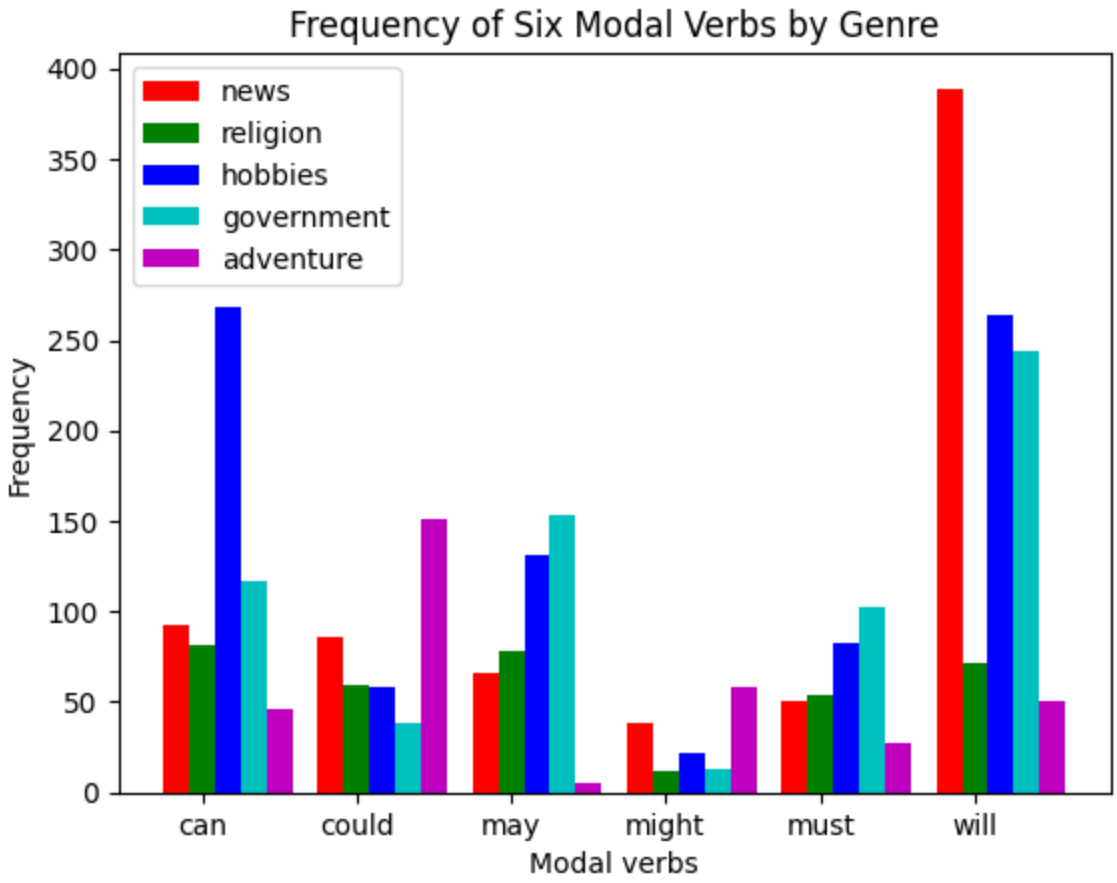
Matplotlib is a Python 2D plotting library which produces high quality figures in a variety of formats and interactive environments across platforms. It can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers.

In [31]:

```
# Draw a barchart for the 6 modal verbs from the brown corpus
from numpy import arange
from matplotlib import pyplot
%matplotlib inline
colors = 'rgbcmyk' # red, green, blue, cyan, magenta, yellow, black
def bar_chart(categories, words, counts):
    "Plot a bar chart showing counts for each word by category"
    ind = arange(len(words))
    width = 1 / (len(categories) + 1)
    bar_groups = []
    for c in range(len(categories)):
        bars = pyplot.bar(ind+c*width, counts[categories[c]], width,
                          color=colors[c % len(colors)])
        bar_groups.append(bars)
    pyplot.xticks(ind+width, words)
    pyplot.legend([b[0] for b in bar_groups], categories, loc='upper
    pyplot.ylabel('Frequency');
    pyplot.xlabel('Modal verbs');
    pyplot.title('Frequency of Six Modal Verbs by Genre');
    pyplot.show();

genres = ['news', 'religion', 'hobbies', 'government', 'adventure']
modals = ['can', 'could', 'may', 'might', 'must', 'will']

cfdist = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in genres
    for word in nltk.corpus.brown.words(categories=genre)
    if word in modals)

counts = {}
for genre in genres:
    counts[genre] = [cfdist[genre][word] for word in modals]
print(genres, modals, counts)

```

```
36 bar_chart(genres, modals, counts)
```

```
['news', 'religion', 'hobbies', 'government', 'adventure'] ['can', 'co
uld', 'may', 'might', 'must', 'will'] {'news': [93, 86, 66, 38, 50, 38
9], 'religion': [82, 59, 78, 12, 54, 71], 'hobbies': [268, 58, 131, 2
2, 83, 264], 'government': [117, 38, 153, 13, 102, 244], 'adventure':
[46, 151, 5, 58, 27, 50]}
```



Exercise 1 (5 points)

Submit befor the practice class of next week

1. Count the number of tokens and types in `data/news/news.txt`. Use the `split()` method to split the text with white space

2. Count and show the words that are common between `data/news/news.txt` and `data/news/news1.txt`.
3. Print the largest word (s) for each letter (a,b,c ...z) from the nltk books corpus using declarative style. Example ... `i->incomprehensible,inextinguishable; t->transubstantiate`, ...
4. Print the most frequent starting and ending characters of words in the nltk books.
5. Design an algorithm to find the `statistically improbable phrases` (SIP) of a document collection. You might just come up with a `Pseudocode`. SIPs are phrases that are less probably in a `universal` collection but common to a particular topic. For example, `dependency parsing` might be a common phrase that occur in computational linguistic journals but less frequent when compared with a global science journal indexes.

---

my name

Solution 1

In [ ]:
```
1
```

In [ ]:
```
1
```

Materials

- spaCy (https://course.spacy.io/)
- NLTK book (http://www.nltk.org/book/)
- Handbook of NLP (https://karczmarczuk.users.greyc.fr/TEACH/TAL/Doc/Handboo