Topic Modelling

Topic Modelling helps us identify the hidden topics among the customer's review. Knowing these topics enable DisneyLand to identify areas experience at DisneyLand.

Import libraries and download the packages

In [1]:

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import string
import gensim
from gensim import corpora
```

Read the dataset

In [2]:

```
df = pd.read_csv("topic_modelling_reviews.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	reviews
0	['everyone', 'said', 'days', 'christmas', 'bus...
1	['went', 'great', 'much', 'better', 'kids', 'g...
2	['good', 'amusement', 'crowded', 'looking', 'f...
3	['love', 'love', 'love', 'place', 'came', 'sin...
4	['somewhere', 'finances', 'afford', 'freedom',...

In [4]:

```
# Converting the string of list of tokens to normal lists.
import ast
df.reviews = df.reviews.apply(ast.literal_eval)
```

In [5]:

```
print("Number of texts in the dataframe:",df.shape[0])
```

```
Number of texts in the dataframe: 10902
```

Data Preparation

In [8]:

```
# Lemmatize the words and keep lemmatized words with lengths above 2.
lemma = WordNetLemmatizer()
for index in range(df.shape[0]):
    normalized = []
    for words in df.reviews[index]:
        normalized_word = lemma.lemmatize(words)
        if len(normalized_word)>2:
            normalized.append(normalized_word)
    df.reviews[index] = normalized
```

In [9]:

```
# Tokenize the words to generate frequency of words.
from nltk.tokenize import word_tokenize

# Get the tokenized words
tokenized_word_list = [word for word_list in df.reviews for word in word_list]

# Generate word frequencies
all_words_frequency = nltk.FreqDist(tokenized_word_list)
```

In [10]:

```
print("Total number of terms in the document:",len(all_words_frequency))
```

```
Total number of terms in the document: 14712
```

In [11]:

```
print("Top 100 common words:",all_words_frequency.most_common(100))
```

```
Top 100 common words: [('ride', 12012), ('time', 9191), ('day', 7557), ('line', 6051), ('place', 5175), ('one',
e', 3415), ('year', 3355), ('kid', 3198), ('experience', 2995), ('food', 2989), ('hour', 2971), ('great', 2969),
2599), ('long', 2560), ('much', 2462), ('make', 2397), ('really', 2371), ('fun', 2364), ('went', 2325), ('many',
ney', 2106), ('going', 2045), ('way', 2031), ('love', 2025), ('good', 1998), ('also', 1974), ('got', 1974), ('lo
('still', 1784), ('every', 1740), ('always', 1736), ('show', 1733), ('could', 1716), ('thing', 1688), ('never',
ell', 1579), ('california', 1557), ('park', 1552), ('crowd', 1525), ('worth', 1512), ('star', 1504), ('pass', 14
e', 1440), ('staff', 1414), ('member', 1410), ('crowded', 1409), ('earth', 1407), ('everything', 1400), ('charac
42), ('old', 1329), ('need', 1324), ('say', 1303), ('know', 1287), ('amazing', 1281), ('best', 1265), ('happiest
242), ('little', 1240), ('magic', 1224), ('waiting', 1223), ('attraction', 1204), ('cast', 1198), ('magical', 11
0), ('since', 1141), ('sure', 1127), ('think', 1124), ('last', 1110), ('employee', 1091), ('made', 1090), ('enjo
```

These words are common and may make it difficult for the model to differentiate the terms to create unqiue topics. Therefore, I will remove

In [12]:

```
# Extract the top 100 common words for removal
common_words = [word for (word,value) in all_words_frequency.most_common(100)]
```

In [13]:

```
print(common_words)
```

```
['ride', 'time', 'day', 'line', 'place', 'one', 'people', 'pas', 'wait', 'like', 'year', 'kid', 'experience', 'f
ong', 'much', 'make', 'really', 'fun', 'went', 'many', 'see', 'visit', 'first', 'money', 'going', 'way', 'love',
ill', 'every', 'always', 'show', 'could', 'thing', 'never', 'around', 'come', 'price', 'well', 'california', 'pa
d', 'parade', 'staff', 'member', 'crowded', 'earth', 'everything', 'character', 'better', 'two', 'mountain', 'ol
venture', 'new', 'expensive', 'little', 'magic', 'waiting', 'attraction', 'cast', 'magical', 'parking', 'service
'made', 'enjoy', 'able', 'pay', 'said']
```

