CS 505 Homework 02: Data Wrangling and BOW

Due Thursday 9/21 at midnight (1 minute after 11:59 pm) in Gradescope (with a grace period of 6 hours)

You may submit the homework up to 24 hours late (with the same grace period) for a penalty of 10%.

All homeworks will be scored with a maximum of 100 points; point values are given for individual problems, and if parts of problems do not have point values given, they will be counted equally toward the total for that problem.

Note: I strongly recommend you work in **Google Colab** (the free version) to complete homeworks in this class; in addition to (probably) being faster than your laptop, all the necessary libraries will already be available to you, and you don't have to hassle with `conda`, `pip`, etc. and resolving problems when the install doesn't work. But it is up to you! You should go through the necessary tutorials listed on the web site concerning Colab and storing files on a Google Drive. And of course, Dr. Google is always ready to help you resolve your problems.

I will post a "walk-through" video ASAP on my Youtube Channel.

Submission Instructions

You must complete the homework by editing **this notebook** and submitting the following two files in Gradescope by the due date and time:

- A file `HW02.ipynb` (be sure to select `Kernel -> Restart and Run All` before you submit, to make sure everything works); and
- A file `HW02.pdf` created from the previous.

  For best results obtaining a clean PDF file on the Mac, select `File -> Print Review` from the Jupyter window, then choose `File-> Print` in your browser and then `Save as PDF`. Something similar should be possible on a Windows machine -- just make sure it is readable and no cell contents have been cut off. Make it easy to grade!

The date and time of your submission is the last file you submitted, so if your IPYNB file is submitted on time, but your PDF is late, then your submission is late.

Collaborators (5 pts)

Describe briefly but precisely

1. Any persons you discussed this homework with and the nature of the discussion;
2. Any online resources you consulted and what information you got from those resources; and
3. Any AI agents (such as chatGPT or CoPilot) or other applications you used to complete the homework, and the nature of the help you received.

A few brief sentences is all that I am looking for here.

```
- www.geeksforgeeks.com : surfed to understand about lemma
and removing special charectors from the text, to create
dictionary, to look for Lemma
- www.regular-expressions.info : for understanding word
bounderies and look ahhead/ look behind for not removing
colons
- chatGPT :  to get an idea on when and how to remove stop
words if they are necessary for the text
```

Overview

We are going to practice converting raw (string form) text into a useful data set using the script of *Pirates of the Caribbean: The Curse of the Black Pearl* (2003), the first in a series of PotC movies starring Johnny Depp. The script is part of the `webtext` corpus in NLTK.

Under the assumption that we wish to perform an analysis of the words spoken by the characters in the movie, we will convert the text in a series of steps from a raw string of ASCII characters into a dictionary holding a sparse BOW model of the words spoken by two of the main characters, Elizabeth Swann and Jack Sparrow. These dictionaries could be the data set for a classification task, or for creating a vector-space model for each character, which we will study later in the course. For this assignment, you will clean up, normalize and tokenize the text, create the dictionaries, and then simply print out the most common words spoken by the two characters.

Text normalization was covered in lecture on Tuesday 9/12 and the BOW model on Thursday 9/14. Before beginning the assignment, you should consult the following for information on using the Python regular expression library:

https://docs.python.org/3/library/re.html

Also useful is

https://docs.python.org/3/howto/regex.html

After reviewing the basic principles of regular expressions (which I will review in my walk-through video), read about the following useful functions:

```
result = re.split(...)

result = re.sub(...)
```

You may ONLY use standard functions from the `re` library for this homework, and you must perform your normalization starting with the string form of the script assigned below to the variable `pirates_txt`. You may NOT use indices of the string to perform your modifications (e.g., deleting the first line by counting how many characters to remove).

The point here is to use regular expressions to do the text wrangling. Don't worry, we shall use the `SpaCy` library later on the course to normalize text for a classification problem set.

