# KMeans and LSA Analysis

In this project we will look into some MEDLINE transcript data of healthcare professionals, and compare how a kmeans model differs when first tranformed with an LSA SVD model.

The code, along with the files necessary and versions of packages in this instance can be found on this repo: https://github.com/Benjamin-Siebold/MSDS-682-Text-Analytics (https://github.com/Benjamin-Siebold/MSDS-682-Text-Analytics)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import re
import nltk
import spacy
import string

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import TruncatedSVD
import sklearn_extra


nlp = spacy.load('en_core_web_lg')
nltk.download('punkt')
stopwords = nltk.corpus.stopwords.words('english')
np.random.seed(50)
```

```
[nltk_data] Downloading package punkt to /Users/z004g33/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

## 1 - Import data, and clean text

The first step in this anlayis is to import the data we are interested in, and clean the text for topic modeling. This may take multiple iterations, as we investigate the data, see what it consists of, and remove words that may be unecessary for classification.

In [2]:
```
def clean_text(docs):

    print('remove classifications')
    docs = [re.sub(r'\b[A-Z]+\b', '', doc) for doc in docs]

    print('removing punc')
    table = str.maketrans({key: None for key in string.punctuation + string
    clean_docs = [d.translate(table) for d in docs]

    print('nlp')
    nlp_docs = [nlp(doc) for doc in clean_docs]

    print('lemmatize')
    lemm_docs = [[w.lower_ if w.lemma_ == '-PRON-'
                  else w.lemma_ for w in doc]
                 for doc in nlp_docs]

    print('remove stopwords')
    lemm_docs = [[lemma for lemma in doc if lemma not in stopwords] for doc

    print('combine words')
    cleaned_docs = [' '.join(word) for word in lemm_docs]

    print('remove leading/trailing spaces')
    cleaned_docs = [doc.strip() for doc in cleaned_docs]

    return cleaned_docs
```

In [3]:
```
transcript_data = pd.read_csv('mtsamples.csv')
```

In [4]:
```
transcript = pd.DataFrame(transcript_data['transcription'])
```

In [5]:
```
transcript = transcript.dropna()
```

In [6]:
```
t1 = transcript['transcription']
```

In [7]:
```
cleaned_transcription = clean_text(t1)
```

```
remove classifications
removing punc
nlp
lemmatize
remove stopwords
combine words
remove leading/trailing spaces
```

In [8]:
```
single_string = ' '.join(cleaned_transcription)
ranking = nltk.FreqDist(single_string.split())
ranking_df = pd.DataFrame.from_dict(ranking, orient='index')
ranking_df.sort_values(by=[0], ascending=False).head(10)
```

Out[8]:

| | 0 |
|---|---|
| **patient** | 23999 |
| **right** | 11087 |
| **use** | 8978 |
| **place** | 8298 |
| **left** | 7126 |
| **normal** | 6979 |
| **well** | 6388 |
| **pain** | 5922 |
| **note** | 4910 |
| **time** | 4683 |

In [9]:
```
stopwords = set(stopwords + ['patient'])
```

In [10]:
```
cleaned_transcription = clean_text(t1)
```

```
remove classifications
removing punc
nlp
lemmatize
remove stopwords
combine words
remove leading/trailing spaces
```

## 2 - Apply KMEANs

With the data cleaned, we will first go through the process of applying a kmeans model to the data. This first section will mirror the week 5 analysis project.

In [11]:
```
vect = TfidfVectorizer(min_df=1500)
features = vect.fit_transform(cleaned_transcription)
features.shape
```

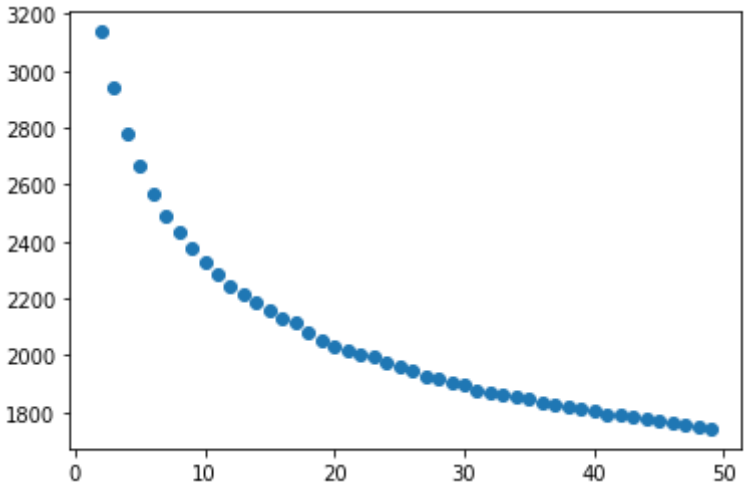
Out[11]:
```
(4966, 35)
```

In [12]:
```
features = features.todense()
```

In [13]:
```
wss = []
for n in range(2,50):
    model = KMeans(n_clusters = n, random_state = 50, n_jobs=-1)
    model.fit(features)
    wss.append(-model.score(features))
```

