Zahra Jalilpour

Email: h19zahja@du.se

Report of AMI23B – Business Intelligence Lab4

Introduction:
The goal of this project is applying sentiment analysis and text mining on the scripts characters' dialogue in three episode of star wars movie.

Data description:
At first we load the datasets.

Eoisode No#	Dimension
episodeIV	(1011 , 3)
episodeV	(840 , 3)
episodeIVI	(675 , 3)

The first five rows in dataset of episodeIV is as below:

```
episodeIV.head()
```

	index	character	dialogue
0	character	dialogue	NaN
1	1	THREEPIO	Did you hear that? They've shut down the main...
2	2	THREEPIO	We're doomed!
3	3	THREEPIO	There'll be no escape for the Princess this time.
4	4	THREEPIO	What's that?

For cleaning the dataset, we search for missing value by using df.isnull(). These datasets do not consist of any missing value. We just remove the first row.
1- For finding the character with most dialogues, I apply value_counts() to the character column, and make new data frame and put the number of dialogue and the name of character.

episodeIV

	character
LUKE	254
HAN	153
THREEPIO	119
BEN	82
LEIA	57
VADER	41
RED LEADER	37
BIGGS	34
TARKIN	28
OWEN	25

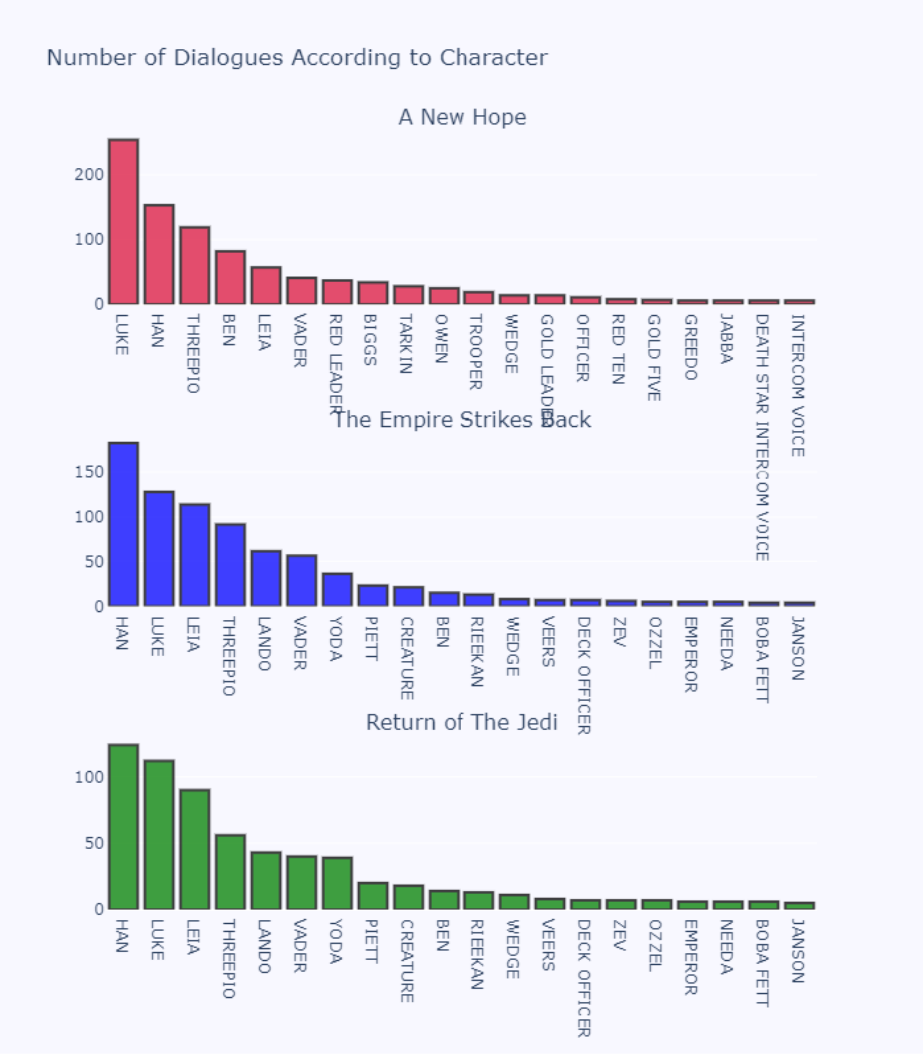
episodeV

	character
HAN	182
LUKE	128
LEIA	114
THREEPIO	92
LANDO	61
VADER	56
YODA	36
PIETT	23
CREATURE	21
BEN	15

episodeVI

	character
HAN	124
LUKE	112
THREEPIO	90
LEIA	56
VADER	43
LANDO	40
EMPEROR	39
JABBA	20
BEN	18
ACKBAR	14

2- For plotting the number of dialogue according to the character for each episode, I use plotly.graph_objects library in Python:



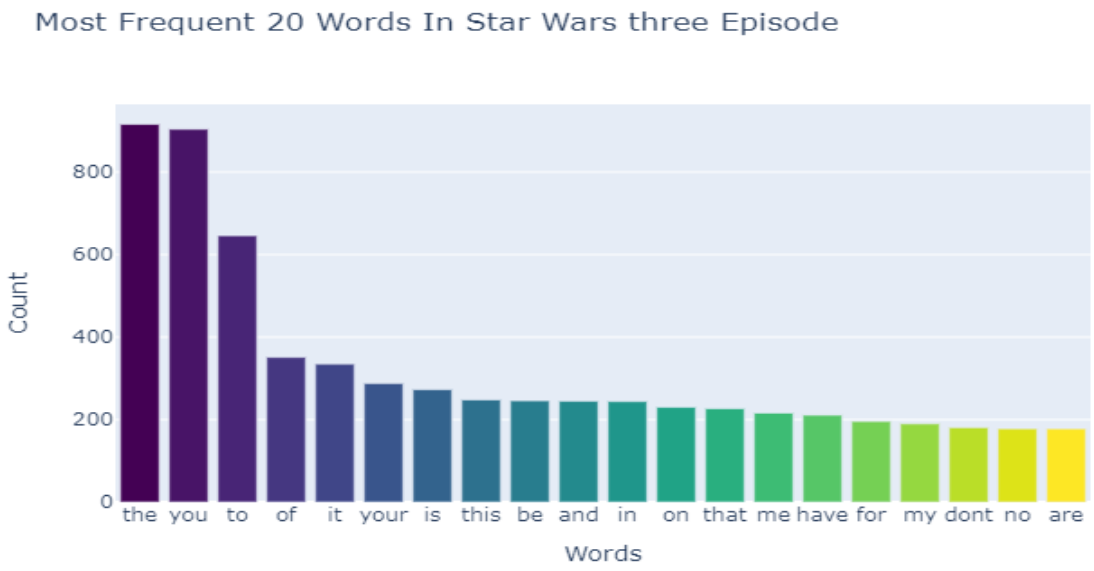
3- For adding a new column "episode" , I put the name of each episode in the new column for each dataset, and by pd.concat(), I concatenate three episides into a new dataset name data1. We can see the name of each episode below the new column and all three datasets are concatenated to one new dataset.

	character	dialogue	episode
1046	LEIA	You're imagining things.	The Empire Strikes Back
748	LEIA	We ran into some old friends.	A New Hope
644	LEIA	Somebody has to save our skins. Into the garb...	A New Hope
1307	HAN	Come on, Chewie!	The Empire Strikes Back
1101	HAN	Hang on, kid.	The Empire Strikes Back
2410	STORMTROOPER	Freeze! Don't move!	Return of The Jedi
169	THREEPIO	I'm sorry, sir, but he appears to have picked ...	A New Hope
490	LUKE	It sure is leaving in a big hurry. If they id...	A New Hope
1428	EMPEROR	We have a new enemy -- Luke Skywalker.	The Empire Strikes Back
1326	THREEPIO	Excuse me, ma'am, but where are we going?	The Empire Strikes Back

4- Our new dataset's name is data1. To discover the frequency distribution of words in new dataset, I use CountVectorizer from sklearn.feature_extraction.text. Here I do not remove any punctuation and stop words, and here we see the word with its frequency.


```
{'did': 687,
 'you': 2757,
 'hear': 1145,
 'that': 2430,
 'they': 2442,
 've': 2620,
 'shut': 2151,
 'down': 738,
 'the': 2432,
 'main': 1452,
 'reactor': 1901,
 'we': 2671,
 'll': 1404,
 'be': 220,
 'destroyed': 665,
 'for': 964,
 'sure': 2353,
 'this': 2450,
 'is': 1296,
```


5-To create frequency distribution plot of most repeated words, I create a new function to count word appearances in text, get word indices and counts, create wordvectors to inverse-transform them, inverse-transfrom the wordvectors and return word and word-counts