In [1]:
```
import numpy as np
import nltk
import re

# The first time you will need to download the corpus:

nltk.download('webtext')

pirates_txt = nltk.corpus.webtext.raw('pirates.txt')
```

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

In [2]:
```
# raw string form
pirates_txt[:1000]
```

Out[2]:
"PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terry Rossio\n[view looking straight down at rolling swells, sound of wind and thunder, then a low heartbeat]\nScene: PORT ROYAL\n[teacups on a table in the rain]\n[sheet music on music stands in the rain]\n[bouquet of white orchids, Elizabeth sitting in the rain holding the bouquet]\n[men rowing, men on horseback, to the sound of thunder]\n[EITC logo on flag blowing in the wind]\n[many rowboats are entering the harbor]\n[Elizabeth sitting alone, at a distance]\n[marines running, kick a door in] \n[a mule is seen on the left in the barn where the marines enter]\n[Liz looking over her shoulder]\n[Elizabeth drops her bouquet]\n[Will is in manacles, being escorted by red coats]\nELIZABETH SWANN: Will...!\n[Elizabeth runs to Will]\nELIZABETH SWANN: Why is this happening? \nWILL TURNER: I don't know. You look beautiful.\nELIZABETH SWANN: I think it's bad luck for the groom to see the bride before the wedding.\n[marines cross their long axes to bar Go"

In [3]:
```
# printing it shows the formatting
print(pirates_txt[:1000])
```

```
PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terry Rossio
[view looking straight down at rolling swells, sound of wind and thunder, th
en a low heartbeat]
Scene: PORT ROYAL
[teacups on a table in the rain]
[sheet music on music stands in the rain]
[bouquet of white orchids, Elizabeth sitting in the rain holding the bouquet
]
[men rowing, men on horseback, to the sound of thunder]
[EITC logo on flag blowing in the wind]
[many rowboats are entering the harbor]
[Elizabeth sitting alone, at a distance]
[marines running, kick a door in]
[a mule is seen on the left in the barn where the marines enter]
[Liz looking over her shoulder]
[Elizabeth drops her bouquet]
[Will is in manacles, being escorted by red coats]
ELIZABETH SWANN: Will...!
[Elizabeth runs to Will]
ELIZABETH SWANN: Why is this happening?
WILL TURNER: I don't know. You look beautiful.
ELIZABETH SWANN: I think it's bad luck for the groom to see the bride before
the wedding.
[marines cross their long axes to bar Go
```

Problem One (35 points): Cleaning up the lines

The first task is to clean up the text so that at the conclusion of this problem, you will have a text with punctuation and extraneous characters removed, and each line consisting of a character's name (human or otherwise), a colon, and a sequence of words, ending in a newline.
(You will keep this as a single string until Problem 3, but we will refer to the "lines" spoken by each character -- also, I hope it will not be confusing to speak of (ASCII) characters and characters played by the actors in the script!)

Part 1.A (5 pts)

1. Convert the string into all lower-case letters.
2. Remove the first line which gives the title and authors. Print out the first 200 characters to show that you have done this.

Hint: Cut everything before the first '\n', using the 'beginning of string' special character `^` in the regular expression.

In [4]:
```
# Your code here
pirates_txt_lowered=pirates_txt.lower()
pirates_txt_lowered=re.sub(r'^.*?\n','', pirates_txt_lowered, 1)
print(pirates_txt_lowered[:200])
```

```
[view looking straight down at rolling swells, sound of wind and thunder, th
en a low heartbeat]
scene: port royal
[teacups on a table in the rain]
[sheet music on music stands in the rain]
[bouquet of
```

Part 1.B (5 pts)

Cut out all the stage directions that are given in square brackets, including the newlines on those lines. Print out the first 200 characters as proof.

In [5]:
```
# Your code here
pirates_txt_lowered = re.sub(r'\[.*?\]\n', '', pirates_txt_lowered, flags=re
print(pirates_txt_lowered[:200])
```

```
scene: port royal
elizabeth swann: will...!
elizabeth swann: why is this happening?
will turner: i don't know. you look beautiful.
elizabeth swann: i think it's bad luck for the groom to see the brid
```

Part 1.C (5 pts)

Cut out the lines where the 'scene' is specified. Again, print out the first 200 characters.

