In [1]:

```
import pandas as pd
import numpy as np

import re
import string
import os
os.chdir('..')
import spacy
import conda

import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from pprint import pprint


# libraries for visualization
import pyLDAvis
import pyLDAvis.gensim
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\saeid\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
review_data=pd.read_csv("C:/Users/saeid/OneDrive/Documents/claremont/466/HW3/Review1.csv")
print(review_data.head(2))
print(len(review_data))
print('Unique Products')
print(len(review_data.groupby('ProductId')))
print('Unique Users')
print(len(review_data.groupby('UserId')))
```

```
   Id   ProductId          UserId ProfileName  HelpfulnessNumerator  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW  delmartian                     1
1   2  B00813GRG4  A1D87F6ZCVE5NK      dll pa                     0

   HelpfulnessDenominator  Score        Time                Summary  \
0                       1      5  1303862400  Good Quality Dog Food
1                       0      1  1346976000      Not as Advertised

                                                Text
0  I have bought several of the Vitality canned d...
1  Product arrived labeled as Jumbo Salted Peanut...
999
Unique Products
207
Unique Users
964
```

In [3]:
```
def clean_text(text ):
    delete_dict = {sp_character: '' for sp_character in string.punctuation}
    delete_dict[' '] = ' '
    table = str.maketrans(delete_dict)
    text1 = text.translate(table)
    #print('cleaned:'+text1)
    textArr= text1.split()
    text2 = ' '.join([w for w in textArr if ( not w.isdigit() and  ( not w.isdigit() and len(w)>3))])
    return text2.lower()
```

In [4]:
```
review_data.dropna(axis = 0, how ='any',inplace=True)


review_data['Text'] = review_data['Text'].apply(clean_text)
review_data['Num_words_text'] = review_data['Text'].apply(lambda x:len(str(x).split()))

print('-------Dataset --------')
print(review_data['Score'].value_counts())
print(len(review_data))
print('-------------------------')
max_review_data_sentence_length  = review_data['Num_words_text'].max()

mask = (review_data['Num_words_text'] < 100) & (review_data['Num_words_text'] >=20)
df_short_reviews = review_data[mask]
df_sampled = df_short_reviews.groupby('Score').apply(lambda x: x.sample(n=30)).reset_index(drop = True)

print('No of Short reviews')
print(len(df_short_reviews))



#all_sentences = train_data['text'].tolist() + test_data['text'].tolist()
```

```
-------Dataset --------
5    642
4    138
1     98
3     75
2     46
Name: Score, dtype: int64
999
-------------------------
No of Short reviews
647
```

In [5]:
```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
# function to remove stopwords
def remove_stopwords(text):
    textArr = text.split(' ')
    rem_text = " ".join([i for i in textArr if i not in stop_words])
    return rem_text

# remove stopwords from the text
df_sampled['Text']=df_sampled['Text'].apply(remove_stopwords)
```

In [6]:
```
nlp = spacy.load('en_core_web_md', disable=['parser', 'ner'])

def lemmatization(texts,allowed_postags=['NOUN', 'ADJ']):
       output = []
       for sent in texts:
             doc = nlp(sent)
             output.append([token.lemma_ for token in doc if token.pos_ in allowed_postags ])
       return output
```

In [7]:
```
from nltk.stem import WordNetLemmatizer
text_list=df_sampled['Text'].tolist()
print(text_list[1])
tokenized_reviews = lemmatization(text_list)
print(tokenized_reviews[1])
```

```
like order kettle spicy thai chips amazon hard find locally probably aggrieved know recipe changed like regul
e himbr yuck
['order', 'kettle', 'spicy', 'thai', 'chip', 'recipe', 'regular', 'potatoe', 'chip', 'husband', 'spicy', 'tha
```

In [8]:
```
dictionary = corpora.Dictionary(tokenized_reviews)
doc_term_matrix = [dictionary.doc2bow(rev) for rev in tokenized_reviews]
```

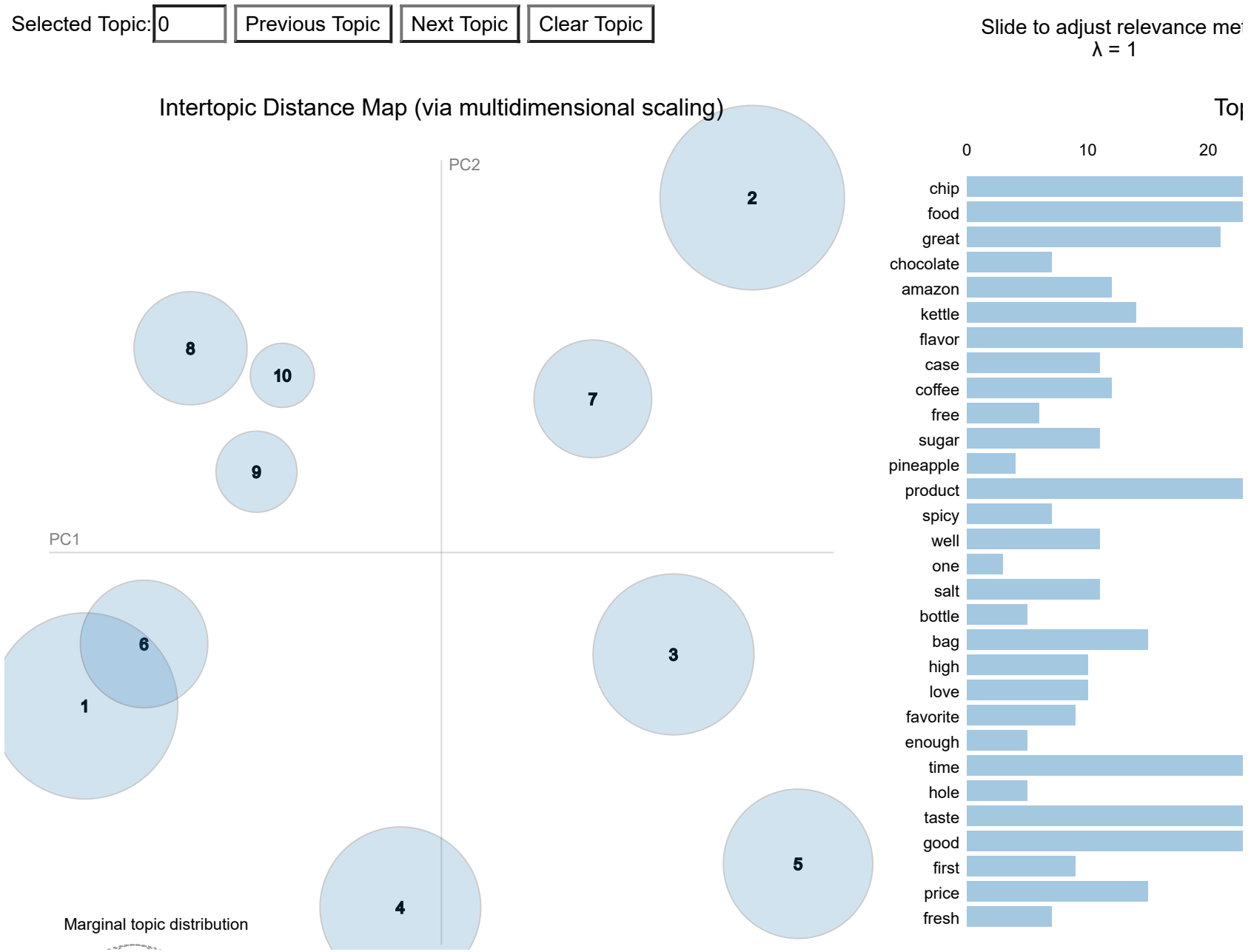
In [9]:
```
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

# Build LDA model
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=10, random_state=100,
                chunksize=1000, passes=50,iterations=100)
```

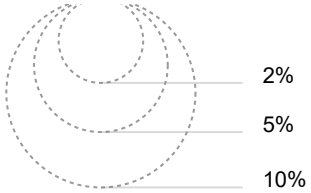
In [10]:
```
lda_model.print_topics()
```

Out[10]:
```
[(0,
  '0.078*"chip" + 0.019*"kettle" + 0.015*"good" + 0.014*"flavor" + 0.014*"salt" + 0.013*"bag" + 0.011*"hole"
 (1,
  '0.018*"chocolate" + 0.009*"first" + 0.009*"coffee" + 0.009*"time" + 0.009*"enough" + 0.009*"back" + 0.009*
a"'),
 (2,
  '0.019*"taste" + 0.019*"product" + 0.019*"good" + 0.017*"chip" + 0.013*"coffee" + 0.013*"well" + 0.013*"sug
 (3,
  '0.027*"great" + 0.020*"good" + 0.014*"flavor" + 0.013*"chocolate" + 0.013*"product" + 0.011*"price" + 0.01
 (4,
  '0.024*"flavor" + 0.024*"food" + 0.015*"case" + 0.015*"pineapple" + 0.012*"love" + 0.012*"time" + 0.012*"sp
 (5,
  '0.035*"chip" + 0.022*"flavor" + 0.011*"potato" + 0.011*"time" + 0.011*"taste" + 0.009*"plastic" + 0.009*"s
 (6,
  '0.013*"product" + 0.013*"taste" + 0.013*"case" + 0.013*"bottle" + 0.010*"price" + 0.010*"coffee" + 0.010*"
 (7,
  '0.034*"food" + 0.018*"chip" + 0.016*"amazon" + 0.013*"time" + 0.009*"little" + 0.009*"natural" + 0.008*"go
```

In [12]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, doc_term_matrix, dictionary)
vis
```

Out[12]:





Overall term freq

Estimated term frequency v

1. saliency(term w) = frequency(w) * [s
2. relevance(term w | topic t) = λ * p(w