6-a- For text mining by NLTK library we import stopwords from nltk.corpus and import re library. I convert uppercase to lower case, remove stopwords, apply tokenization and so on.

```
['hear shut main reactor destroyed sure madness',
 'doomed',
 'escape princess time',
 '',
 'known better trust logic half sized thermocapsulary dehousing assister',
 'hurry come waiting get gear',
 'artoo artoo detoo',
 'last',
 'heading direction going sent spice mine kessel smashed know',
 'wait minute going',
 'death star plan main computer',
 'transmission intercepted',
 'intercepted transmission aaah consular ship diplomatic mission',
 'consular ship ambassador',
 'commander tear ship apart found plan bring ambassador want alive',
 'set stun',
 'right inform lord vader prisoner',
 'hey permitted restricted deactivated sure',
 'call mindless philosopher overweight glob grease come somebody see',
```

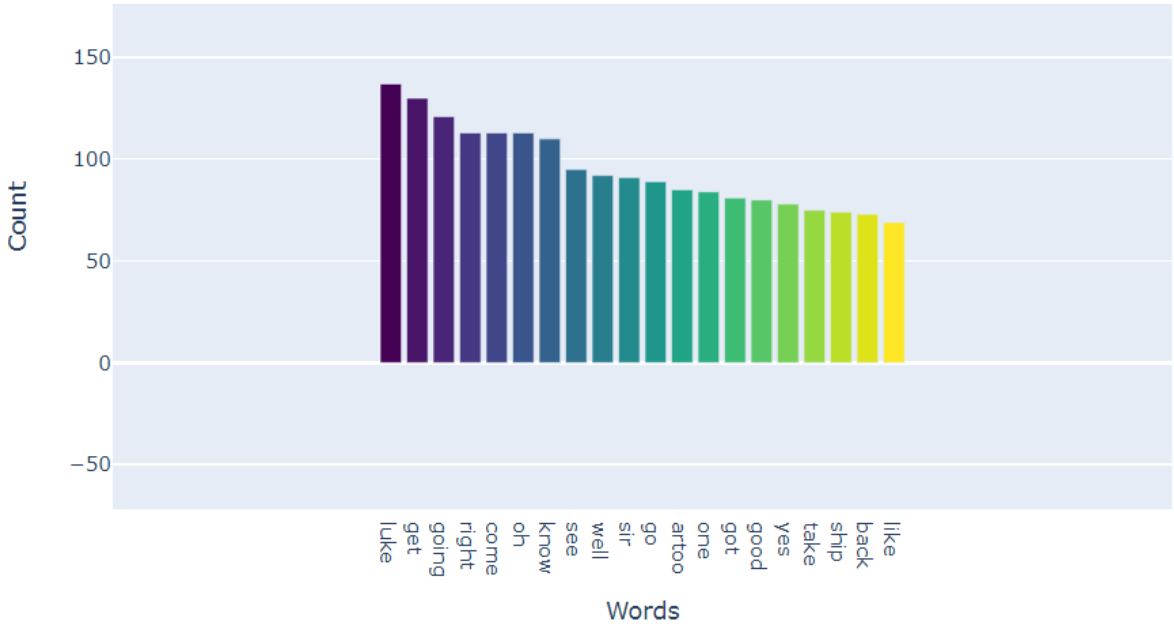
6-b- Then I add the resulting array list to the dataset as a new column "new_script". In the new script column we can see the stopwords are removed, like did, you, they,...

	character	dialogue	episode	new_script
0	THREEPIO	Did you hear that? They've shut down the main...	A New Hope	hear shut main reactor destroyed sure madness
1	THREEPIO	We're doomed!	A New Hope	doomed
2	THREEPIO	There'll be no escape for the Princess this time.	A New Hope	escape princess time
3	THREEPIO	What's that?	A New Hope	
4	THREEPIO	I should have known better than to trust the l...	A New Hope	known better trust logic half sized thermocaps...
...	...	...	...	...
2518	LANDO	Wedge, I don't think we're going to make it.	Return of The Jedi	wedge think going make
2519	WEDGE	You'll make it. Just follow me Gold Leader.	Return of The Jedi	make follow gold leader
2520	LANDO	I promised to return his ship without a scratc...	Return of The Jedi	promised return ship without scratch sure hope...
2521	HAN	Lando...	Return of The Jedi	lando
2522	THREEPIO	They did it!	Return of The Jedi	

7-4-In the new script column I should find the frequency distribution of words. we will do it by CountVectorizer(). Here the words that we see differ from part4, because we remove stopwords and so on.

```
{'hear': 1031,
 'shut': 1923,
 'main': 1304,
 'reactor': 1701,
 'destroyed': 600,
 'sure': 2109,
 'madness': 1299,
 'doomed': 658,
 'escape': 737,
 'princess': 1634,
 'time': 2209,
 'known': 1208,
 'better': 231,
 'trust': 2272,
 'logic': 1269,
 'half': 1008,
 'sized': 1952,
 'thermocapsulary': 2183,
 'dehousing': 575,
```

7-5-Frequency Distribution plot of the most repeated words in new script column.