In [6]:
```
# Your code here
pirates_txt_lowered = re.sub(r'^scene:.*?\n', '', pirates_txt_lowered, flags
print(pirates_txt_lowered[:200])
```

```
elizabeth swann: will...!
elizabeth swann: why is this happening?
will turner: i don't know. you look beautiful.
elizabeth swann: i think it's bad luck for the groom to see the bride before
the weddi
```

Part 1.D (20 pts)

Now, we still have a lot of punctuation and some miscellaneous odd things occurring in this text, and we need to do further cleaning. But you will have to figure this out for yourself!

The main thing to do is to remove punctuation and anything that does not contribute to the goal of making a BOW model for our two characters.

But you can't just remove all non-word characters! Make sure you take account of the following:

1. You need to keep the character's names at the beginning of the line, so **do not remove the colon after the name** (note that these always occur at the beginning of a line, i.e., at the very beginning or immediately after the newline from the previous line).

2. In the next problem we will normalize the words, so **we DON'T want to change anything that might be a word,** such as,

   ```
   charges  don't  it's  'er  ah-ha  ha-ha-ha-ha-ha  stealin'
   ```

3. After observing the caveats above, **remove all punctuation.**

1. There are some places where apparently the transcriber was not sure what the word was and gave alternatives:

```
weren't/wasn't
```

and some places where it is not clear what is intended:

```
oy /quick
```

Just treat `/` like ordinary punctuation and replace it by a blank.

1. Finally, there are miscellanous weird things in the text, such as

```
###         ?:
```

(and possibly others) which **should be removed.**

How to proceed: To explore the data, print out the text after the modifications in Parts 1.A -- 1.D:

```
print(pirates_txt_01)
```

and think about what needs to be removed, paying careful attention to the comments above. (You could use the `Find` function in your browser to flip through various possibilities.)

After examining the text, **comment out the `print(pirates_txt_01)` so that we don't have to look at it!** This was just for exploration!

The result of your cleaning in this part should be assigned to `pirates_txt_01`.

Hint: At this stage, it might be better to **replace substrings with blanks** instead of deleting them (replacing with the empty string) to preserve the separation of words (just in case!).

In [7]:
```
#print(pirates_txt_lowered) just to explore
```

Part 1.D.1 (5 pts)

Write a short description here of what you removed, giving your reasoning. You must account for at least what is listed above, but you may find other things you want to change.

- Removed all the symbols and punctuations that are not apt for the text. To do this , i have given all the possible punctuations and symbols that might occur in the text to check and remove .
- there are few cases where '*shine*" the words had astrids before and after removed it .
- in one particular there is ### at the begining .

Also print out some portion of the text to show at least some of the changes you have made.

Part 1.D.2 (15 pts)

Write your code in the following cell. The result at the end should be stored in `pirates_txt_01`. Print out the first 2000 characters.

