NMF Tuning Approach 3

Document = part between interludes

Imports

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import WordCloud, STOPWORDS
from gensim.utils import simple_preprocess,SaveLoad
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk.stem.porter import *
import time
from pickle import load, dump
import my_nmf_utils
from my_nmf_utils import my_nmf_model
from gensim.models import CoherenceModel
```

Initializations

In [2]:
```
np.random.seed(2018)
stemmer = SnowballStemmer('english')
```

In [3]:
```
stop_words = stopwords.words('english')
stop_words.extend(STOPWORDS)
stop_words2 = stop_words
```

Helpers

In [4]:
```
# Cleaning functions
def remove_stopwords(texts):
    return [word for word in texts if word not in stop_words ]

def stem_and_lemmatize(texts):
    return [stemmer.stem(WordNetLemmatizer().lemmatize(word, pos='v')) for word in texts]
```

Import Data

In [5]:
```
clean_data_folder = "../clean_data/"
with open("../clean_data/"+"characters_with_gender.pckl", "rb") as f2:
    characters = load(f2)
```

In [6]:
```
clean_data_folder_lda = "../clean_data_lda"
with open(clean_data_folder_lda+"/all_data.pkl", "rb") as f:
    all_scripts = load(f)
data_nmf = pd.merge(all_scripts, characters, on="character", how="inner")
```

In [7]:
```
all_scripts.reset_index(inplace=True, drop=True)
```

In [8]:
```
data = pd.merge(all_scripts, characters, on="character", how="inner")
```

In [9]:
```
data2 = data_nmf[['text', 'doc_id']]
data3 = data2.groupby(['doc_id'])['text'].agg(sum).reset_index()
documents = data3.text.apply(stem_and_lemmatize)
```

Tuning

Number of Topics

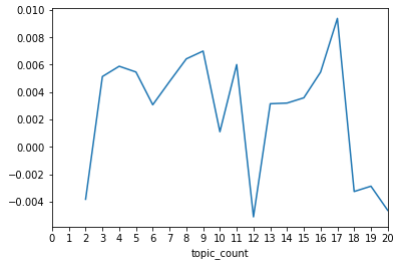
In [10]:
```
coherence_scores_nmpi = []
```

In [11]:
```
for i in range(2,21,1):
    nmf_model_x = my_nmf_model(documents)
    nmf_model_x.train(num_topics = i, passes = 5, tfidf_threshold = 0.0)
    coherence_model_nmf = CoherenceModel(model=nmf_model_x.nmf_model,
                                         texts=nmf_model_x.documents,
                                         dictionary=nmf_model_x.dictionary,
                                         coherence='c_npmi')
    coherence_nmf = coherence_model_nmf.get_coherence()
    coherence_scores_nmpi.append([i, coherence_nmf,nmf_model_x])
    print("coherence: ",[i, coherence_nmf])
```

```
coherence:  [2, -0.0038219776298660827]
coherence:  [3, 0.0051281807705889325]
coherence:  [4, 0.005878216759463891]
coherence:  [5, 0.005457957777626058]
coherence:  [6, 0.0030678964780150256]
coherence:  [7, 0.004762925758525366]
coherence:  [8, 0.006417130648771702]
coherence:  [9, 0.006985653328160192]
coherence:  [10, 0.0011019745253528933]
coherence:  [11, 0.006001431108803349]
coherence:  [12, -0.0051035381797437775]
coherence:  [13, 0.003144620506580966]
coherence:  [14, 0.0031925500299962418]
coherence:  [15, 0.0035761088395421746]
coherence:  [16, 0.005453375960921927]
coherence:  [17, 0.009367533715563543]
coherence:  [18, -0.0032572704312115387]
coherence:  [19, -0.0028694618074973995]
coherence:  [20, -0.00462992432798858]
```

In [13]:
```
df = pd.DataFrame(coherence_scores_nmpi)
df.columns = ["topic_count", "coherence", "model"]
df = df.set_index("topic_count")
df.coherence.plot(kind='line',
                  xticks=[x for x in range(21)],
                  xlim=[0,20])
df.coherence.idxmax()
```

Out[13]: 17



In [14]:
```
df.coherence[:6].idxmax()
```

Out[14]: 4

In [15]:
```
test model = df.model[9]
```

In [16]:
```
test model.nmf model.print topics()
```

Out[16]:
```
[(0,
  '0.128*"im" + 0.029*"think" + 0.028*"yeah" + 0.026*"say" + 0.023*"gonna" + 0.023*"sorri" + 0.016*"sure" + 0.013*"take" + 0.011*"thing" + 0.011*"someth"'),
 (1,
  '0.055*"colonel" + 0.023*"neil" + 0.019*"give" + 0.017*"general" + 0.017*"gate" + 0.015*"say" + 0.010*"let" + 0.009*"believ" + 0.009*"team" + 0.009*"peopl"'),
 (2,
  '0.157*"sir" + 0.026*"know" + 0.024*"yes" + 0.012*"captain" + 0.011*"right" + 0.010*"order" + 0.010*"general" + 0.009*"want" + 0.009*"need" + 0.008*"carter"'),
 (3,
  '0.073*"know" + 0.026*"one" + 0.012*"whi" + 0.011*"come" + 0.010*"thing" + 0.010*"take" + 0.010*"see" + 0.010*"want" + 0.009*"tell" + 0.008*"onli"'),
 (4,
  '0.119*"goal" + 0.018*"yes" + 0.017*"host" + 0.010*"whi" + 0.009*"believ" + 0.009*"jackson" + 0.009*"may" + 0.009*"human" + 0.008*"dr" + 0.008*"tell"'),
 (5,
  '0.240*"go" + 0.028*"let" + 0.020*"back" + 0.019*"tri" + 0.014*"wait" + 0.013*"daniel" + 0.011*"see" + 0.011*"tell" + 0.008*"earth" + 0.008*"kill"'),
 (6,
  '0.075*"teal" + 0.020*"one" + 0.018*"jack" + 0.017*"right" + 0.014*"hes" + 0.014*"carter" + 0.012*"find" + 0.011*"see" + 0.009*"door" + 0.009*"ffa"'),
 (7,
  '0.042*"think" + 0.030*"well" + 0.021*"us" + 0.016*"mean" + 0.013*"use" + 0.011*"yeah" + 0.010*"stargat" + 0.009*"even" + 0.009*"okay" + 0.009*"time"'),
 (8,
  '0.047*"look" + 0.039*"well" + 0.031*"back" + 0.026*"daniel" + 0.025*"come" + 0.019*"time" + 0.019*"jack" + 0.014*"sam" + 0.011*"see" + 0.009*"gate"')]
```

In [17]:
```
#df.to_pickle("c_nmpi_coherence_approach3_nmf.pkl")
```

In [18]:
```
with open("c_nmpi_coherence_approach3_nmf.pkl", "rb") as f:
    coherence_scores_topic_counts = load(f)
```

In [19]:
```
coherence_scores_topic_counts.head(2)
```

Out[19]:

topic_count	coherence	model
2	-0.003822	<my_nmf_utils.my_nmf_model object at 0x000001F...
3	0.005128	<my_nmf_utils.my_nmf_model object at 0x000001F...

In [20]:
```
df = coherence_scores_topic_counts
```

17, 9, 4 maxima

pick topic count 8 for better comparisoin