In [1]:
```
import nltk; nltk.download('stopwords')
```

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

Out[1]:
```
True
```

In [2]:
```
import re
import numpy as np
import pandas as pd
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

In [3]:
```
# NLTK Stop words
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

In [4]:

```
# Import Dataset
df = pd.read_json('https://raw.githubusercontent.com/selva86/datasets/master/r
print(df.target_names.unique())
df.head()
```

```
['rec.autos' 'comp.sys.mac.hardware' 'rec.motorcycles' 'misc.forsale'
 'comp.os.ms-windows.misc' 'alt.atheism' 'comp.graphics'
 'rec.sport.baseball' 'rec.sport.hockey' 'sci.electronics' 'sci.space'
 'talk.politics.misc' 'sci.med' 'talk.politics.mideast'
 'soc.religion.christian' 'comp.windows.x' 'comp.sys.ibm.pc.hardware'
 'talk.politics.guns' 'talk.religion.misc' 'sci.crypt']
```

Out[4]:

	content	target	target_names
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	7	rec.autos
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	4	comp.sys.mac.hardware
10	From: irwin@cmptrc.lonestar.org (Irwin Arnstei...	8	rec.motorcycles
100	From: tchen@magnus.acs.ohio-state.edu (Tsung-K...	6	misc.forsale
1000	From: dabl2@nlm.nih.gov (Don A.B. Lindbergh)\n...	2	comp.os.ms-windows.misc

In [5]:

```
# Convert to list
data = df.content.values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]

pprint(data[:1])
```

```
['From: (wheres my thing) Subject: WHAT car is this!? Nntp-Posting-Host: '
 'rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: '
 '15 I was wondering if anyone out there could enlighten me on this car I saw '
 'the other day. It was a 2-door sports car, looked to be from the late 60s/ '
 'early 70s. It was called a Bricklin. The doors were really small. In '
 'addition, the front bumper was separate from the rest of the body. This is '
 'all I know. If anyone can tellme a model name, engine specs, years of '
 'production, where this car is made, history, or whatever info you have on '
 'this funky looking car, please e-mail. Thanks, - IL ---- brought to you by '
 'your neighborhood Lerxst ---- ']
```

In [6]:
```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # de

data_words = list(sent_to_words(data))

print(data_words[:1])
```

```
[['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nr
ryland', 'college', 'park', 'lines', 'was', 'wondering', 'if', 'anyone', 'out'
'it', 'was', 'door', 'sports', 'car', 'looked', 'to', 'be', 'from', 'the', 'la
'in', 'addition', 'the', 'front', 'bumper', 'was', 'separate', 'from', 'the',
'model', 'name', 'engine', 'specs', 'years', 'of', 'production', 'where', 'thi
'funky', 'looking', 'car', 'please', 'mail', 'thanks', 'il', 'brought', 'to',
```

In [7]:
```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # highe
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nnt
ark', 'lines', 'was', 'wondering', 'if', 'anyone', 'out', 'there', 'could', 'e
ports', 'car', 'looked', 'to', 'be', 'from', 'the', 'late', 'early', 'it', 'wa
e', 'front_bumper', 'was', 'separate', 'from', 'the', 'rest', 'of', 'the', 'bc
e', 'specs', 'years', 'of', 'production', 'where', 'this', 'car', 'is', 'made'
r', 'please', 'mail', 'thanks', 'il', 'brought', 'to', 'you', 'by', 'your', 'r
```

In [8]:
```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allov
    return texts_out
```

In [9]:
```
# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN',

print(data_lemmatized[:1])
```

```
[['where', 's', 'thing', 'car', 'nntp_poste', 'host', 'umd', 'organization', '
'car', 'see', 'day', 'door', 'sport', 'car', 'look', 'late', 'early', 'call',
y', 'know', 'anyone', 'tellme', 'model', 'name', 'engine', 'specs', 'year', 'p
ng', 'neighborhood', 'lerxst']]
```