In [8]:
```
# Your code here
import re
empty_set = []
for line in pirates_txt_lowered.split('\n'):
    line = re.sub(r'^###|\?:', ' ', line)
    if ':' in line:
        char_name, after_colon = line.split(':', 1)
        taking_line = char_name + ':'
        after_colon = re.sub(r"(?<![a-zA-Z])[-']|[-'](?![a-zA-Z])|[!\"#$%&()
        after_colon = re.sub(r'###|\?:', ' ', after_colon);after_colon = re.
        taking_line += after_colon

    else:
        taking_line = re.sub(r"(?<![a-zA-Z])[-']|[-'](?![a-zA-Z])|[!\"#$%&()
    empty_set.append(taking_line)

pirates_txt_01 = '\n'.join(empty_set)
print(pirates_txt_01[:2000])
```

```
elizabeth swann: will
elizabeth swann: why is this happening
will turner: i don't know  you look beautiful
elizabeth swann: i think it's bad luck for the groom to see the bride before
the wedding
lord cutler beckett: governor weatherby swann  it's been too long
lord cutler beckett: his lord now    actually
lord cutler beckett: in fact  i  do   mister mercer  the warrant for the arr
est of one william turner
lord cutler beckett: oh  is it  that's annoying  my mistake  arrest her
elizabeth swann: on what charges
will turner: no
lord cutler beckett: ah-ha  here's the one for william turner  and i have an
other one for a mister james norrington  is he present
elizabeth swann:  what are the charges
lord cutler beckett: i don't believe that's the answer to the question i ask
ed
will turner: lord beckett  in the category of questions  not  answered
elizabeth swann: we are under the jurisdiction of the king's governor of por
t royal  and you will tell us what we are charged with
lord cutler beckett: for which the punishment  regrettably  is  also  death
perhaps you remember a certain pirate named jack sparrow
elizabeth swann:  captain  jack sparrow
lord cutler beckett: captain jack sparrow  yes  i thought you might
gibbs: fifteen men on a dead man's chest  yo ho ho  and a  bottle  of rum  d
rink and the devil had done for the rest  yo ho ho  and a  bottle  of rum  h
a-ha-ha-ha-ha
jack sparrow: sorry  mate
jack sparrow: mind if we make a little side trip  i didn't think so
gibbs: not  quite  according to plan
jack sparrow: complications arose  ensued  were overcome
gibbs: you got what you went in for  then
jack sparrow: mm-hmm
gibbs: captain  i think the crew  meaning me as well  were expecting somethi
ng a bit more     shiny   what with the isla de muerta going all pear shaped
reclaimed by the sea  and the treasure with it
leech: and the royal navy chasing us all around the atlantic
marty: and the hurricane  aye
crew: aye  aye
gibbs: all i
```

In [ ]:

Problem Two (30 points): Normalizing, Stemming, and Lemmatization

In this problem we are going to do **some** normalizing of the words, first of all to normalize certain words with apostrophes, and then performing stemming and lemmatization. We are not intended to be absolutely thorough here, just to try a few obvious possibilities.

Part 2.A Normalizing (15 pts)

There are several ways that apostophes (single quotes) are used to compress two words into one (to give a better sense for how they are pronounced, *especially by pirates*):

```
    didn't  =  did not    we've  = we have       there'd  =
there would
```

Your task: Find as many examples of these as you can, and replace the compressed word with the two-word phrase it represents.

Note: **Do NOT process any words with `'s`,** as these will be done in the next part.

Do NOT simply compile a list of specific examples, but look for general patterns for substitution, for example:

```
          n't =>  _not       've =>  _have       # where _
represents a blank
```

Simply find as many examples which seem to have a general rule, and perform those substitutions, putting the result in `pirates_txt_02`.

Finally, print out the first 2000 characters.

In [9]:
```
# Your code here

#went through possible conversion, here are the most i foundout to convert.
#pretty much decent conversions keeping the context as it is

pirates_txt_02 = re.sub(r"n't\b", " not", pirates_txt_01)
pirates_txt_02 = re.sub(r"'ve\b", " have", pirates_txt_02)
pirates_txt_02 = re.sub(r"(?<![sS])'d\b", " would", pirates_txt_02)
pirates_txt_02 = re.sub(r"\bwon't\b", "will not", pirates_txt_02)
#removing 'll as it is a stop word ,but will help preserve charector name "w
pirates_txt_02 = re.sub(r"\b'll\b", "", pirates_txt_02)
pirates_txt_02 = re.sub(r"\b're\b", " are", pirates_txt_02)
pirates_txt_02 = re.sub(r"\bi'm\b", "i am", pirates_txt_02)
pirates_txt_02 = re.sub(r"\bcap'n\b", "captain", pirates_txt_02)

print(pirates_txt_02[:2000])
```

```
elizabeth swann: will
elizabeth swann: why is this happening
will turner: i do not know  you look beautiful
elizabeth swann: i think it's bad luck for the groom to see the bride before
the wedding
lord cutler beckett: governor weatherby swann  it's been too long
lord cutler beckett: his lord now    actually
lord cutler beckett: in fact  i  do   mister mercer  the warrant for the arr
est of one william turner
lord cutler beckett: oh  is it  that's annoying  my mistake  arrest her
elizabeth swann: on what charges
will turner: no
lord cutler beckett: ah-ha  here's the one for william turner  and i have an
other one for a mister james norrington  is he present
elizabeth swann:  what are the charges
lord cutler beckett: i do not believe that's the answer to the question i as
ked
will turner: lord beckett  in the category of questions  not  answered
elizabeth swann: we are under the jurisdiction of the king's governor of por
t royal  and you will tell us what we are charged with
lord cutler beckett: for which the punishment  regrettably  is  also  death
perhaps you remember a certain pirate named jack sparrow
elizabeth swann:  captain  jack sparrow
lord cutler beckett: captain jack sparrow  yes  i thought you might
gibbs: fifteen men on a dead man's chest  yo ho ho  and a  bottle  of rum  d
rink and the devil had done for the rest  yo ho ho  and a  bottle  of rum  h
a-ha-ha-ha-ha
jack sparrow: sorry  mate
jack sparrow: mind if we make a little side trip  i did not think so
gibbs: not  quite  according to plan
jack sparrow: complications arose  ensued  were overcome
gibbs: you got what you went in for  then
jack sparrow: mm-hmm
gibbs: captain  i think the crew  meaning me as well  were expecting somethi
ng a bit more     shiny   what with the isla de muerta going all pear shaped
reclaimed by the sea  and the treasure with it
leech: and the royal navy chasing us all around the atlantic
marty: and the hurricane  aye
crew: aye  aye
gibbs: al
```

Part 2.B Stemming and Lemmatization (15 pts)

Stemming

There are multiple occurrence of the suffix `'s` in the text, some standing for a two word phrase:

```
    he's = he is       it's = it is       what's = what is
here's = here is
```

and some being possessives:

```
jack's     man's       hangman's
```

In the first case, the word `is` is very common, and would be removed later when we remove "stop words"; in the second, we will assume there is little difference in the BOW model between a noun and its possessive. So we will remove the `'s` from all words.

Lemmatization

There are eight "official' different forms of the verb 'to be', all of which occur in the text. These must be replaced by the lemma 'be'. (These eight forms do not include modal expressions such as 'will be' or 'would be'.)

Your tasks:

1. Stem these words by removing all instances of `'s`.
2. Lemmatize all the 8 forms of the verb 'to be' by replacing them by their stem 'be'. Be sure to ONLY replace separate words, not substrings of other words, i.e., don't change 'mistake' to 'mbetake'!
3. Put the result in `pirates_txt_02` and print out the first 2000 characters.

In [10]:
```
# Your code here
pirates_txt_02 = re.sub(r"'s\b", "", pirates_txt_02)
to_be_forms = ["am", "is", "are", "was", "were", "being", "been","be"]
for form in to_be_forms:
    pirates_txt_02 = re.sub(rf"\b{form}\b", "be", pirates_txt_02)
print(pirates_txt_02[:2000])
```

```
elizabeth swann: will
elizabeth swann: why be this happening
will turner: i do not know  you look beautiful
elizabeth swann: i think it bad luck for the groom to see the bride before t
he wedding
lord cutler beckett: governor weatherby swann  it be too long
lord cutler beckett: his lord now    actually
lord cutler beckett: in fact  i  do   mister mercer  the warrant for the arr
est of one william turner
lord cutler beckett: oh  be it  that annoying  my mistake  arrest her
elizabeth swann: on what charges
will turner: no
lord cutler beckett: ah-ha  here the one for william turner  and i have anot
her one for a mister james norrington  be he present
elizabeth swann:  what be the charges
lord cutler beckett: i do not believe that the answer to the question i aske
d
will turner: lord beckett  in the category of questions  not  answered
elizabeth swann: we be under the jurisdiction of the king governor of port r
oyal  and you will tell us what we be charged with
lord cutler beckett: for which the punishment  regrettably  be  also  death
perhaps you remember a certain pirate named jack sparrow
elizabeth swann:  captain  jack sparrow
lord cutler beckett: captain jack sparrow  yes  i thought you might
gibbs: fifteen men on a dead man chest  yo ho ho  and a  bottle  of rum  dri
nk and the devil had done for the rest  yo ho ho  and a  bottle  of rum  ha-
ha-ha-ha-ha
jack sparrow: sorry  mate
jack sparrow: mind if we make a little side trip  i did not think so
gibbs: not  quite  according to plan
jack sparrow: complications arose  ensued  be overcome
gibbs: you got what you went in for  then
jack sparrow: mm-hmm
gibbs: captain  i think the crew  meaning me as well  be expecting something
a bit more     shiny   what with the isla de muerta going all pear shaped  r
eclaimed by the sea  and the treasure with it
leech: and the royal navy chasing us all around the atlantic
marty: and the hurricane  aye
crew: aye  aye
gibbs: all in all  it seems some
```