In [14]:

```
# Iterate through the dataframe and remove the top common words.
for index in range(df.shape[0]):
    uncommon = []
    for words in df.reviews[index]:
        if words not in common_words:
            uncommon.append(words)
    df.reviews[index] = uncommon
```

In [15]:

```
# Creating the term matrix
dictionary = corpora.Dictionary(df.reviews) # unique terms

# Creating term count
doc_term_matrix = [dictionary.doc2bow(doc) for doc in df.reviews]
```

In [18]:

```
# To ensure sufficient but not too much topics, I will test topics in range 4 to 8 with intervals of 1.
topic_num_list = list(np.arange(4,9,1))

topic_num_list
```

Out[18]:

```
[4, 5, 6, 7, 8]
```

Modelling

In [19]:

```
from pprint import pprint
from gensim.models import CoherenceModel

no_topic_tested = []
perplexity_scores = []
coherence_score = []
Lda = gensim.models.LdaMulticore
for number_of_topics in topic_num_list:
    print("Testing with",number_of_topics,"topics.")
#     print("Running the LDA Model")
    ldamodel = Lda(doc_term_matrix, num_topics = number_of_topics, id2word = dictionary, passes=10, random_s
    # The default number of passes is 1.
    # A higher pass also means the model learns more from the data.
    # However, the model will become computationally expensive and time consuming.
    # To reduce the computational time while ensuring sufficient iterations are made, I will use only 10 pas

    # random_state is passed to ensure reproducable results for training and tuning.

    perplexity_score = ldamodel.log_perplexity(doc_term_matrix)

    # Coherence c_v is generally the most interpretable, therefore I set the coherence to c_v.
    coherence_model_lda = CoherenceModel(model=ldamodel, texts=df.reviews, dictionary=dictionary, coherence=
    no_topic_tested.append(number_of_topics)
    perplexity_scores.append(float(perplexity_score))
    coherence_score.append(coherence_model_lda.get_coherence())
    print()

model_result = pd.DataFrame({'Number of Topics':no_topic_tested, 'Perplexity':perplexity_scores, 'Coherence
model_result.sort_values(by='Coherence', ascending=False)
```

```
Testing with 4 topics.

Testing with 5 topics.

Testing with 6 topics.

Testing with 7 topics.

Testing with 8 topics.
```

Out[19]:

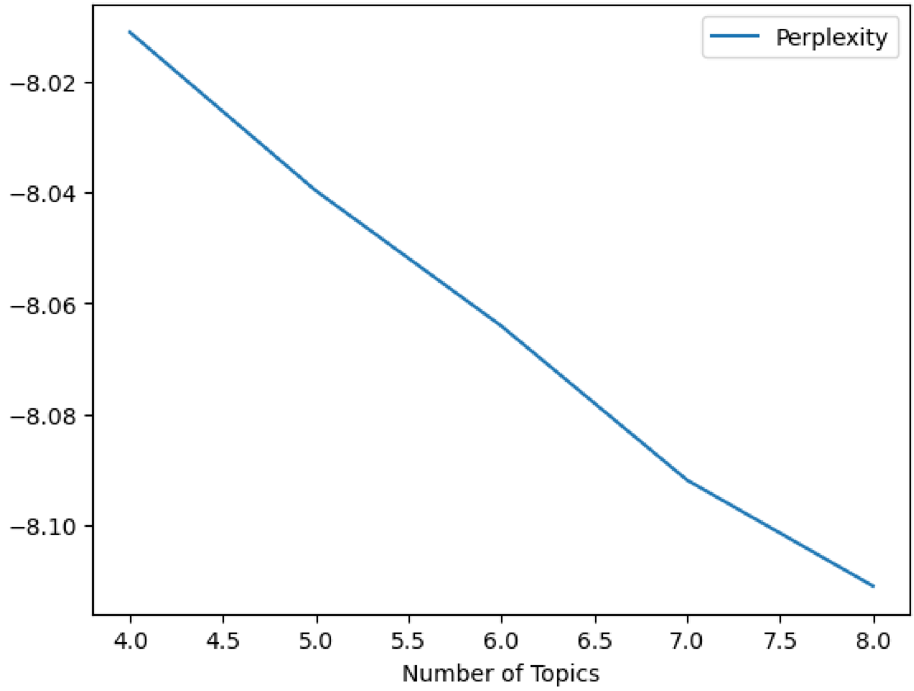
	Number of Topics	Perplexity	Coherence
1	5	-8.039696	0.397245
4	8	-8.111040	0.376143
2	6	-8.064139	0.375133
0	4	-8.011132	0.370975
3	7	-8.091944	0.369993