In [10]:
```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1), (1, 2), (2, 1), (3, 1), (4, 1), (5, 1), (6, 5), (7, 1), (8, 1), (9,
1), (20, 1), (21, 1), (22, 2), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (2
8, 1), (39, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1), (45, 1), (46, 1),
```

In [11]:
```
id2word[0]
```

Out[11]:
```
'addition'
```

In [12]:
```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

Out[12]:
```
[[('addition', 1),
  ('anyone', 2),
  ('body', 1),
  ('bricklin', 1),
  ('bring', 1),
  ('call', 1),
  ('car', 5),
  ('could', 1),
  ('day', 1),
  ('door', 2),
  ('early', 1),
  ('engine', 1),
  ('enlighten', 1),
  ('front_bumper', 1),
  ('funky', 1),
  ('history', 1),
  ('host', 1),
  ('info', 1),
  ('know', 1),
  ('late', 1),
  ('lerxst', 1),
  ('line', 1),
  ('look', 2),
  ('mail', 1),
  ('make', 1),
  ('maryland_college', 1),
  ('model', 1),
  ('name', 1),
  ('neighborhood', 1),
  ('nntp_poste', 1),
  ('organization', 1),
  ('park', 1),
  ('production', 1),
  ('really', 1),
  ('rest', 1),
  ('s', 1),
  ('see', 1),
  ('separate', 1),
  ('small', 1),
  ('specs', 1),
  ('sport', 1),
  ('tellme', 1),
  ('thank', 1),
  ('thing', 1),
  ('umd', 1),
  ('university', 1),
  ('where', 1),
  ('wonder', 1),
  ('year', 1)]]
```

In [13]:

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [14]:

```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.245*"window" + 0.102*"driver" + 0.063*"peter" + 0.019*"sweden" + '
  '0.018*"microsoft" + 0.017*"window_manager" + 0.016*"dos" + 0.014*"win" + '
  '0.012*"henry_spencer" + 0.011*"gate"'),
 (1,
  '0.039*"kid" + 0.032*"cover" + 0.029*"drug" + 0.024*"food" + 0.022*"rate" +
  '0.019*"genocide" + 0.017*"tax" + 0.016*"health" + 0.016*"private" + '
  '0.016*"germany"'),
 (2,
  '0.027*"use" + 0.019*"system" + 0.017*"also" + 0.015*"problem" + 0.014*"new"
  '+ 0.013*"run" + 0.012*"work" + 0.012*"drive" + 0.011*"need" + 0.011*"card"'
 (3,
  '0.019*"state" + 0.017*"people" + 0.015*"right" + 0.015*"law" + 0.013*"gun"
  '+ 0.012*"kill" + 0.012*"government" + 0.010*"group" + 0.009*"country" + '
  '0.008*"issue"'),
 (4,
  '0.064*"ca" + 0.034*"rise" + 0.033*"prove" + 0.028*"pittsburgh" + '
  '0.026*"mike" + 0.026*"door" + 0.025*"univ" + 0.025*"engine" + 0.020*"city"
  '+ 0.019*"pin"'),
 (5,
  '0.093*"display" + 0.051*"brother" + 0.039*"air" + 0.038*"picture" + '
  '0.024*"pointer" + 0.017*"greatly_appreciate" + 0.015*"xlib" + 0.013*"sc" +
  '0.012*"eg" + 0.010*"cursor"'),
 (6,
  '0.051*"game" + 0.048*"team" + 0.036*"year" + 0.032*"play" + 0.030*"win" + '
  '0.028*"player" + 0.023*"hockey" + 0.019*"season" + 0.017*"hit" + '
  '0.015*"canada"'),
 (7,
  '0.109*"line" + 0.097*"organization" + 0.053*"write" + 0.051*"article" + '
  '0.047*"university" + 0.038*"host" + 0.024*"reply" + 0.023*"nntp_poste" + '
  '0.022*"thank" + 0.021*"anyone"'),
 (8,
  '0.079*"car" + 0.058*"sale" + 0.056*"chip" + 0.055*"price" + 0.042*"sell" +
  '0.036*"buy" + 0.027*"package" + 0.027*"model" + 0.022*"oil" + '
  '0.020*"extra"'),
 (9,
  '0.052*"not" + 0.032*"do" + 0.026*"would" + 0.025*"be" + 0.019*"know" + '
  '0.019*"go" + 0.018*"get" + 0.015*"think" + 0.015*"say" + 0.014*"good"'),
 (10,
  '0.047*"key" + 0.037*"president" + 0.031*"israel" + 0.023*"israeli" + '
  '0.021*"report" + 0.018*"government" + 0.017*"encryption" + 0.016*"arab" + '
  '0.014*"center" + 0.014*"security"'),
 (11,
  '0.017*"say" + 0.016*"god" + 0.013*"christian" + 0.013*"believe" + '
  '0.012*"people" + 0.011*"life" + 0.010*"claim" + 0.010*"man" + 0.009*"word"
  '+ 0.008*"fact"'),
 (12,
  '0.086*"image" + 0.074*"color" + 0.067*"bit" + 0.051*"mode" + 0.031*"datum"
  '+ 0.023*"output" + 0.023*"input" + 0.022*"interface" + 0.022*"sample" + '
  '0.021*"serial"'),
 (13,
  '0.829*"ax" + 0.058*"max" + 0.006*"font" + 0.004*"mhz" + 0.004*"bbs" + '
```

