

# NMF Tuning Approach 3

Document = part between interludes

## Imports

```
In [1]: 1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from wordcloud import WordCloud, STOPWORDS
5 from gensim.utils import simple_preprocess, SaveLoad
6 from nltk.corpus import stopwords
7 from nltk.stem import WordNetLemmatizer, SnowballStemmer
8 from nltk.stem.porter import *
9 import time
10 from pickle import load, dump
11 import my_nmf_utils
12 from my_nmf_utils import my_nmf_model
13 from gensim.models import CoherenceModel
```

## Initializations

```
In [2]: 1 np.random.seed(2018)
2 stemmer = SnowballStemmer('english')
```

```
In [3]: 1 stop_words = stopwords.words('english')
2 stop_words.extend(STOPWORDS)
3 stop_words2 = stop_words
```

## Helpers

```
In [4]: 1 # Cleaning functions
2 def remove_stopwords(texts):
3     return [word for word in texts if word not in stop_words ]
4
5 def stem_and_lemmatize(texts):
6     return [stemmer.stem(WordNetLemmatizer().lemmatize(word, pos='v')) for word in texts]
```

## Import Data

```
In [5]: 1 clean_data_folder = "../clean_data/"
2 with open("../clean_data/"+characters_with_gender.pkl", "rb") as f2:
3     characters = load(f2)
```

```
In [6]: 1 clean_data_folder_lda = "../clean_data_lda"
2 with open(clean_data_folder_lda+"/all_data.pkl", "rb") as f:
3     all_scripts = load(f)
4 data_nmf = pd.merge(all_scripts, characters, on="character", how="inner")
```

```
In [7]: 1 all_scripts.reset_index(inplace=True, drop=True)
```

```
In [8]: 1 data = pd.merge(all_scripts, characters, on="character", how="inner")
```

```
In [9]: 1 data2 = data_nmf[['text', 'doc_id']]
2 data3 = data2.groupby(['doc_id'])['text'].agg(sum).reset_index()
3 documents = data3.text.apply(stem_and_lemmatize)
```

## Tuning

### Number of Topics

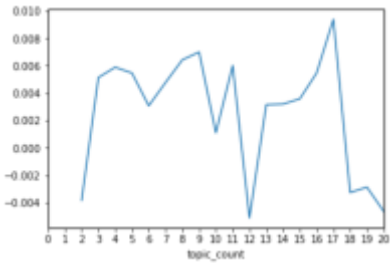
```
In [10]: 1 coherence_scores_nmpi = []
```

```
In [11]: 1 for i in range(2,21,1):
2     nmf_model_x = my_nmf_model(documents)
3     nmf_model_x.train(num_topics = i, passes = 5, tfidf_threshold = 0.0)
4     coherence_model_nmf = CoherenceModel(model=nmf_model_x.nmf_model,
5                                         texts=nmf_model_x.documents,
6                                         dictionary=nmf_model_x.dictionary,
7                                         coherence='c_npmi')
8     coherence_nmf = coherence_model_nmf.get_coherence()
9     coherence_scores_nmpi.append([i, coherence_nmf, nmf_model_x])
10    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]: 1 df = pd.DataFrame(coherence_scores_nmpi)
2 df.columns = ["topic_count", "coherence", "model"]
3 df = df.set_index("topic_count")
4 df.coherence.plot(kind='line',
5                  xticks=[x for x in range(21)],
6                  xlim=[0,20])
7 df.coherence.idxmax()
```

Out[13]: 17



```
In [14]: 1 df.coherence[:6].idxmax()
```

Out[14]: 4

```
In [15]: 1 test_model = df.model[9]
```

```
In [16]: 1 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]: 1 #df.to_pickle("c_nmpi_coherence_approach3_nmf.pkl")
```

```
In [18]: 1 with open("c_nmpi_coherence_approach3_nmf.pkl", "rb") as f:
2 coherence_scores_topic_counts = load(f)
```

```
In [19]: 1 coherence_scores_topic_counts.head(2)
```

Out[19]:

	coherence	model
topic_count		
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]: 1 df = coherence_scores_topic_counts
```

17, 9, 4 maxima

pick topic count 8 for better comparisoin