**Exploratory Data Analysis**

**Introduction**

After the data cleaning step where we put our data into a few standard formats, the next step is to take a look at the data and see if what we're looking at makes sense. Before applying any fancy algorithms, it's always important to explore the data first.

When working with numerical data, some of the exploratory data analysis (EDA) techniques we can use include finding the average of the data set, the distribution of the data, the most common values, etc. The idea is the same when working with text data. We are going to find some more obvious patterns with EDA before identifying the hidden patterns with machines learning (ML) techniques. We are going to look at the following for each comedian:

1. **Most common words** - find these and create word clouds
2. **Size of vocabulary** - look number of unique words and also how quickly someone speaks
3. **Amount of profanity** - most common terms

## Most Common Words

### Analysis[¶](https://render.githubusercontent.com/view/ipynb?commit=b9bfd3055ba188c55824d99f971be318a2816eef&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f61646173686f66646174612f6e6c702d696e2d707974686f6e2d7475746f7269616c2f623962666433303535626131383863353538323464393966393731626533313861323831366565662f322d4578706c6f7261746f72792d446174612d416e616c797369732e6970796e62&nwo=adashofdata%2Fnlp-in-python-tutorial&path=2-Exploratory-Data-Analysis.ipynb&repository_id=142180641&repository_type=Repository#Analysis)

*# Read in the document-term matrix*

**import** **pandas** **as** **pd**

data = pd.read\_pickle('dtm.pkl')

data = data.transpose()

data.head()

*# Find the top 30 words said by each comedian*

top\_dict = {}

**for** c **in** data.columns:

top = data[c].sort\_values(ascending=**False**).head(30)

top\_dict[c]= list(zip(top.index, top.values))

top\_dict

*# Print the top 15 words said by each comedian*

**for** comedian, top\_words **in** top\_dict.items():

print(comedian)

print(', '.join([word **for** word, count **in** top\_words[0:14]]))

print('---')

**NOTE:** At this point, we could go on and create word clouds. However, by looking at these top words, you can see that some of them have very little meaning and could be added to a stop words list, so let's do just that.

*# Look at the most common top words --> add them to the stop word list*

**from** **collections** **import** Counter

*# Let's first pull out the top 30 words for each comedian*

words = []

**for** comedian **in** data.columns:

top = [word **for** (word, count) **in** top\_dict[comedian]]

**for** t **in** top:

words.append(t)

words

*# Let's aggregate this list and identify the most common words along with how many routines they occur in*

Counter(words).most\_common()

*# If more than half of the comedians have it as a top word, exclude it from the list*

add\_stop\_words = [word **for** word, count **in** Counter(words).most\_common() **if** count > 6]

add\_stop\_words

*# Let's update our document-term matrix with the new list of stop words*

**from** **sklearn.feature\_extraction** **import** text

**from** **sklearn.feature\_extraction.text** **import** CountVectorizer

*# Read in cleaned data*

data\_clean = pd.read\_pickle('data\_clean.pkl')

*# Add new stop words*

stop\_words = text.ENGLISH\_STOP\_WORDS.union(add\_stop\_words)

*# Recreate document-term matrix*

cv = CountVectorizer(stop\_words=stop\_words)

data\_cv = cv.fit\_transform(data\_clean.transcript)

data\_stop = pd.DataFrame(data\_cv.toarray(), columns=cv.get\_feature\_names())

data\_stop.index = data\_clean.index

*# Pickle it for later use*

**import** **pickle**

pickle.dump(cv, open("cv\_stop.pkl", "wb"))

data\_stop.to\_pickle("dtm\_stop.pkl")

*# Let's make some word clouds!*

*# Terminal / Anaconda Prompt: conda install -c conda-forge wordcloud*

**from** **wordcloud** **import** WordCloud

wc = WordCloud(stopwords=stop\_words, background\_color="white", colormap="Dark2",

max\_font\_size=150, random\_state=42)

*# Reset the output dimensions*

**import** **matplotlib.pyplot** **as** **plt**

plt.rcParams['figure.figsize'] = [16, 6]

full\_names = ['Ali Wong', 'Anthony Jeselnik', 'Bill Burr', 'Bo Burnham', 'Dave Chappelle', 'Hasan Minhaj',

'Jim Jefferies', 'Joe Rogan', 'John Mulaney', 'Louis C.K.', 'Mike Birbiglia', 'Ricky Gervais']

*# Create subplots for each comedian*

**for** index, comedian **in** enumerate(data.columns):

wc.generate(data\_clean.transcript[comedian])

plt.subplot(3, 4, index+1)

plt.imshow(wc, interpolation="bilinear")

plt.axis("off")

plt.title(full\_names[index])

plt.show()

### **Findings**

* Ali Wong says the s-word a lot and talks about her husband. I guess that's funny to me.
* A lot of people use the F-word. Let's dig into that later.