In [20]:

```
model_result.plot.line(y='Perplexity', x='Number of Topics')
```

Out[20]:

<AxesSubplot:xlabel='Number of Topics'>



- Perplexity is calculated by taking the log likelihood of unseen text documents given the topics defined by a topic model.
- A good model will have a high likelihood and resultantly low perplexity.
- But sometimes these metrics are not correlated with human interpretability of the model, which can be impractical in a business setting

From 4 to 8 topics, the perplexity score decreased increasingly.

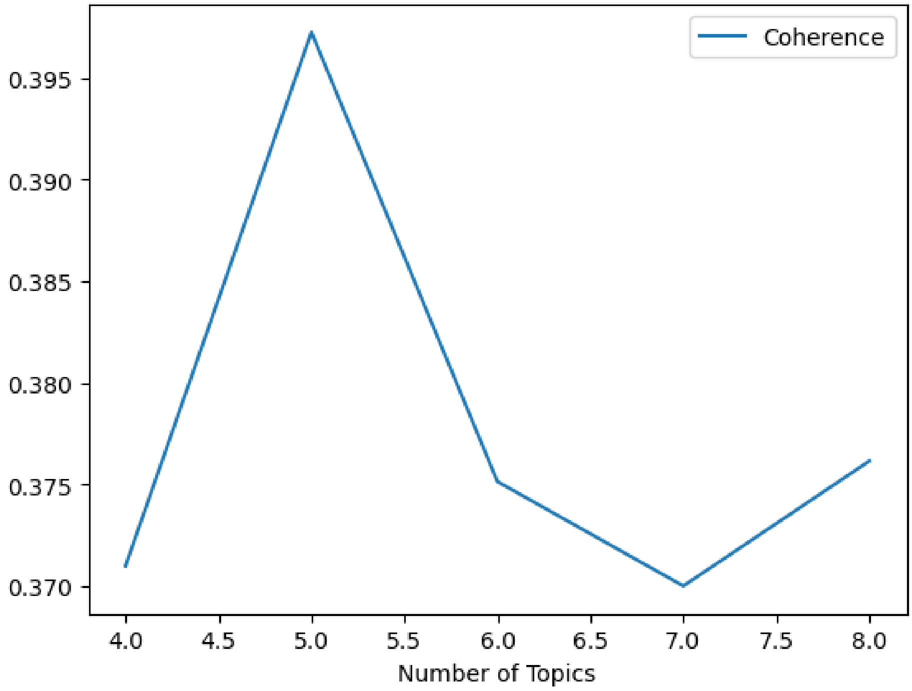
Although a lower perplexity score is better, a lower perplexity score is achieved as there will be less unique topics in each document. Since score instead.

In [21]:

```
# Look for the highest coherence value
model_result.plot.line(y='Coherence', x='Number of Topics')
```

Out[21]:

```
<AxesSubplot:xlabel='Number of Topics'>
```



Topic coherence looks at a set of words in generated topics and rates the interpretability of the topics. The higher the value, the better the m

Since 5 topics has the highest coherence score of 0.3972, I will check out its intertopic distance

In [22]:

```
ldamodel = Lda(doc_term_matrix, num_topics = int(model_result['Number of Topics'][model_result['Coherence']
```

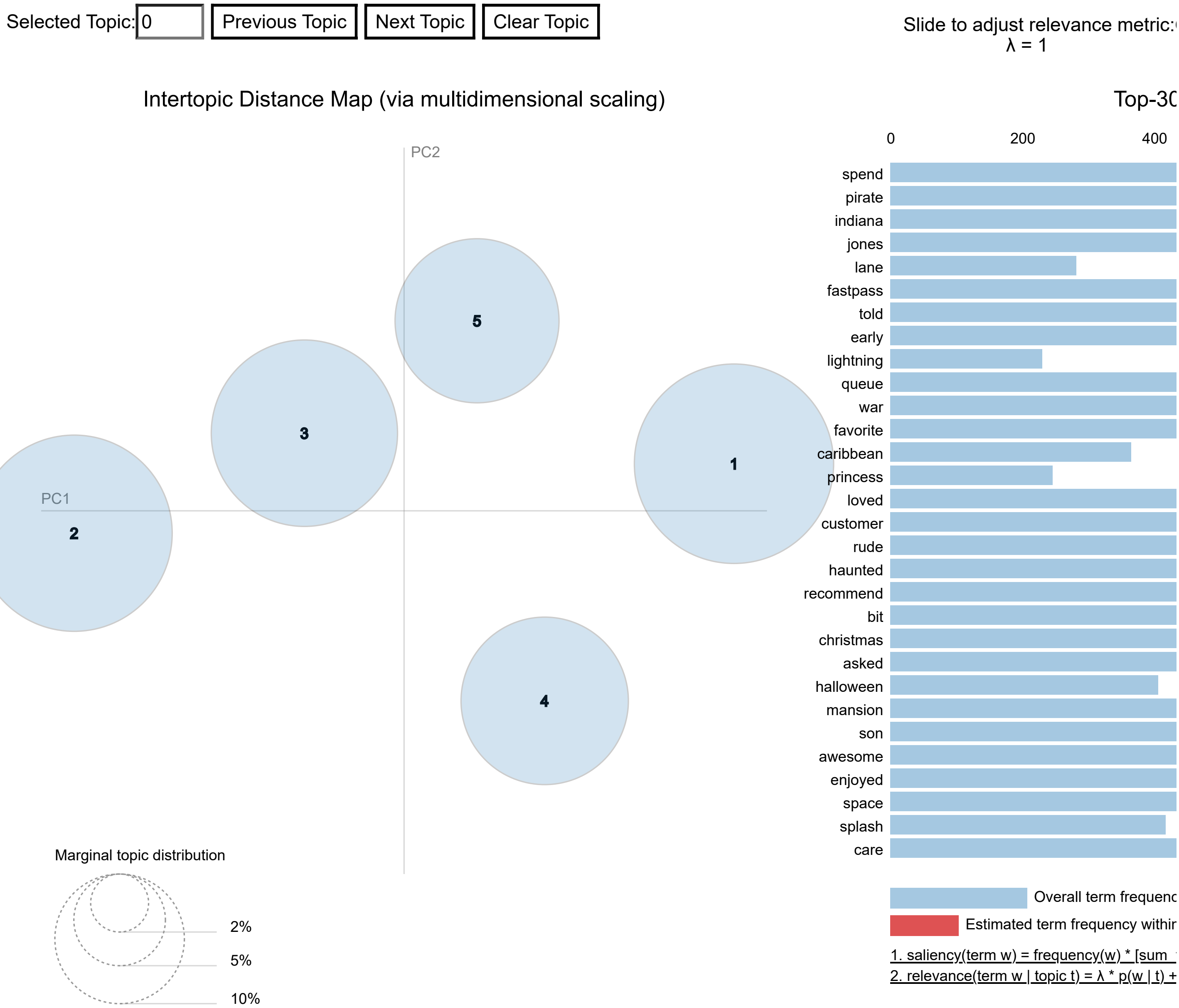
In [23]:

```
# Enter codes here
import pyLDAvis
import pyLDAvis.gensim_models
import matplotlib.pyplot as plt
%matplotlib inline

# visualize the topics and keywords
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(ldamodel, doc_term_matrix, dictionary)
vis
```

```
C:\Users\fangg\anaconda3_new\lib\site-packages\past\builtins\misc.py:45: DeprecationWarning: the imp module is d
ion for alternative uses
  from imp import reload
C:\Users\fangg\anaconda3_new\lib\site-packages\pyLDAvis\_prepare.py:246: FutureWarning: In a future version of p
'labels' will be keyword-only.
  default_term_info = default_term_info.sort_values(
```

Out[23]:



With 100 most common words removed and lemmatized words left in the dataframe, the topics are now much more easy to interpret and th the previous intertopic distance map, where all topics have roughly the same terms.

In [24]:

```
pprint(ldamodel.print_topics(num_topics=5, num_words=20))
```

```
[(0,
  '0.007*"told" + 0.005*"customer" + 0.004*"rude" + 0.004*"another" + '
  '0.004*"ever" + 0.004*"used" + 0.004*"let" + 0.004*"guest" + 0.004*"annual" '
  '+ 0.004*"asked" + 0.003*"paid" + 0.003*"use" + 0.003*"disappointed" + '
  '0.003*"spent" + 0.003*"phone" + 0.003*"help" + 0.003*"nothing" + '
  '0.003*"worst" + 0.003*"person" + 0.003*"daughter"'),
 (1,
  '0.007*"early" + 0.005*"pirate" + 0.005*"jones" + 0.005*"indiana" + '
  '0.005*"space" + 0.005*"fastpass" + 0.005*"favorite" + 0.005*"closed" + '
  '0.005*"app" + 0.004*"water" + 0.004*"small" + 0.004*"haunted" + '
  '0.004*"awesome" + 0.004*"area" + 0.004*"recommend" + 0.004*"bring" + '
  '0.004*"use" + 0.004*"big" + 0.003*"mansion" + 0.003*"plan"'),
 (2,
  '0.006*"spend" + 0.004*"least" + 0.004*"cost" + 0.003*"lane" + 0.003*"let" + '
  '0.003*"ever" + 0.003*"guest" + 0.003*"bad" + 0.003*"something" + '
  '0.003*"getting" + 0.003*"keep" + 0.003*"land" + 0.003*"lightning" + '
  '0.003*"care" + 0.003*"customer" + 0.003*"everyone" + 0.003*"half" + '
  '0.003*"genie" + 0.003*"must" + 0.003*"life"'),
 (3,
  '0.005*"war" + 0.005*"took" + 0.004*"walk" + 0.004*"queue" + '
  '0.004*"daughter" + 0.004*"security" + 0.004*"land" + 0.003*"adult" + '
  '0.003*"min" + 0.003*"rude" + 0.003*"everyone" + 0.003*"closed" + '
  '0.003*"princess" + 0.003*"away" + 0.003*"told" + 0.003*"another" + '
  '0.003*"spent" + 0.003*"disappointed" + 0.003*"bring" + 0.003*"picture"'),
 (4,
  '0.005*"loved" + 0.005*"definitely" + 0.005*"feel" + 0.004*"recommend" + '
  '0.004*"son" + 0.004*"everyone" + 0.004*"theme" + 0.004*"bit" + '
  '0.004*"whole" + 0.004*"christmas" + 0.003*"night" + 0.003*"halloween" + '
  '0.003*"special" + 0.003*"week" + 0.003*"though" + 0.003*"different" + '
  '0.003*"app" + 0.003*"visited" + 0.003*"use" + 0.003*"found"')]
```

Looking at the individual topic and their terms, the topics are not very clear. These are the topics I assume it to be:

- Topic 0: Rude Service
- Topic 1: Themes
- Topic 2: Costs
- Topic 3: War
- Topic 4: Customer Recommendation

From these identified topics, DisneyLand could consider reviewing their staff's attitude and approach to managing their customers. By ident could potentially improve their customer's experience and satisfaction level, leading to higher chances of custome rrecommendations.