In [14]:
```
plt.scatter(range(2,50),wss)
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x1aa8cfdc50>
```




In [15]:
```
model = KMeans(n_clusters=18, random_state=50, n_jobs=-1)
model.fit(features)
cluster_labels = model.predict(features)
```

In [16]:
```
transcripts = pd.DataFrame(cleaned_transcription)
clusters_df = pd.DataFrame(cluster_labels, columns=['prediction'])
```

In [17]:
```
clustered_transcripts = pd.concat([transcripts, clusters_df], axis=1, join=
```

In [18]:
```
clustered_transcripts.head(10)
```

Out[18]:

| | 0 | prediction |
|---|---|---|
| 0 | yearold white female present complaint allergy... | 9 |
| 1 | difficulty climb stair difficulty airline seat... | 12 |
| 2 | see today pleasant gentleman year old ... | 12 |
| 3 | leave atrial enlargement left atrial diameter ... | 17 |
| 4 | left ventricular cavity size wall thickness ap... | 2 |
| 5 | morbid obesity morbid obesity Laparoscopi... | 14 |
| 6 | deformity right breast reconstruction excess... | 14 |
| 7 | multiple view heart great vessel reveal normal... | 17 |
| 8 | Lipodystrophy abdomen thigh Lipodystrophy a... | 9 |
| 9 | normal cardiac chamber size normal left vent... | 17 |

In [19]:
```
### Taken from Week 5 solutions

clean_text_array=np.array(cleaned_transcription)
for i in range(18):
    cluster_docs=clean_text_array[cluster_labels==i]
    wordcounts=nltk.FreqDist(' '.join(cluster_docs).split())
    print('cluster', i)
    print('number of datapoints in cluster:', np.sum(cluster_labels==i))
    print('top 10 words and counts:')
    print(wordcounts.most_common()[:10])
```

```
move', 566), ('well', 520), ('suture', 514), ('procedure', 500), ('clos
e', 470), ('room', 468)]
cluster 2
number of datapoints in cluster: 332
top 10 words and counts:
[('left', 2177), ('right', 1260), ('artery', 1178), ('leave', 710), ('cor
onary', 657), ('normal', 615), ('use', 474), ('catheter', 454), ('diseas
e', 347), ('anterior', 339)]
cluster 3
number of datapoints in cluster: 236
top 10 words and counts:
[('blood', 739), ('pressure', 415), ('mg', 376), ('negative', 279), ('rat
e', 265), ('day', 255), ('heart', 217), ('show', 203), ('disease', 199),
('normal', 197)]

cluster 4
number of datapoints in cluster: 254
top 10 words and counts:
[('procedure', 706), ('use', 212), ('well', 201), ('place', 189), ('ris
k', 189), ('perform', 167), ('obtain', 152), ('position', 142), ('take',
```

## 3 - KMEANs Summary

From the above, one can see there isn't a great eblow in the kmeans curve, which makes it difficult to chose the number of clusters. There looks to be a very small elbow at 18 clusters, so 18 was chosen. We then look at the top words from each cluster, and find the clusters are hard to separate from eachother.

For example, the word "right" is in the top 5 words in 9 of the 18 clusters, and well is in the top 5 words in 11 of the 18 clusters. This makes it difficult to use top words to show what the clusters entail. This could potentially be fixed by removing certain words from the texts, but it is hard to know with healthcare and the english language if the word right is referring to the right side of the body, or if everything is all right; thus removing all instances of right could cause very negative impacts.

We also look at more specific words and can make some interpretations about the groupings. Cluster 5 seems to be about surgery to some capacity with words like "operating", "procedure", "incision" and "room", so the model isn't a total failure.

## 4 - Add LSA decomposition to see affects

Next, LSA will be applied to the data to try and apply a better model that gives more descriptive or unique models. First, we will apply the TruncatedSVD and then apply a transform to our features in hopes that the model will can accuracy.

In [20]:
```
svd_model = TruncatedSVD(n_components=18, algorithm='randomized', n_iter=-1

svd_model.fit(features)

len(svd_model.components_)
```

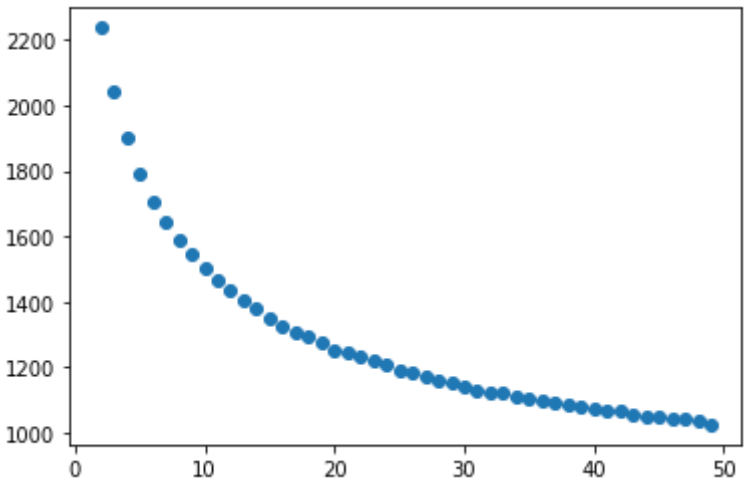
Out[20]: 18

In [21]:
```
LSA_model = svd_model.fit_transform(features)
```

In [22]:
```
wss = []
for n in range(2,50):
    model = KMeans(n_clusters = n, random_state = 50, n_jobs=-1)
    model.fit(LSA_model)
    wss.append(-model.score(LSA_model))
```