## Number of Words

### **Analysis**

*# Find the number of unique words that each comedian uses*

*# Identify the non-zero items in the document-term matrix, meaning that the word occurs at least once*

unique\_list = []

**for** comedian **in** data.columns:

uniques = data[comedian].nonzero()[0].size

unique\_list.append(uniques)

*# Create a new dataframe that contains this unique word count*

data\_words = pd.DataFrame(list(zip(full\_names, unique\_list)), columns=['comedian', 'unique\_words'])

data\_unique\_sort = data\_words.sort\_values(by='unique\_words')

data\_unique\_sort

*# Calculate the words per minute of each comedian*

*# Find the total number of words that a comedian uses*

total\_list = []

**for** comedian **in** data.columns:

totals = sum(data[comedian])

total\_list.append(totals)

*# Comedy special run times from IMDB, in minutes*

run\_times = [60, 59, 80, 60, 67, 73, 77, 63, 62, 58, 76, 79]

*# Let's add some columns to our dataframe*

data\_words['total\_words'] = total\_list

data\_words['run\_times'] = run\_times

data\_words['words\_per\_minute'] = data\_words['total\_words'] / data\_words['run\_times']

*# Sort the dataframe by words per minute to see who talks the slowest and fastest*

data\_wpm\_sort = data\_words.sort\_values(by='words\_per\_minute')

data\_wpm\_sort

*# Let's plot our findings*

**import** **numpy** **as** **np**

y\_pos = np.arange(len(data\_words))

plt.subplot(1, 2, 1)

plt.barh(y\_pos, data\_unique\_sort.unique\_words, align='center')

plt.yticks(y\_pos, data\_unique\_sort.comedian)

plt.title('Number of Unique Words', fontsize=20)

plt.subplot(1, 2, 2)

plt.barh(y\_pos, data\_wpm\_sort.words\_per\_minute, align='center')

plt.yticks(y\_pos, data\_wpm\_sort.comedian)

plt.title('Number of Words Per Minute', fontsize=20)

plt.tight\_layout()

plt.show()

### Findings

* **Vocabulary**
  + Ricky Gervais (British comedy) and Bill Burr (podcast host) use a lot of words in their comedy
  + Louis C.K. (self-depricating comedy) and Anthony Jeselnik (dark humor) have a smaller vocabulary
* **Talking Speed**
  + Joe Rogan (blue comedy) and Bill Burr (podcast host) talk fast
  + Bo Burnham (musical comedy) and Anthony Jeselnik (dark humor) talk slow

Ali Wong is somewhere in the middle in both cases. Nothing too interesting here.

## Amount of Profanity

### **Analysis**[¶](https://render.githubusercontent.com/view/ipynb?commit=b9bfd3055ba188c55824d99f971be318a2816eef&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f61646173686f66646174612f6e6c702d696e2d707974686f6e2d7475746f7269616c2f623962666433303535626131383863353538323464393966393731626533313861323831366565662f322d4578706c6f7261746f72792d446174612d416e616c797369732e6970796e62&nwo=adashofdata%2Fnlp-in-python-tutorial&path=2-Exploratory-Data-Analysis.ipynb&repository_id=142180641&repository_type=Repository#Analysis)

*# Earlier I said we'd revisit profanity. Let's take a look at the most common words again.*

Counter(words).most\_common()

*# Let's isolate just these bad words*

data\_bad\_words = data.transpose()[['fucking', 'fuck', 'shit']]

data\_profanity = pd.concat([data\_bad\_words.fucking + data\_bad\_words.fuck, data\_bad\_words.shit], axis=1)

data\_profanity.columns = ['f\_word', 's\_word']

data\_profanity

*# Let's create a scatter plot of our findings*

plt.rcParams['figure.figsize'] = [10, 8]

**for** i, comedian **in** enumerate(data\_profanity.index):

x = data\_profanity.f\_word.loc[comedian]

y = data\_profanity.s\_word.loc[comedian]

plt.scatter(x, y, color='blue')

plt.text(x+1.5, y+0.5, full\_names[i], fontsize=10)

plt.xlim(-5, 155)

plt.title('Number of Bad Words Used in Routine', fontsize=20)

plt.xlabel('Number of F Bombs', fontsize=15)

plt.ylabel('Number of S Words', fontsize=15)

plt.show()

### **Findings**

* **Averaging 2 F-Bombs Per Minute!** - I don't like too much swearing, especially the f-word, which is probably why I've never heard of Bill Bur, Joe Rogan and Jim Jefferies.
* **Clean Humor** - It looks like profanity might be a good predictor of the type of comedy I like. Besides Ali Wong, my two other favorite comedians in this group are John Mulaney and Mike Birbiglia.

**Side Note**

What was our goal for the EDA portion of our journey? **To be able to take an initial look at our data and see if the results of some basic analysis made sense.**

My conclusion - yes, it does, for a first pass. There are definitely some things that could be better cleaned up, such as adding more stop words or including bi-grams. But we can save that for another day. The results, especially the profanity findings, are interesting and make general sense, so we're going to move on.

As a reminder, the data science process is an interative one. It's better to see some non-perfect but acceptable results to help you quickly decide whether your project is a dud or not, instead of having analysis paralysis and never delivering anything.

**Alice's data science (and life) motto: Let go of perfectionism!**

**Additional Exercises**

1. What other word counts do you think would be interesting to compare instead of the f-word and s-word? Create a scatter plot comparing them.