Problem Three (30 points): Removing Stop Words, Tokenizing, and Creating the BOW Models

3.A Removing Stop Words (10 pts)

"Stop words" are common words which do not give much information about a text, since they occur in almost all texts. There is a standard set of such words which can be accessed through NLTK (notice that these include some with apostrophes, which we will have already removed):

In [11]:
```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're
", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', '
he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it'
, "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had
', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with
', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'af
ter', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off'
, 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'whe
n', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'mos
t', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so'
, 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 's
hould', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", '
hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan'
t", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "wo
n't", 'wouldn', "wouldn't"]
```

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

For the first part of this problem, you must **remove all stop words from the text,** and store the result in `pirates_txt_03`. However, since `will` is the name of a character in the script, **do NOT remove the stopword `will`**! Make SURE that you only remove words, and not substrings of larger words, e.g., do not remove all occurrences of the character `i` from the text just because the word `i` is a stop word! Replace stop words with single blanks to preserve the word boundaries.

Put your code in the next cell and print out the first 2000 characters.

In [12]:
```
# Your code here
s_words=(stopwords.words('english'))
s_words.remove('will')
lines = pirates_txt_02.split('\n')
empty_set2 = []
for line in lines:
    for word in s_words:
        line = re.sub(rf"\b{word}\b\s*", ' ', line)
    #I have again process to remove any charectors like - and ' that have oc
    #This makes the text more clean , i believe [NOTE : havent removed the c
    line = re.sub(r"(^|\s)-(|\s)", ' ', re.sub(r"(?<=\s)'\s", ' ', re.sub(r"
    empty_set2.append(line.strip())
pirates_txt_03 = '\n'.join(empty_set2)
print(pirates_txt_03[:2000])
```

elizabeth swann: will
elizabeth swann: happening
will turner: know look beautiful
elizabeth swann: think bad luck groom see bride wedding
lord cutler beckett: governor weatherby swann long
lord cutler beckett: lord actually
lord cutler beckett: fact mister mercer warrant arrest one william t
urner
lord cutler beckett: oh annoying mistake arrest
elizabeth swann: charges
will turner:
lord cutler beckett: ah-ha one william turner another one mister james
norrington present
elizabeth swann: charges
lord cutler beckett: believe answer question asked
will turner: lord beckett category questions answered
elizabeth swann: jurisdiction king governor port royal will tell us
charged
lord cutler beckett: punishment regrettably also death perhaps rememb
er certain pirate named jack sparrow
elizabeth swann: captain jack sparrow
lord cutler beckett: captain jack sparrow yes thought might
gibbs: fifteen men dead man chest yo ho ho bottle rum drink devil d
one rest yo ho ho bottle rum ha-ha-ha-ha-ha
jack sparrow: sorry mate
jack sparrow: mind make little side trip think
gibbs: quite according plan
jack sparrow: complications arose ensued overcome
gibbs: got went
jack sparrow: mm-hmm
gibbs: captain think crew meaning well expecting something bit shi
ny isla de muerta going pear shaped reclaimed sea treasure
leech: royal navy chasing us around atlantic
marty: hurricane aye
crew: aye aye
gibbs: seems time since speck honest pirating
jack sparrow: shiny
gibbs: aye shiny
jack sparrow: feeling perhaps dear old jack serving best interests
captain
cotton parrot: awk walk plank
jack sparrow: bird say
leech: blame bird show us piece cloth
jack sparrow: ohhh
gibbs: know good
jack sparrow:
marty: key
jack sparrow: much better drawing key
jack sparrow: gentlemen keys
leech: keys unlock things
gibbs: whatever key