In [23]:
```
plt.scatter(range(2,50),wss)
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1aca2c6b50>
```




## 5 - Apply Best Model

Similar to the regular kmeans model, the curve does not have a hard elbow. The scores are overall significantly lower, which should give us a better model overall. There are two slight drops, one at 18, and one around 30. For consistancy in comparing models, we will choose 18 clusters again.

In [24]:
```
lsa_means_model = KMeans(n_clusters = 18, random_state = 50, n_jobs = -1)
lsa_means_model.fit(LSA_model)
lsa_clusters = lsa_means_model.predict(LSA_model)
```

In [25]:
```
lsa_clusters_df = pd.DataFrame(lsa_clusters, columns=['prediction'])
```

In [26]:
```
lsa_clustered_transcripts = pd.concat([transcripts, lsa_clusters_df], axis=
```

In [27]:
```
lsa_clustered_transcripts.head(10)
```

Out[27]:

| | 0 | prediction |
|---|---|---|
| 0 | yearold white female present complaint allergy... | 9 |
| 1 | difficulty climb stair difficulty airline seat... | 4 |
| 2 | see today pleasant gentleman year old ... | 4 |
| 3 | leave atrial enlargement left atrial diameter ... | 13 |
| 4 | left ventricular cavity size wall thickness ap... | 10 |
| 5 | morbid obesity morbid obesity Laparoscopi... | 1 |
| 6 | deformity right breast reconstruction excess... | 1 |
| 7 | multiple view heart great vessel reveal normal... | 13 |
| 8 | Lipodystrophy abdomen thigh Lipodystrophy a... | 9 |
| 9 | normal cardiac chamber size normal left vent... | 10 |

## 6 - Compare Models

Initially, it looks like the model may not be any different from the original KMeans model; however, we will create the top words from each cluster and look at the top 10 words in each cluster.

In [28]:
```
### Taken from Week 5 solutions

clean_text_array=np.array(cleaned_transcription)
for i in range(18):
    cluster_docs=clean_text_array[lsa_clusters==i]
    wordcounts=nltk.FreqDist(' '.join(cluster_docs).split())
    print('cluster', i+1)
    print('number of datapoints in cluster:', np.sum(lsa_clusters==i))
    print('top 10 words and counts:')
    print(wordcounts.most_common()[:10])
```

```
[('note', 858), ('normal', 299), ('well', 283), ('use', 213), ('time', 20
1), ('place', 195), ('incision', 189), ('right', 181), ('procedure', 15
6), ('without', 150)]
cluster 7
number of datapoints in cluster: 193
top 10 words and counts:
[('area', 740), ('left', 377), ('skin', 351), ('place', 315), ('procedur
e', 297), ('incision', 294), ('suture', 256), ('use', 254), ('right', 24
5), ('remove', 221)]
cluster 8
number of datapoints in cluster: 156
top 10 words and counts:
[('well', 602), ('mg', 259), ('day', 208), ('good', 170), ('daily', 154),
('normal', 148), ('follow', 142), ('give', 142), ('also', 140), ('clear',
139)]
cluster 9
number of datapoints in cluster: 241
top 10 words and counts:
[('time', 1010), ('mg', 450), ('history', 373), ('day', 371), ('report',
349), ('well', 335), ('normal', 297), ('use', 288), ('year', 283), ('als
```

## 7 - LSA Summary

After reading through the clusters above, a more thorough investigation of the original KMeans model is necessary. First we look at the LSA tranformed model and can make general assumptions about the first five clusters:

1. (artery, coronary, catheter, perform, place) all suggest this could be about hear procedures, or placing a stint?
2. (remove, suture, incision, mm) all suggest this could be about the cutting or incisions made and how the healed from an operation
3. (blood, pressure, normal, heart, history) all suggest this cluster involves patients with either previous heart issues, or being seen for blood pressure related issues
4. (pain, normal, deny, history) The 4th cluster is harder to understand what is occuring. The pain may be looking at patients with pain, but their aren't many other words to suggest specifics, and there are many general words.
5. (pain, back, deny, history) The 5th cluster has some overlap of the 4th top words, with back being a distinct difference that may suggest back pain vs another type of pain.

From there looking through the list of clusters is still difficult to fully distinguish what is different between the clusters. Overall, this model needs much improvement in order for it to be used in production