8- For wordcloud and wordcloud mask, I import Image from PIL library, and import wordcloud and stopwords from wordcloud library.

mask of vader:
The most repeated words by Vader are: master, want, rebel, find, ship,force,obi,admiral,...

mask of yoda:
The most repeated words by Yoda are: force, mind, jedi, must, luke, vader,old,yes,..

9- For discovering the most repeated words, for example here I select, max_feature=40,
I import Countvectorizer from sklearn.feature_extraction.text, and below we see the most repeatd words:

```
mosltly used 40 words: ['artoo', 'chewie', 'come', 'father', 'force', 'friend', 'going', 'good', 'go
t', 'help', 'jedi', 'know', 'leia', 'let', 'like', 'little', 'look', 'lord', 'luke', 'make', 'maste
r', 'need', 'oh', 'old', 'power', 'right', 'say', 'shield', 'ship', 'sir', 'sure', 'tell', 'thing',
'think', 'time', 'vader', 'wait', 'want', 'way', 'yes']
```

Now we want to find word frequency with TFIDFvectorizer.
TF = Term Frequency: This summarizes how often a given word appears within a document.
IDF=Inverse Document Frequency: This downscales words that appear a lot across documents. TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. fre**quen**t in a document but not across documents.

	tfidf
madness	0.473183
reactor	0.400363
destroyed	0.385755
shut	0.360072
hear	0.356062
...	...
fearless	0.000000
feeble	0.000000
feel	0.000000
feeling	0.000000
zone	0.000000

2450 rows × 1 columns

The scores above make sense. The more common the word across documents, the lower its score and the more unique a word is to our first document (e.g. ‘madness’ and ‘reactor) the higher the score.
The second method to find the most relevant words is using TfidfTransformer.
In order to start using TfidfTransformer I will first have to create a CountVectorizer to count the number of words (term frequency).
Dimension of counted words is : `(2523, 2450)`

Then apply **tfidf_transformer**.fit() to the counted words and put the IDF value in a new dataframe. The values will be sorted in ascending order.

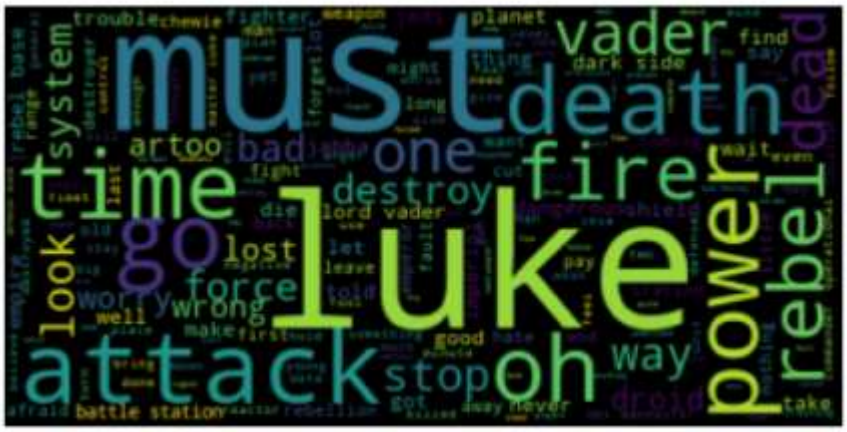
	tfidf
madness	0.473183
reactor	0.400363
destroyed	0.385755
shut	0.360072
hear	0.356062
...	...
fearless	0.000000
feeble	0.000000
feel	0.000000
feeling	0.000000
zone	0.000000

2450 rows × 1 columns

10- For performing sentiment analysis, I use "**vader_lexicon**" and import SentimentIntensityAnalyzer from nltk.sentiment.vader, and creat a new column name score. Vades sentiment analysis gives each word a score. Then put the average of scores in new column named compound. The range of the compound is [-1, +1], then divide this range to three categories, positive, negative, neutral. Then remove specific unimportant words. Finally apply wordcloud for positive and negative words.

Positive emotion words:

Negative emotion words:



By python libraries, we can not indicate which character has a bad feeling or belongs to the dark side. But by using R and a dictionary `nrc` we can indicate the personality of each character. NRC categorizes words in a binary fashion into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.