```
 '0.003*"pro" + 0.003*"usenet" + 0.003*"electrical" + 0.003*"hook" + '
 '0.003*"shipping"'),
(14,
 '0.126*"space" + 0.031*"matthew" + 0.027*"mark" + 0.026*"earth" + '
 '0.021*"satellite" + 0.020*"command" + 0.019*"orbit" + 0.018*"flight" + '
 '0.018*"moon" + 0.017*"somebody"'),
(15,
 '0.065*"king" + 0.047*"staff" + 0.033*"waco" + 0.023*"kent_sandvik" + '
 '0.021*"bureau" + 0.012*"cookamunga_tourist" + 0.008*"tammy" + '
 '0.005*"branch_davidian" + 0.000*"pts_pt" + 0.000*"sj"'),
(16,
 '0.052*"front" + 0.035*"field" + 0.029*"newsletter" + 0.028*"exact" + '
 '0.024*"hospital" + 0.024*"oppose" + 0.024*"docs" + 0.023*"logo" + '
 '0.020*"patch" + 0.016*"salary"'),
(17,
 '0.101*"contact" + 0.059*"mb" + 0.056*"bus" + 0.051*"scsi" + '
 '0.034*"excellent" + 0.029*"shell" + 0.021*"component" + 0.020*"entry" + '
 '0.019*"configuration" + 0.014*"vesa_local"'),
(18,
 '0.086*"bike" + 0.059*"circuit" + 0.048*"ride" + 0.040*"print" + 0.039*"dod"
 '+ 0.037*"unix" + 0.030*"load" + 0.029*"dog" + 0.029*"eric" + 0.015*"gene"')
(19,
 '0.127*"file" + 0.043*"obvious" + 0.031*"site" + 0.026*"directory" + '
 '0.025*"commercial" + 0.023*"available" + 0.023*"motif" + 0.019*"archive" +
 '0.016*"convert" + 0.016*"computing"')]
```

In [15]:

```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, c
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -14.67218499612371