3.B Tokenizing and Creating the BOW Dictionary (20 pts)

What we wish to do is to create a BOW model with a dictionary for two characters in the script, `elizabeth swann` and `jack sparrow`.

Part 3.B.1 (2 pts)

Using `split(...)`, split the text on the newlines `\n` to get a list of each line as a string. Print out the first 10 lines.

In [13]:
```
# Your code here
lines = pirates_txt_03.split('\n')
for line in lines[:10]:
    print(line)
```

```
elizabeth swann: will
elizabeth swann:  happening
will turner:    know   look beautiful
elizabeth swann:  think  bad luck  groom  see  bride  wedding
lord cutler beckett: governor weatherby swann     long
lord cutler beckett:  lord  actually
lord cutler beckett:  fact    mister mercer   warrant  arrest  one william t
urner
lord cutler beckett: oh   annoying   mistake  arrest
elizabeth swann:  charges
will turner:
```

Part 3.B.2 (18 pts)

Create a dictionary to hold the BOW models for these two characters, each being a `defaultdict` with a default value of 0 (this is a representation of the sparce matrix representing the BOW for the character).

Then go through the lines and calculate the frequency of each word spoken by that character. Print out the 20 most common words spoken by each character and the number of times spoken.

Hint: Scan through the lines created in 3.B.1, and just check if the line contains that character's name. Hint: you can use `in` to check if a substring occurs in a string.

```
   'wayne snyder:' in 'wayne snyder: hi there folks!'   =>
True
```

In order to make sure you are not finding an instance of the character's name as spoken by another character, be sure to include the colon (that's why we left it there!).

Then split the line on blanks, and add all but the first two words (the name of the character) to the BOW for that character. If the empty word '' occurs, ignore it (do not add it to the BOW).

In [14]:
```
from collections import defaultdict

elizabeth_bow = defaultdict(int)
jack_bow = defaultdict(int)

for line in lines:
    if 'elizabeth swann' in line or 'jack sparrow' in line:
        words = line.split()[2:]
        target_bow = elizabeth_bow if 'elizabeth swann' in line else jack_bo
        for word in words:
            if word != '':
                target_bow[word] += 1

#lines : just for exploring
```

In [15]:
```
# Elizabeth Swann's BOW (9 pts)

# Your code here
print("Elizabeth Swann's 20 most common words:")
from operator import itemgetter
for word, freq in sorted(elizabeth_bow.items(), key=itemgetter(1), reverse=T
    print(f"{word}: {freq}")
```

```
Elizabeth Swann's 20 most common words:
will: 18
jack: 12
know: 7
oh: 7
want: 6
find: 6
man: 6
good: 5
would: 4
something: 4
chance: 4
us: 3
sparrow: 3
compass: 3
give: 3
going: 3
way: 3
yes: 3
happening: 2
see: 2
```

In [16]:
```
# Jack Sparrows's BOW   (9 pts)
# Your code here
from operator import itemgetter
print("\nJack Sparrow's 20 most common words:")
for word, freq in sorted(jack_bow.items(), key=itemgetter(1), reverse=True)[
    print(f"{word}: {freq}")
```

```
Jack Sparrow's 20 most common words:
jack: 27
sparrow: 22
want: 14
captain: 11
come: 11
know: 10
oh: 8
will: 8
jones: 8
ship: 8
chest: 8
dirt: 8
say: 7
bugger: 7
one: 7
love: 7
mate: 6
key: 6
would: 6
oy: 6
```

In [ ]: