IS434
Social Analytics & Applications
Lab 9: Text Analysis

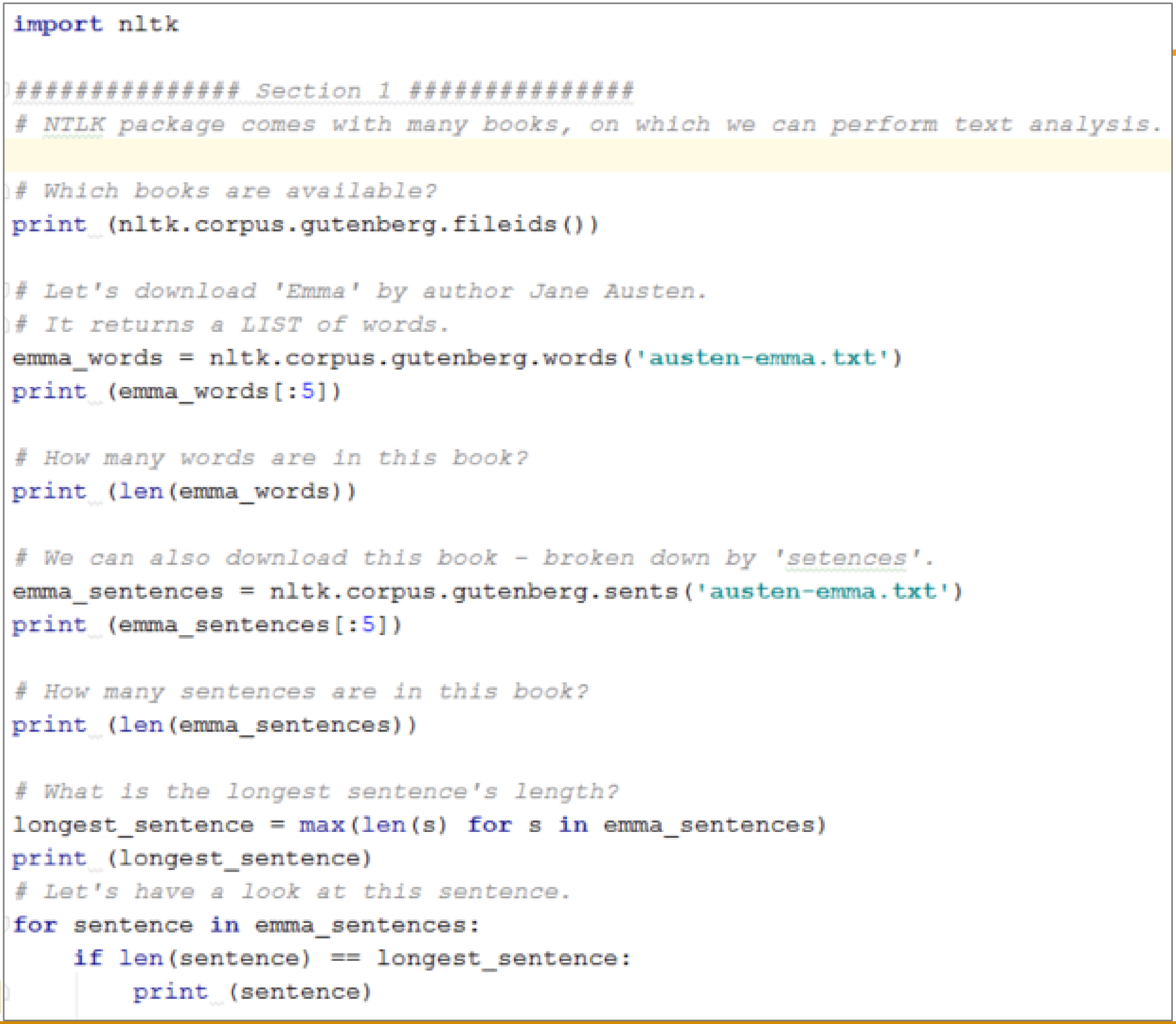
Instructor: K. Shim

Email: kjshim@smu.edu.sg
Office: SIS Level 5, Room 5027
Phone (O): 6808-5275

Lab 9 – Get Lab files from eLearn

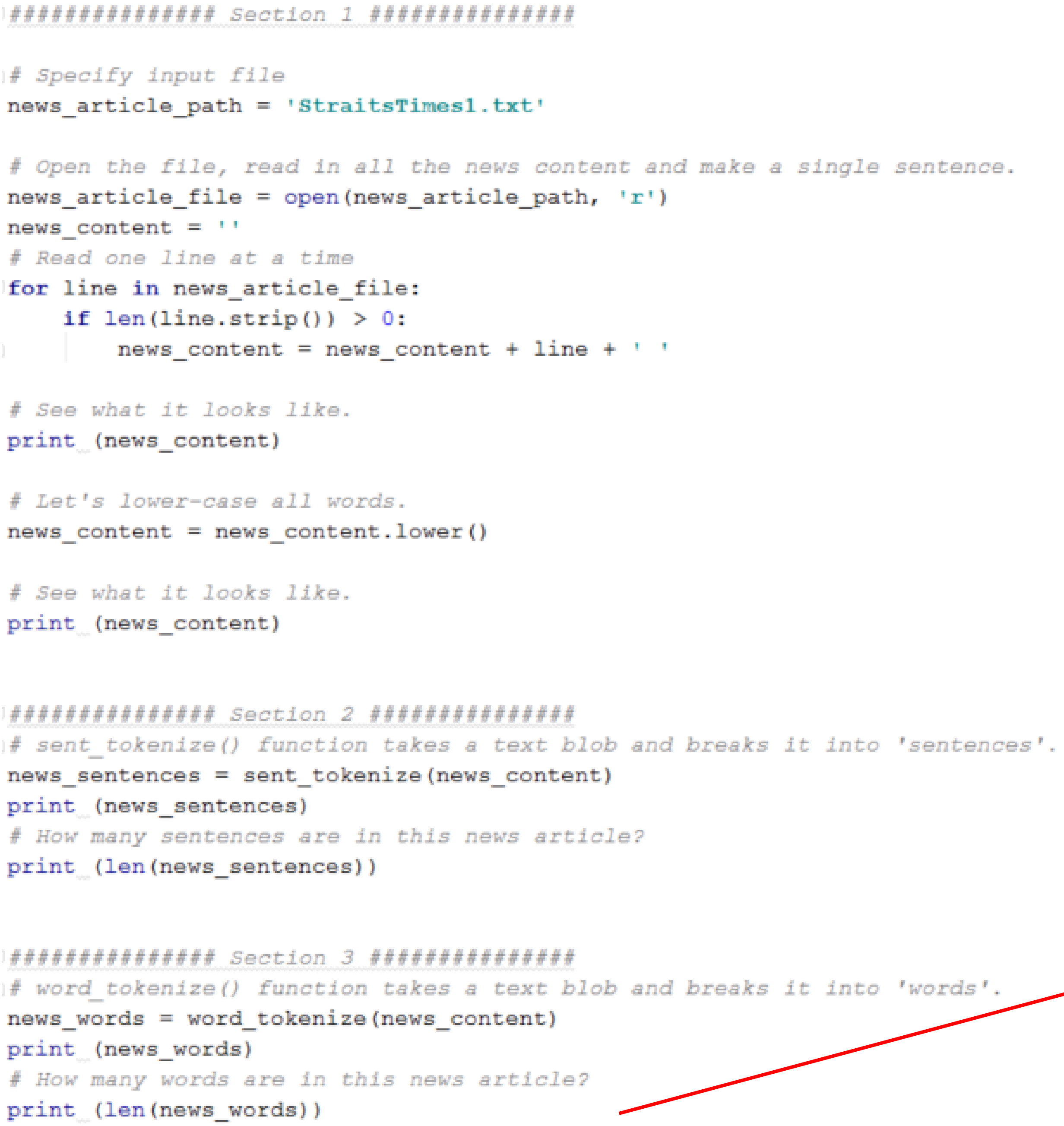
- eLearn → Content → Labs → **Lab9.zip**
- Unzip it.
- Port all files/directories over to your EC2 instance!

Lab9_News_TextAnalysis.ipynb



```
import nltk

############### Section 1 ###############
# NTLK package comes with many books, on which we can perform text analysis.

# Which books are available?
print (nltk.corpus.gutenberg.fileids())

# Let's download 'Emma' by author Jane Austen.
# It returns a LIST of words.
emma_words = nltk.corpus.gutenberg.words('austen-emma.txt')
print (emma_words[:5])

# How many words are in this book?
print (len(emma_words))

# We can also download this book - broken down by 'setences'.
emma_sentences = nltk.corpus.gutenberg.sents('austen-emma.txt')
print (emma_sentences[:5])

# How many sentences are in this book?
print (len(emma_sentences))

# What is the longest sentence's length?
longest_sentence = max(len(s) for s in emma_sentences)
print (longest_sentence)
# Let's have a look at this sentence.
for sentence in emma_sentences:
    if len(sentence) == longest_sentence:
        print (sentence)
```

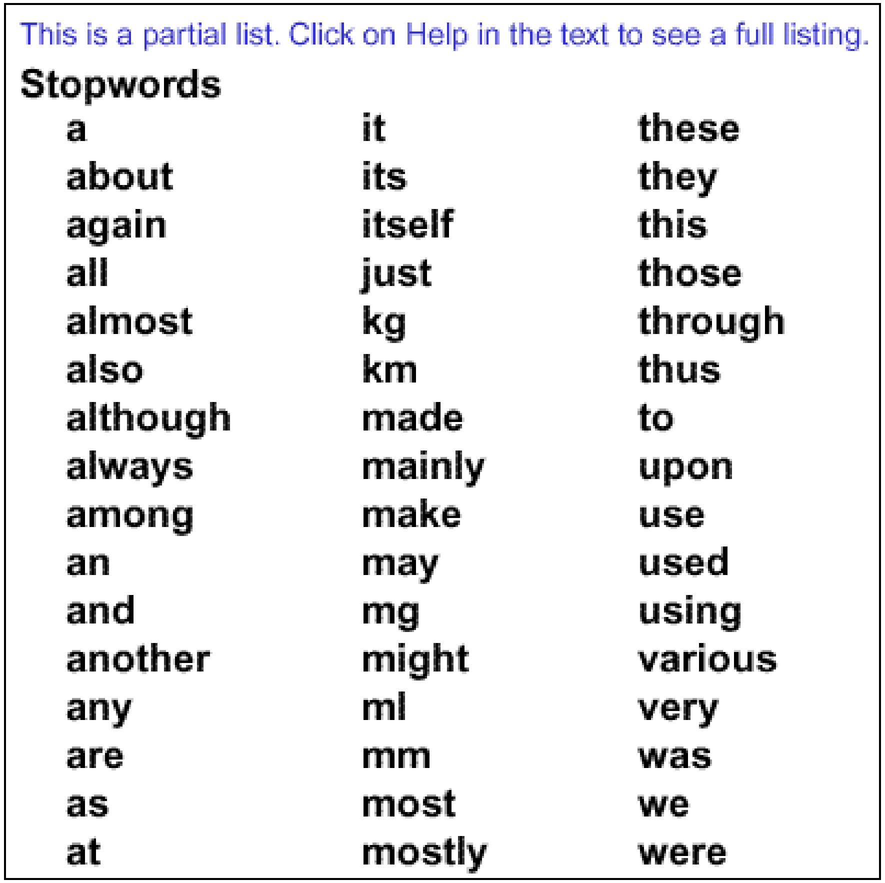
Lab9_News_TextAnalysis.ipynb



```
############### Section 1 ###############

# Specify input file
news_article_path = 'StraitsTimes1.txt'

# Open the file, read in all the news content and make a single sentence.
news_article_file = open(news_article_path, 'r')
news_content = ''
# Read one line at a time
for line in news_article_file:
    if len(line.strip()) > 0:
        news_content = news_content + line + ' '

# See what it looks like.
print (news_content)

# Let's lower-case all words.
news_content = news_content.lower()

# See what it looks like.
print (news_content)


############### Section 2 ###############
# sent_tokenize() function takes a text blob and breaks it into 'sentences'.
news_sentences = sent_tokenize(news_content)
print (news_sentences)
# How many sentences are in this news article?
print (len(news_sentences))


############### Section 3 ###############
# word_tokenize() function takes a text blob and breaks it into 'words'.
news_words = word_tokenize(news_content)
print (news_words)
# How many words are in this news article?
print (len(news_words))
```

Do you notice…
punctuations and
stop words?

Stop Words



This is a partial list. Click on Help in the text to see a full listing.

Stopwords

a
about
again
all
almost
also
although
always
among
an
and
another
any
are
as
at

it
its
itself
just
kg
km
made
mainly
make
may
mg
might
ml
mm
most
mostly

these
they
this
those
through
thus
to
upon
use
used
using
various
very
was
we
were

Words that don’t represent any **entity** or **sentiment**.
→ In most cases, they don’t mean anything.

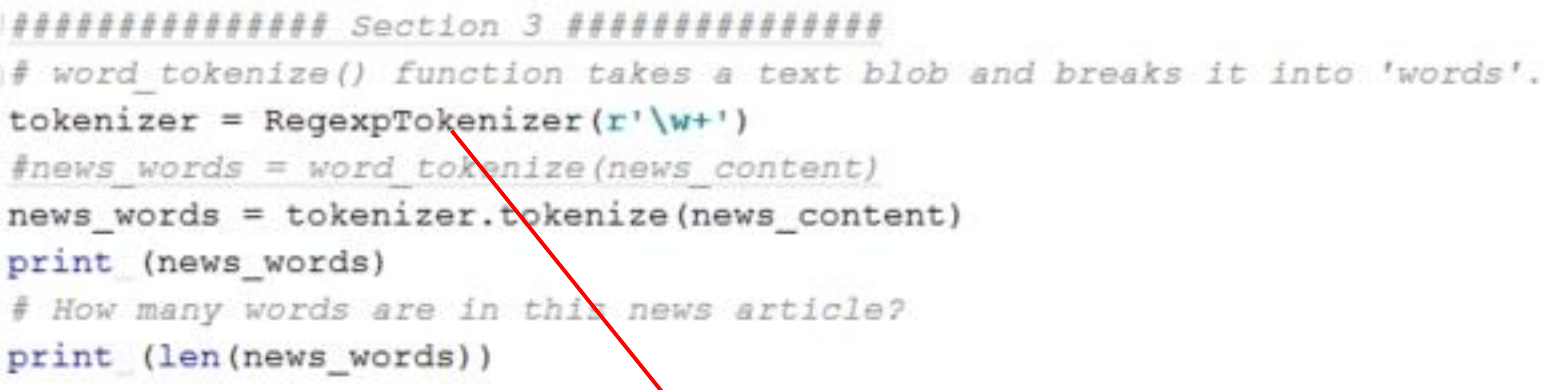
Frequent appearance of “a”, “an”, “the”, “it” in English phrases… more frequent than entities or sentiment words.

Entities are… people, company names (e.g. “Donald Trump”).

Sentiment words are… adjectives or emotion-indicating words (e.g. “awesome”, “awful”, “bad”, “good”)

So… what do we do with them?
→ We **remove them**. This is part of **data cleaning**.

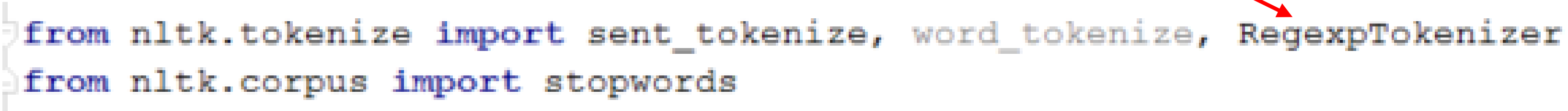
Removing Punctuations



```
############### Section 3 ###############
# word_tokenize() function takes a text blob and breaks it into 'words'.
tokenizer = RegexpTokenizer(r'\w+')
#news_words = word_tokenize(news_content)
news_words = tokenizer.tokenize(news_content)
print (news_words)
# How many words are in this news article?
print (len(news_words))
```

Go back to **Section 3** and use **RegexpTokenizer**.
Using this **tokenizer** will remove all the **punctuations**.

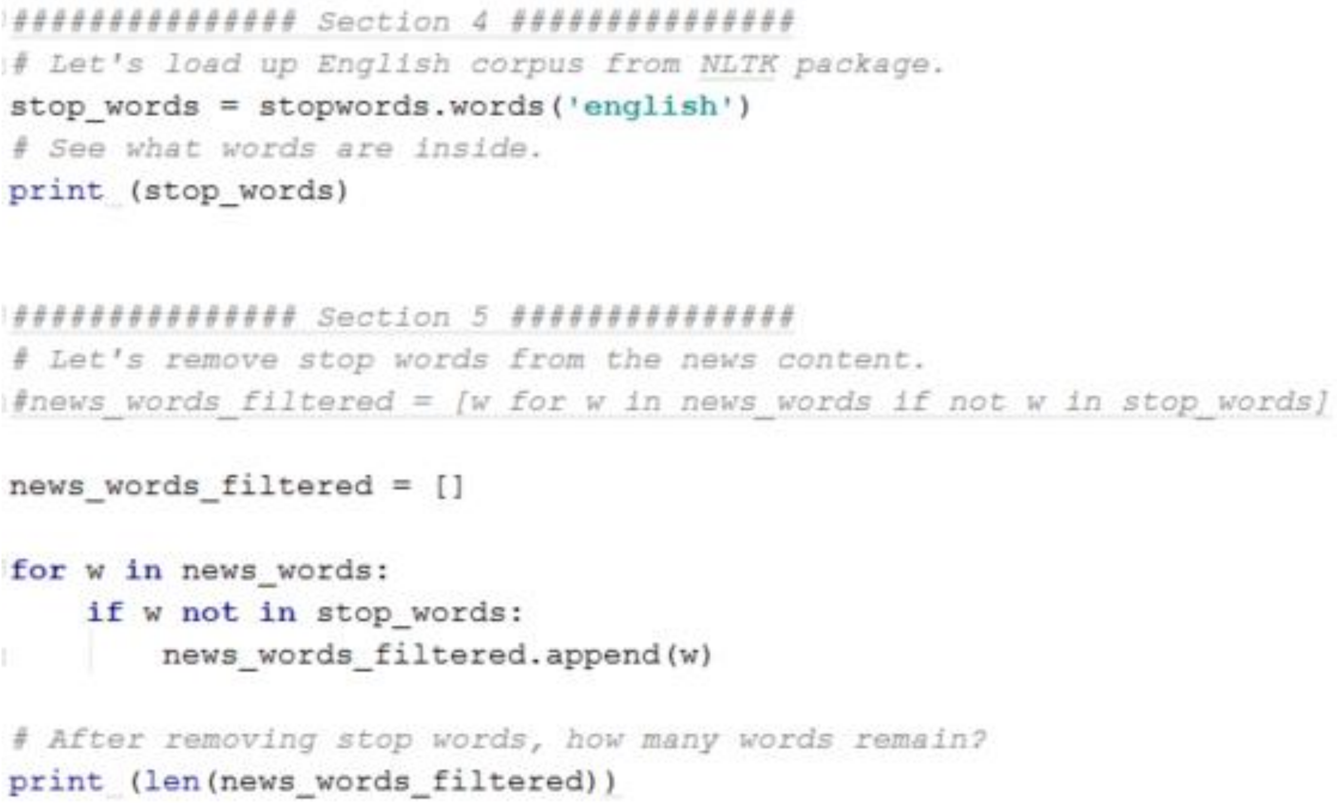
Don’t forget to import it…



```
from nltk.tokenize import sent_tokenize, word_tokenize, RegexpTokenizer
from nltk.corpus import stopwords
```

Lab9_News_TextAnalysis.ipynb

Removing Stop Words



```
############### Section 4 ###############
# Let's load up English corpus from NLTK package.
stop_words = stopwords.words('english')
# See what words are inside.
print (stop_words)


############### Section 5 ###############
# Let's remove stop words from the news content.
#news_words_filtered = [w for w in news_words if not w in stop_words]

news_words_filtered = []

for w in news_words:
    if w not in stop_words:
        news_words_filtered.append(w)

# After removing stop words, how many words remain?
print (len(news_words_filtered))
```

You can also use this instead of FOR loop a few lines below.

Stemming

I was taking a ride in the car.
I was riding in the car.

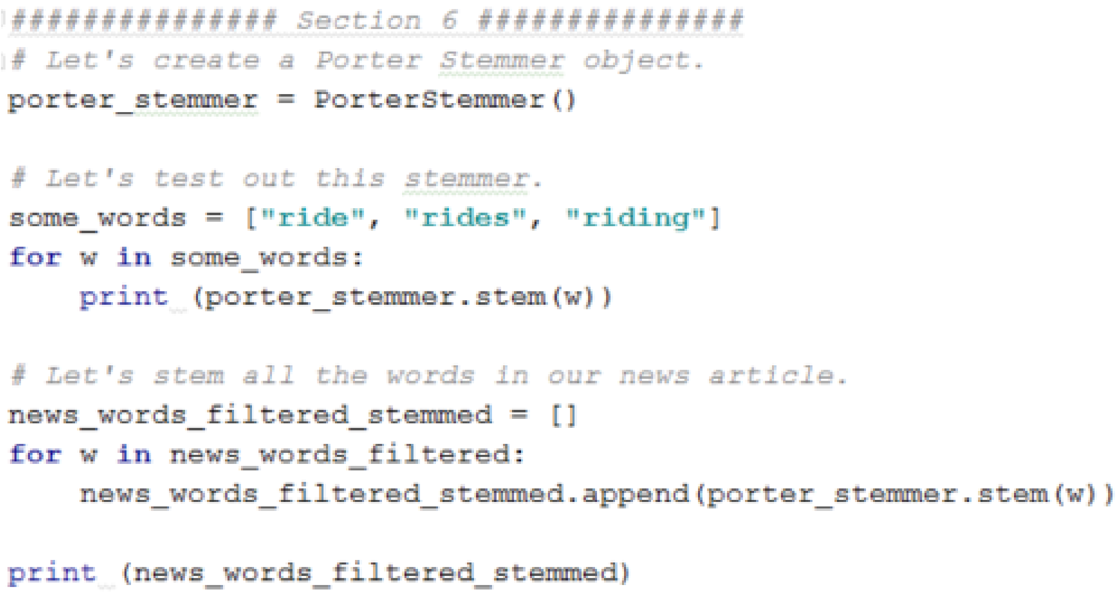
- If we’re simply trying to figure out… what are people’s favorite **activities** from a large number of **tweets** or **blogs**;
- **“ride”, “riding”** → these two words mean the **same** in our analysis.

- **This is just one example**. Can you imagine… every word in English language, tense and variations...

- Porter **stemmer** is one of the most widely used stemming algorithms, developed in 1979.

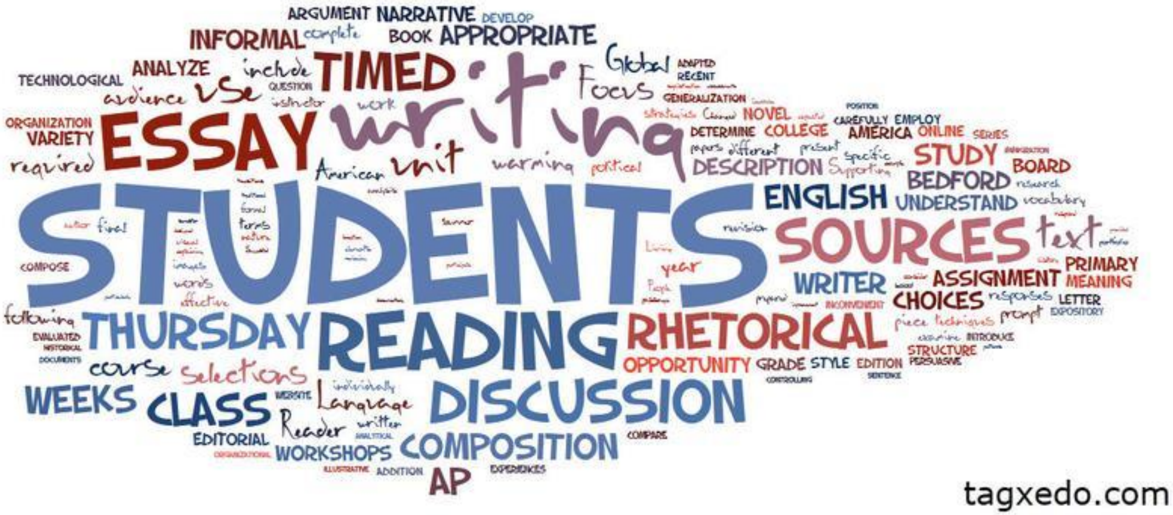
- We will use **stemmer(s)** to standardize words.

Stemming



```
############### Section 6 ###############
# Let's create a Porter Stemmer object.
porter_stemmer = PorterStemmer()

# Let's test out this stemmer.
some_words = ["ride", "rides", "riding"]
for w in some_words:
    print (porter_stemmer.stem(w))

# Let's stem all the words in our news article.
news_words_filtered_stemmed = []
for w in news_words_filtered:
    news_words_filtered_stemmed.append(porter_stemmer.stem(w))

print (news_words_filtered_stemmed)
```

Word Cloud



tagxedo.com

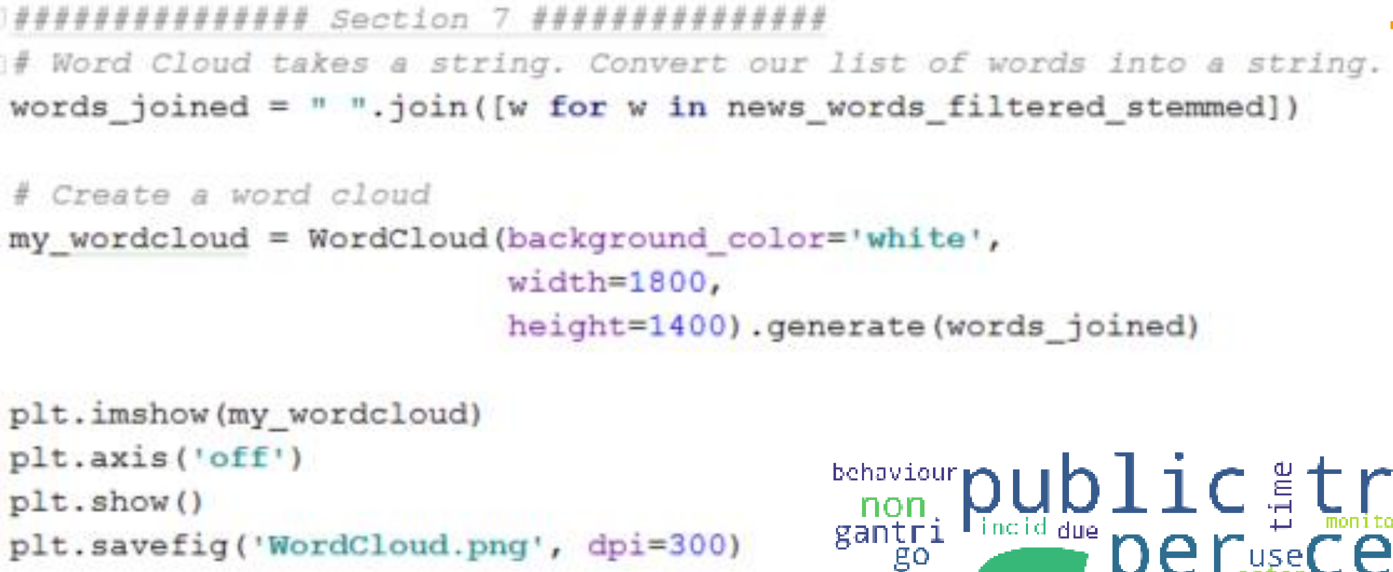
- Word Cloud = Weighted List
  - Weight = Term frequency
- Can easily see frequently mentioned terms
- Cannot see term-to-term associations
- A bit of cleaning… (stop words, punctuations) is needed.
  - We did this step already.

Word Cloud

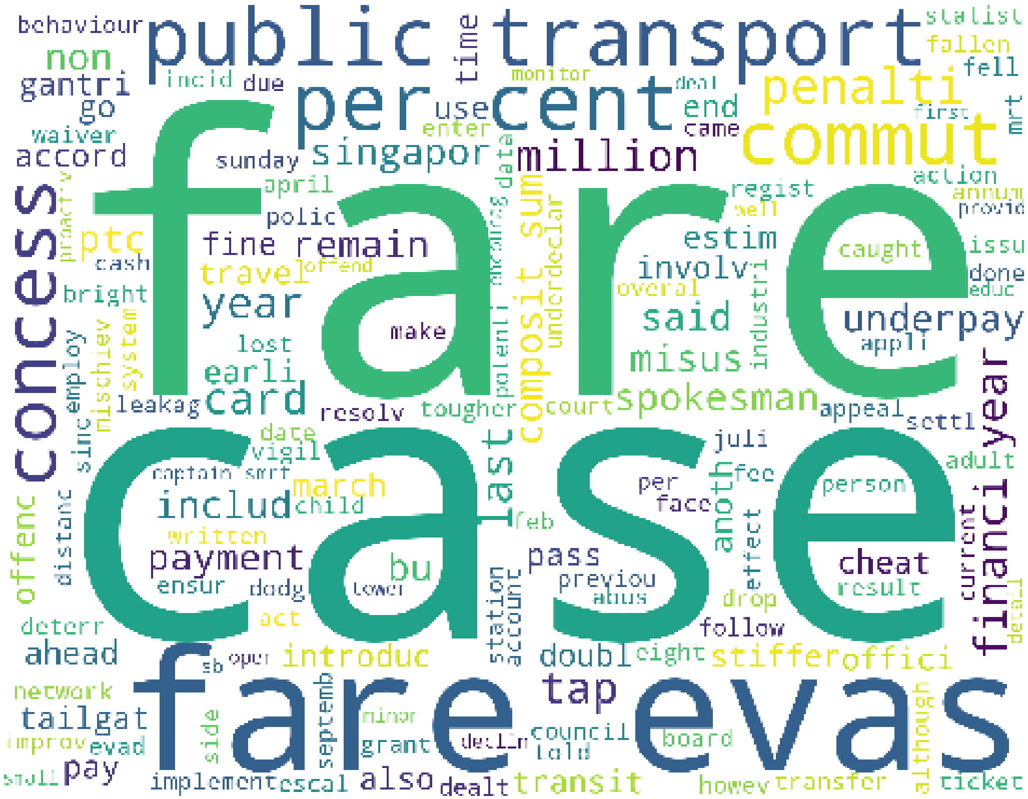
- Install (if you don’t have it) **wordcloud** package and **Pillow** package in Anaconda.
- Google search “Cabin Sketch” font. Download the font and install it in your local computer.
  - – https://www.fontsquirrel.com/fonts/cabinsketch
- For tutorials:
  - – Windows: http://www.softwareok.com/?seite=faq-Windows-8&faq=87
  - – Mac: http://www.fontspring.com/support/installing/how-do-i-install-fonts-on-my-mac

Lab9_News_TextAnalysis.ipynb

Word Cloud

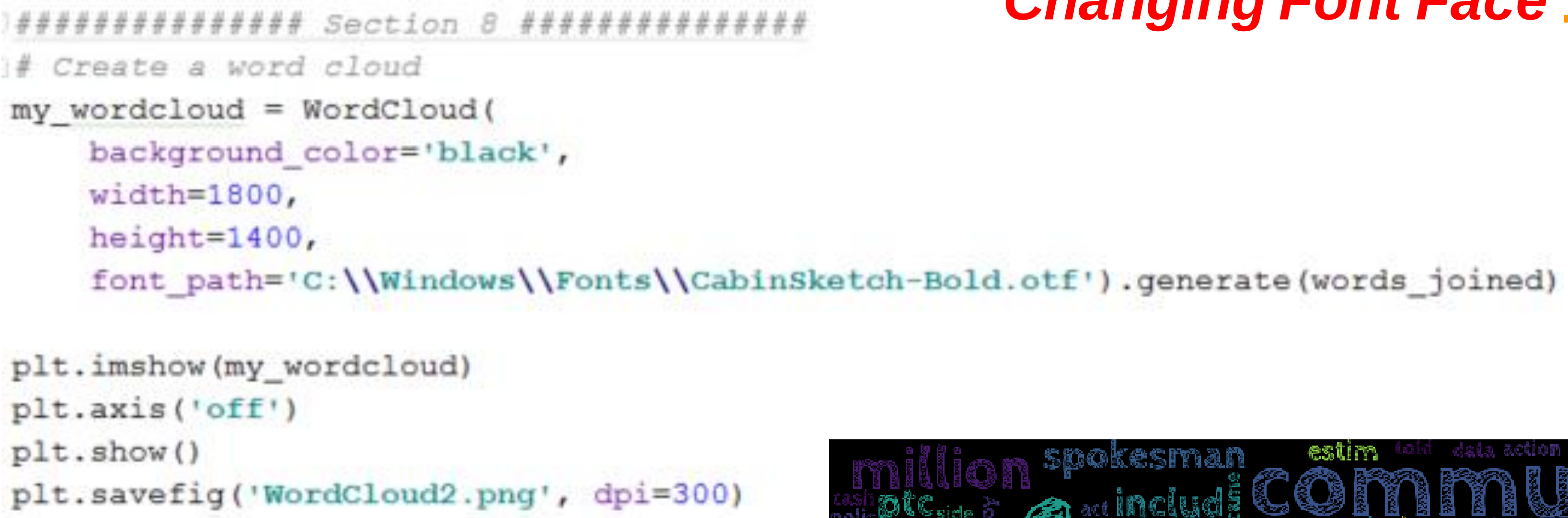


```
############### Section 7 ###############
# Word Cloud takes a string. Convert our list of words into a string.
words_joined = " ".join([w for w in news_words_filtered_stemmed])

# Create a word cloud
my_wordcloud = WordCloud(background_color='white',
                         width=1800,
                         height=1400).generate(words_joined)

plt.imshow(my_wordcloud)
plt.axis('off')
plt.show()
plt.savefig('WordCloud.png', dpi=300)
```

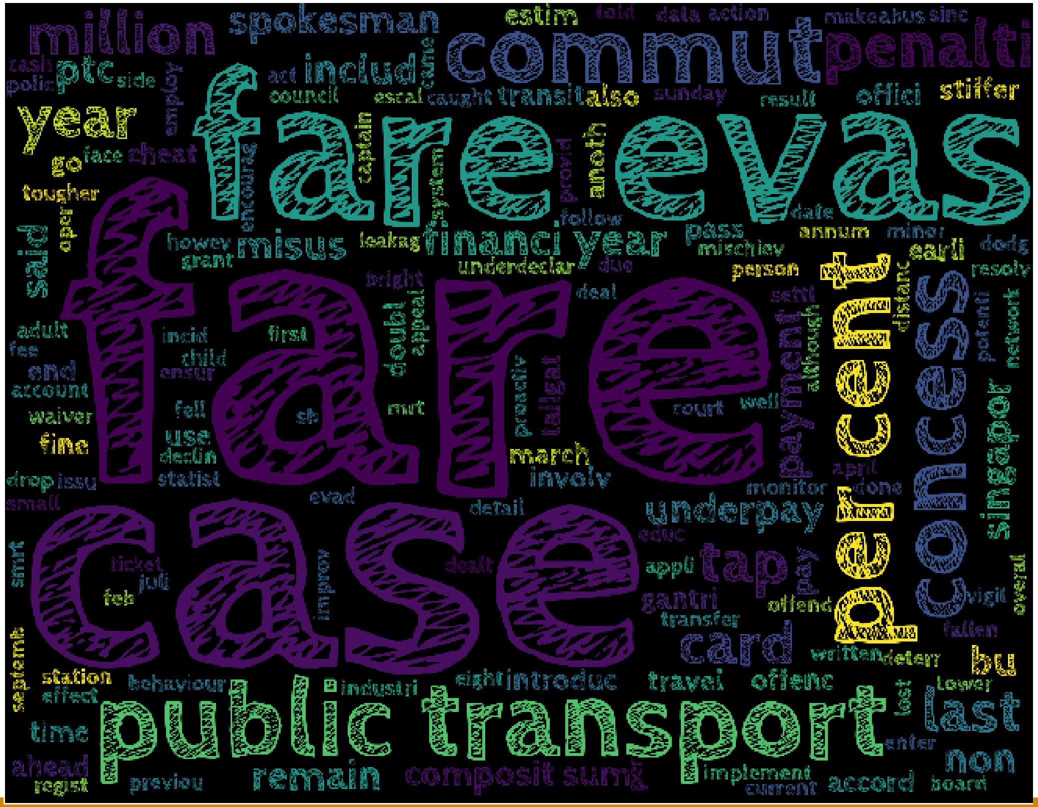


Lab9_News_TextAnalysis.ipynb

Word Cloud
Changing Font Face



```
############### Section 8 ###############
# Create a word cloud
my_wordcloud = WordCloud(
    background_color='black',
    width=1800,
    height=1400,
    font_path='C:\\Windows\\Fonts\\CabinSketch-Bold.otf').generate(words_joined)

plt.imshow(my_wordcloud)
plt.axis('off')
plt.show()
plt.savefig('WordCloud2.png', dpi=300)
```



If you're running this code from Linux OS (e.g. EC2 instance), you need to change the **font_path** to point to the correct OTF file location in Linux.

Sentiment Analysis – Rule-Based (Part 1)

- Lab9_SentimentAnalysis1.ipynb
- **Corpus/Corpora** (inside **dict/** and **dict2/** directories)
  - Positive corpus
  - Negative corpus
  - Incrementer corpus
  - Decrementer corpus
  - Inverter corpus

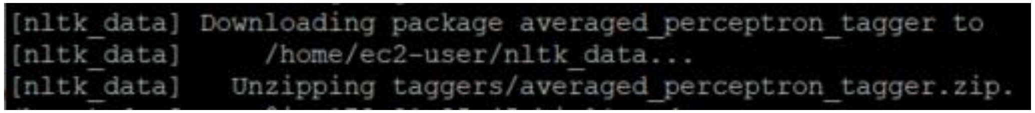
- **Before you run this code, you need to do the following**
  - Go to EC2 Linux command line
  - Go to your **Anaconda's bin directory**. For example:

```
> cd /home/ec2-user/anaconda/bin
```

  - You need to **download** the below dictionary

```
> python -m nltk.downloader averaged_perceptron_tagger
```

You will see messages like this upon successful download:



```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/ec2-user/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

Sentiment Analysis – Rule-Based (Part 1)

- Run **Cell 2 & Cell 3**
  - [Cell 2] Load up libraries & specify the location of **dictionaries**
  - [Cell 3] Defines helper **Classes** and **functions**
- Scroll all the way to the bottom… to the **LAST CELL** in the Jupyter workbook.
  - Here is a sample restaurant review. Clearly, it is a negative review. Let's perform sentiment scoring on this.
- **run_analysis(___)** function will take a review comment (which can span multiple sentences).
- **run_analysis()** performs the following:
  1. Split the review comment into one or more sentences.
  2. Part-Of-Speech (POS) tagging will be done on each sentence.
  3. Calculates **sentiment score** based on +ve/-ve corpora.
     - But what about incrementers and decrementers?
  4. Calculates **sentiment score** based on +ve/-ve corpora & incrementer/decrementer corpora.
  5. Calculates **sentiment score** based on +ve/-ve corpora & incrementer/decrementer corpora & inverter corpus.

Sentiment Analysis – Rule-Based (Part 2)

- **Lab9_SentimentAnalysis2.ipynb**
- In **Part 1**, all positives received **+1** and all negatives received **-1**.
- In **Part 2**, there are **3 levels** of **positivity** and **3 levels** of **negativity**.
- You can define a scale of your own. For example, you might define **5 levels** of positivity/negativity.

```
excellent: [positive3]
good:[positive2]
okay: [positive1]
```

```
lousy: [negative2]
awful: [negative3]
mediocre: [negative1]
```

okay < good < excellent

mediocre < lousy < awful

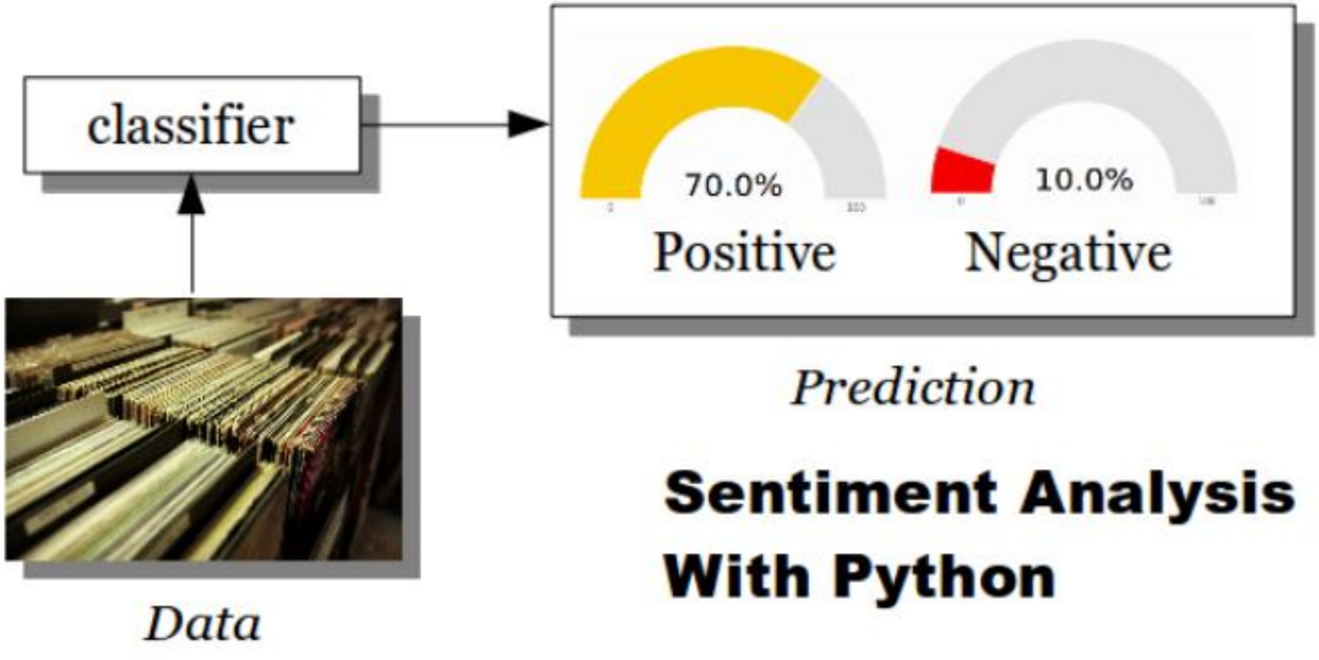
- The **sentiment dictionaries** for **Part 2** are located in **dict2/** directory.

Lab 9: Yelp Restaurant Review Challenge

- Copy **Lab9_SentimentAnalysis2.ipynb** into a new file called "**Lab9_SentimentAnalysis2_Yelp.ipynb**".
- This new script must:
  1. Read from **Yelp_Food_Input.txt**.
  2. Use appropriate **chart(s)** to summarize the customer reviews.
     1. How many restaurant reviews are there?
     2. Word Cloud
     3. Top 20 **words/terms** mentioned in the review comments
  3. For each restaurant review, compute and assign a **sentiment score**.
  4. Use appropriate **chart(s)** to summarize sentiment analysis results. Are there more positive comments than negative comments? What does the distribution (of sentiment scores) look like? (e.g. boxplot)
  5. Post one or more charts with description to **Twitter** using hashtag **#is434**

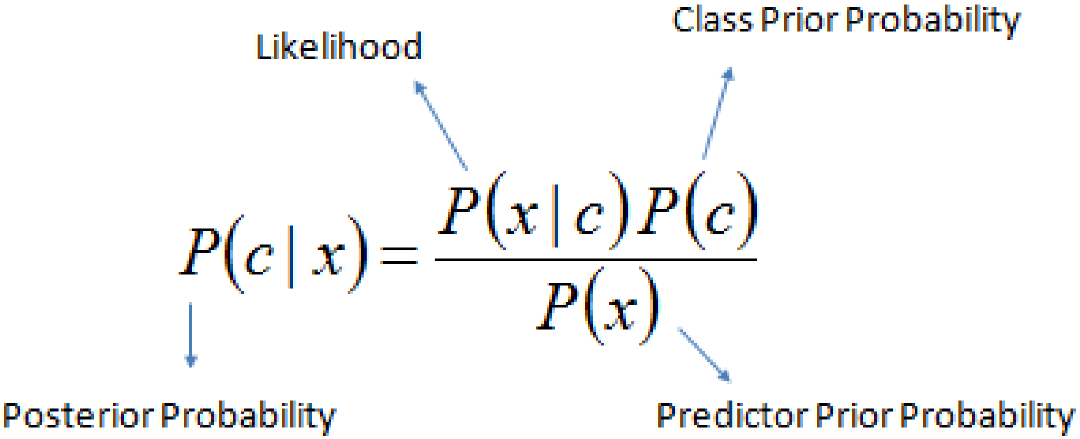
Lab 9 – Sentiment Classification

- **Lab9_SentimentClassification1.ipynb**
- Given a movie review or a tweet, it can be automatically classified in categories.
- These categories can be user defined (positive, negative) or whichever classes you want.



Lab 9 – Sentiment Classification

- **Lab9_SentimentClassification2.ipynb**
- Here, we will use **IMDB** movie reviews as the baseline dataset (training/testing) to build **Sentiment Classification** model.
  - This dataset comes with **NLTK package**.
- In doing so, we will use **Naïve Bayes** algorithm.



Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

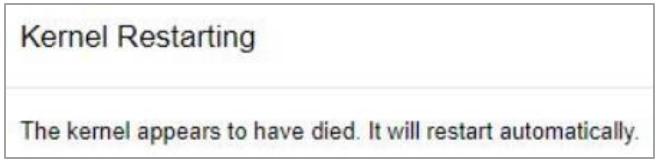
Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Lab 9: Topic Modeling

- **Files**
  - **Lab9_TopicModeling_V1.ipynb**
  - **Lab9_TopicModeling_V2.ipynb**
- What is topic modeling?
  - Technique to extract **hidden topics** from **large volumes of text**
  - *Examples: feeds from **social media**, **customer reviews** of businesses, **emails of customer complaints**, etc.*
  - It is very time-consuming to read through large volumes of text and compile the topics.
  - Latent Dirichlet Allocation (LDA) is one of the automated algorithms that can read through text documents and automatically output **topics**.
- For you to do:
  - Try each script above. Both are very well commented – please read the tutorial comments.
  - Note that topic modeling is a very memory-intensive task. Running the above scripts on a large file on a **micro EC2 instance** may lead to a "Dead Kernel".
  - Solutions
    - Try the scripts on your local laptop computer's Jupyter first.
    - So that you understand what the scripts do.
    - You can try the scripts on a larger EC2 instance (e.g. xLarge).
      - Note that the AWS Free Tier does NOT include > micro EC2.
      - Using larger capacity servers will be charged to your account.



Kernel Restarting

The kernel appears to have died. It will restart automatically.