Coherence Score:  0.46495485049524843
```

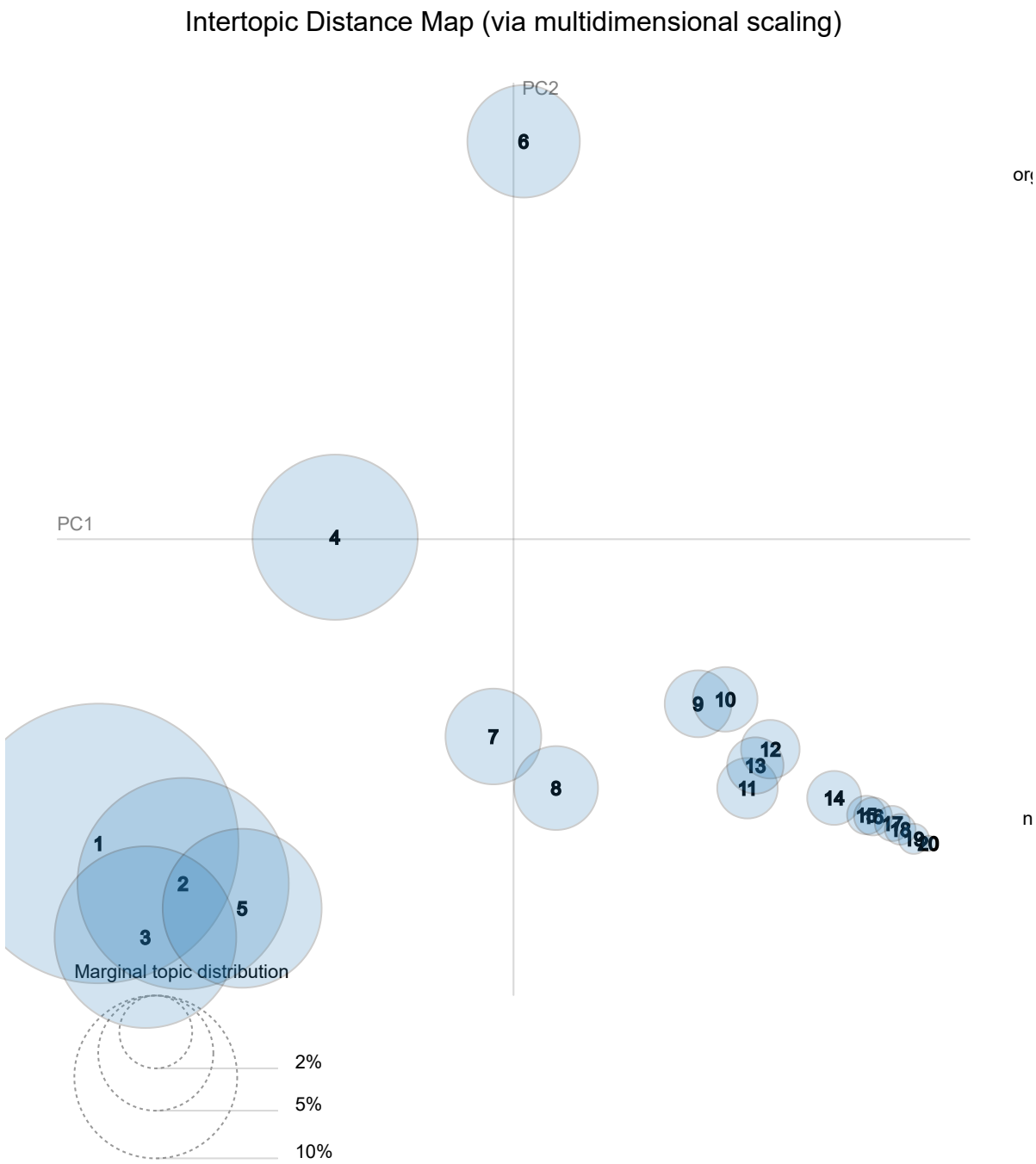
In [16]:

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

```
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages\pyLDAv
future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  return pd.concat([default_term_info] + list(topic_dfs))
```

Out[16]:

Selected Topic: 0 Previous Topic Next Topic Clear Topic



In [19]:

```
import os
from gensim.models.wrappers import LdaMallet

os.environ['MALLET_HOME'] = 'C:\\Users\\finan\\OneDrive\\Documents\\mallet-2.0

# Download File: http://mallet.cs.umass.edu/dist/mallet-2.0.8.zip
mallet_path = "C:\\Users\\finan\\OneDrive\\Documents\\mallet-2.0.8\\bin\\malle
ldamallet = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_t
```

In [20]:

```
# Show Topics
pprint(ldamallet.show_topics(formatted=False))

# Compute Coherence Score
coherence_model_ldamallet = CoherenceModel(model=ldamallet, texts=data_lemmati
coherence_ldamallet = coherence_model_ldamallet.get_coherence()
print('\nCoherence Score: ', coherence_ldamallet)
```

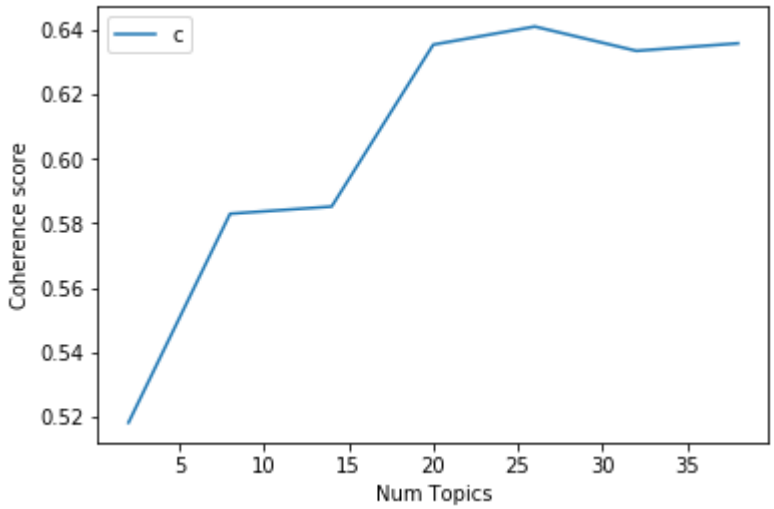
```
   ('problem', 0.007787039895715415),
   ('article', 0.007615519193166615),
   ('food', 0.00655209083736407),
   ('time', 0.00579739974614936),
   ('doctor', 0.00579739974614936),
   ('effect', 0.0055401186923261635),
   ('disease', 0.0055401186923261635),
   ('medical', 0.005505814551816405)]),
 (7,
  [('key', 0.031501737640332476),
   ('system', 0.013949168014605788),
   ('encryption', 0.01137073397287039),
   ('bit', 0.01002546403805192),
   ('security', 0.009400874425457633),

   ('government', 0.008776284812863343),
   ('chip', 0.008648163866690156),
   ('technology', 0.008584103393603562),
   ('public', 0.007735302125206195),
```

In [21]:

```
def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3
    """
    Compute c_v coherence for various number of topics

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    texts : List of input texts
    limit : Max num of topics

    Returns:
    -------
    model_list : List of LDA topic models
    coherence_values : Coherence values corresponding to the LDA model with re
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, r
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=d
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

In [22]:
```
# Can take a long time to run.
model_list, coherence_values = compute_coherence_values(dictionary=id2word, co
```

In [23]:
```
# Show graph
limit=40; start=2; step=6;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [24]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.5181
Num Topics = 8  has Coherence Value of 0.583
Num Topics = 14  has Coherence Value of 0.5852
Num Topics = 20  has Coherence Value of 0.6355
Num Topics = 26  has Coherence Value of 0.6411
Num Topics = 32  has Coherence Value of 0.6336
Num Topics = 38  has Coherence Value of 0.6359
```

In [25]:

```
# Select the model and print the topics
optimal_model = model_list[3]
model_topics = optimal_model.show_topics(formatted=False)
pprint(optimal_model.print_topics(num_words=10))
```

```
[(0,
  '0.102*"line" + 0.098*"organization" + 0.076*"write" + 0.071*"university" +
  '0.069*"host" + 0.066*"article" + 0.040*"nntp_poste" + 0.039*"nntp_posting"
  '+ 0.029*"reply" + 0.012*"distribution_world"'),
 (1,
  '0.023*"mail" + 0.018*"information" + 0.018*"post" + 0.016*"send" + '
  '0.014*"list" + 0.013*"internet" + 0.012*"address" + 0.012*"group" + '
  '0.012*"email" + 0.011*"include"'),
 (2,
  '0.022*"make" + 0.021*"thing" + 0.020*"time" + 0.019*"write" + 0.017*"good"
  '+ 0.017*"people" + 0.012*"start" + 0.012*"hear" + 0.012*"bad" + '
  '0.012*"happen"'),
 (3,
  '0.009*"drug" + 0.009*"article" + 0.007*"problem" + 0.007*"organization" + '
  '0.007*"food" + 0.006*"study" + 0.006*"doctor" + 0.006*"disease" + '
  '0.006*"medical" + 0.005*"effect"'),
 (4,
  '0.020*"gun" + 0.017*"state" + 0.015*"law" + 0.013*"people" + 0.008*"case" +
  '0.008*"government" + 0.007*"crime" + 0.007*"weapon" + 0.006*"firearm" + '
  '0.006*"article"'),
 (5,
  '0.025*"line" + 0.015*"sale" + 0.012*"organization" + 0.010*"work" + '
  '0.009*"power" + 0.008*"good" + 0.008*"box" + 0.007*"sell" + 0.007*"ground"
  '+ 0.007*"wire"'),
 (6,
  '0.032*"key" + 0.015*"system" + 0.012*"encryption" + 0.010*"bit" + '
  '0.009*"technology" + 0.009*"chip" + 0.009*"security" + 0.009*"government" +
  '0.008*"message" + 0.008*"public"'),
 (7,
  '0.029*"god" + 0.017*"christian" + 0.014*"people" + 0.009*"bible" + '
  '0.008*"religion" + 0.008*"man" + 0.007*"life" + 0.007*"church" + '
  '0.007*"faith" + 0.006*"atheist"'),
 (8,
  '0.862*"ax" + 0.059*"max" + 0.002*"qax" + 0.002*"qq" + 0.001*"mb" + '
  '0.001*"giz" + 0.001*"mf" + 0.001*"bs" + 0.001*"tm" + 0.001*"sl"'),
 (9,
  '0.021*"window" + 0.012*"image" + 0.011*"file" + 0.011*"program" + '
  '0.010*"version" + 0.010*"display" + 0.009*"run" + 0.009*"server" + '
  '0.009*"application" + 0.008*"graphic"'),
 (10,
  '0.024*"space" + 0.007*"launch" + 0.007*"system" + 0.007*"earth" + '
  '0.006*"nasa" + 0.006*"satellite" + 0.006*"organization" + 0.006*"center" +
  '0.006*"datum" + 0.006*"project"'),
 (11,
  '0.015*"write" + 0.014*"israel" + 0.011*"article" + 0.011*"israeli" + '
  '0.010*"jew" + 0.010*"people" + 0.010*"state" + 0.009*"arab" + 0.008*"war" +
  '0.008*"world"'),
 (12,
  '0.025*"game" + 0.018*"team" + 0.015*"year" + 0.015*"play" + 0.012*"player"
  '+ 0.012*"win" + 0.008*"good" + 0.008*"season" + 0.007*"hockey" + '
  '0.006*"fan"'),
```

```
(13,
 '0.015*"make" + 0.015*"year" + 0.014*"work" + 0.012*"money" + 0.010*"pay" +
 '0.010*"president" + 0.010*"people" + 0.008*"time" + 0.008*"job" + '
 '0.007*"give"'),
(14,
 '0.017*"point" + 0.017*"question" + 0.012*"make" + 0.012*"reason" + '
 '0.011*"write" + 0.011*"claim" + 0.011*"exist" + 0.010*"thing" + '
 '0.010*"post" + 0.010*"argument"'),
(15,
 '0.025*"drive" + 0.019*"card" + 0.017*"problem" + 0.015*"system" + '
 '0.013*"scsi" + 0.013*"driver" + 0.012*"window" + 0.011*"mac" + 0.010*"bit"
 '+ 0.010*"disk"'),
(16,
 '0.019*"armenian" + 0.014*"people" + 0.007*"turkish" + 0.007*"kill" + '
 '0.007*"woman" + 0.006*"greek" + 0.006*"time" + 0.006*"turk" + 0.006*"man" +
 '0.006*"leave"'),
(17,
 '0.028*"car" + 0.013*"article" + 0.011*"bike" + 0.010*"good" + '
 '0.010*"organization" + 0.010*"line" + 0.009*"write" + 0.009*"buy" + '
 '0.007*"ride" + 0.007*"drive"'),
(18,
 '0.033*"file" + 0.025*"line" + 0.015*"change" + 0.014*"program" + '
 '0.014*"read" + 0.014*"write" + 0.012*"set" + 0.011*"time" + 0.011*"number"
 '+ 0.011*"follow"'),
(19,
 '0.048*"organization" + 0.040*"line" + 0.022*"_" + 0.017*"ca" + 0.007*"ed" +
 '0.006*"air" + 0.006*"md" + 0.006*"reply" + 0.005*"te" + 0.004*"mv"')]
```

In [26]:

```
def format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each docu
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_nu
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Ke

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=optimal_model, corp

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Cont

# Show
df_dominant_topic.head(10)
```

Out[26]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords
0	0	17.0	0.1810	car, article, bike, good, organization, line, ..
1	1	15.0	0.2031	drive, card, problem, system, scsi, driver, wi..
2	2	17.0	0.3414	car, article, bike, good, organization, line, ..
3	3	9.0	0.3737	window, image, file, program, version, display..
4	4	15.0	0.2128	drive, card, problem, system, scsi, driver, wi..
5	5	17.0	0.4739	car, article, bike, good, organization, line, ..
6	6	0.0	0.1820	line, organization, write, university, host, a..
7	7	1.0	0.1486	mail, information, post, send, list, internet,..
8	8	4.0	0.1828	gun, state, law, people, case, government, cri..
9	9	7.0	0.2545	god, christian, people, bible, religion, man, ..

In [27]:

```
# Group top 5 sentences under each topic
sent_topics_sorteddf_mallet = pd.DataFrame()

sent_topics_outdf_grpd = df_topic_sents_keywords.groupby('Dominant_Topic')

for i, grp in sent_topics_outdf_grpd:
    sent_topics_sorteddf_mallet = pd.concat([sent_topics_sorteddf_mallet,
                                             grp.sort_values(['Perc_Contributi
                                            axis=0)

# Reset Index
sent_topics_sorteddf_mallet.reset_index(drop=True, inplace=True)

# Format
sent_topics_sorteddf_mallet.columns = ['Topic_Num', "Topic_Perc_Contrib", "Key

# Show
sent_topics_sorteddf_mallet.head()
```

Out[27]:

	Topic_Num	Topic_Perc_Contrib	Keywords	
0	0.0	0.3271	line, organization, write, university, host, a...	Nntp-Posting-Ho
1	1.0	0.6614	mail, information, post, send, list, internet,...	From: (Robert Lipma
2	2.0	0.6451	make, thing, time, write, good, people, start,...	From: (Scott W I
3	3.0	0.7523	drug, article, problem, organization, food, st...	From: Subject: I
4	4.0	0.7441	gun, state, law, people, case, government, cri...	From: (Larr

In [28]:

```
# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)

# Topic Number and Keywords
topic_num_keywords = df_topic_sents_keywords[['Dominant_Topic', 'Topic_Keywor

# Concatenate Column wise
df_dominant_topics = pd.concat([topic_num_keywords, topic_counts, topic_contri

# Change Column names
df_dominant_topics.columns = ['Dominant_Topic', 'Topic_Keywords', 'Num_Documer

# Show
df_dominant_topics
```

Out[28]:

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Doc
0	17.0	car, article, bike, good, organization, line, ...	803.0	
1	15.0	drive, card, problem, system, scsi, driver, wi...	428.0	
2	17.0	car, article, bike, good, organization, line, ...	368.0	
3	9.0	window, image, file, program, version, display...	513.0	
4	15.0	drive, card, problem, system, scsi, driver, wi...	522.0	
5	17.0	car, article, bike, good, organization, line, ...	660.0	
6	0.0	line, organization, write, university, host, a...	526.0	
7	1.0	mail, information, post, send, list, internet,...	940.0	
8	4.0	gun, state, law, people, case, government, cri...	10.0	
9	7.0	god, christian, people, bible, religion, man, ...	935.0	
10	0.0	line, organization, write, university, host, a...	571.0	
11	12.0	game, team, year, play, player, win, good, sea...	446.0	
12	17.0	car, article, bike, good, organization, line, ...	1089.0	
13	12.0	game, team, year, play, player, win, good, sea...	241.0	
14	6.0	key, system, encryption, bit, technology, chip...	285.0	
15	17.0	car, article, bike, good, organization, line, ...	1392.0	
16	10.0	space, launch, system, earth, nasa, satellite,...	202.0	
17	4.0	gun, state, law, people, case, government, cri...	1048.0	
18	15.0	drive, card, problem, system, scsi, driver, wi...	128.0	
19	10.0	space, launch, system, earth, nasa, satellite,...	207.0	
20	7.0	god, christian, people, bible, religion, man, ...	NaN	
21	0.0	line, organization, write, university, host, a...	NaN	
22	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Doc
23	11.0	write, israel, article, israeli, jew, people, ...	NaN	
24	17.0	car, article, bike, good, organization, line, ...	NaN	
25	7.0	god, christian, people, bible, religion, man, ...	NaN	
26	13.0	make, year, work, money, pay, president, peopl...	NaN	
27	2.0	make, thing, time, write, good, people, start,...	NaN	
28	12.0	game, team, year, play, player, win, good, sea...	NaN	
29	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
...	...	...	...	
11284	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
11285	12.0	game, team, year, play, player, win, good, sea...	NaN	
11286	16.0	armenian, people, turkish, kill, woman, greek,...	NaN	
11287	6.0	key, system, encryption, bit, technology, chip...	NaN	
11288	9.0	window, image, file, program, version, display...	NaN	
11289	16.0	armenian, people, turkish, kill, woman, greek,...	NaN	
11290	7.0	god, christian, people, bible, religion, man, ...	NaN	
11291	0.0	line, organization, write, university, host, a...	NaN	
11292	3.0	drug, article, problem, organization, food, st...	NaN	
11293	7.0	god, christian, people, bible, religion, man, ...	NaN	
11294	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
11295	0.0	line, organization, write, university, host, a...	NaN	
11296	0.0	line, organization, write, university, host, a...	NaN	
11297	7.0	god, christian, people, bible, religion, man, ...	NaN	
11298	7.0	god, christian, people, bible, religion, man, ...	NaN	
11299	1.0	mail, information, post, send, list, internet,...	NaN	
11300	3.0	drug, article, problem, organization, food, st...	NaN	
11301	12.0	game, team, year, play, player, win, good, sea...	NaN	
11302	0.0	line, organization, write, university, host, a...	NaN	
11303	2.0	make, thing, time, write, good, people, start,...	NaN	
11304	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
11305	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
11306	5.0	line, sale, organization, work, power, good, b...	NaN	
11307	7.0	god, christian, people, bible, religion, man, ...	NaN	
11308	6.0	key, system, encryption, bit, technology, chip...	NaN	
11309	5.0	line, sale, organization, work, power, good, b...	NaN	
11310	17.0	car, article, bike, good, organization, line, ...	NaN	
11311	12.0	game, team, year, play, player, win, good, sea...	NaN	

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Doc
11312	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	
11313	15.0	drive, card, problem, system, scsi, driver, wi...	NaN	

11314 rows × 4 columns

In [ ]: