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=log

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

In [10]:
```
# 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/news
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 [11]:
```
# 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 [12]:
```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc

data_words = list(sent_to_words(data))

print(data_words[:1])
```

```
[['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nntp',
nd', 'college', 'park', 'lines', 'was', 'wondering', 'if', 'anyone', 'out', 'there
s', 'door', 'sports', 'car', 'looked', 'to', 'be', 'from', 'the', 'late', 'early',
ion', 'the', 'front', 'bumper', 'was', 'separate', 'from', 'the', 'rest', 'of', 't
'engine', 'specs', 'years', 'of', 'production', 'where', 'this', 'car', 'is', 'mad
'car', 'please', 'mail', 'thanks', 'il', 'brought', 'to', 'you', 'by', 'your', 'ne
```

In [13]:
```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher t
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', 'nntp_po
k', 'lines', 'was', 'wondering', 'if', 'anyone', 'out', 'there', 'could', 'enlight
s', 'car', 'looked', 'to', 'be', 'from', 'the', 'late', 'early', 'it', 'was', 'cal
t_bumper', 'was', 'separate', 'from', 'the', 'rest', 'of', 'the', 'body', 'this',
'years', 'of', 'production', 'where', 'this', 'car', 'is', 'made', 'history', 'or
il', 'thanks', 'il', 'brought', 'to', 'you', 'by', 'your', 'neighborhood', 'lerxst
```

In [14]:
```
# 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_wor

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 allowed_
    return texts_out
```

In [15]:
```
# 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', 'ADJ

print(data_lemmatized[:1])
```

```
[['where', 's', 'thing', 'car', 'nntp_post', 'host', 'umd', 'organization', 'unive
r', 'see', 'day', 'door', 'sport', 'car', 'look', 'late', 'early', 'call', 'brickl
ow', 'anyone', 'tellme', 'model', 'name', 'engine', 'specs', 'year', 'production',
borhood', 'lerxst']]
```

In [16]:
```
# 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, 2),
(20, 1), (21, 1), (22, 2), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1),
9, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1), (45, 1), (46, 1), (47, 1), (48
```

In [17]:
```
id2word[0]
```

Out[17]:
```
'addition'
```

```
In [18]: # Human readable format of corpus (term-frequency)
         [[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

```
Out[18]: [[('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_post', 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 [19]:
```
# 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 [20]:
```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.238*"window" + 0.099*"driver" + 0.031*"win" + 0.025*"professional" + '
  '0.023*"ohio_state" + 0.023*"rd" + 0.017*"microsoft" + '
  '0.016*"window_manager" + 0.015*"dos" + 0.014*"button"'),
 (1,
  '0.038*"card" + 0.034*"mail" + 0.026*"computer" + 0.025*"run" + '
  '0.025*"system" + 0.023*"software" + 0.023*"email" + 0.023*"sale" + '
  '0.020*"info" + 0.018*"thank"'),
 (2,
  '0.120*"file" + 0.046*"machine" + 0.043*"copy" + 0.041*"disk" + '
  '0.039*"peter" + 0.028*"bus" + 0.023*"case_western" + '
  '0.023*"reserve_university" + 0.022*"msg" + 0.016*"wide"'),
 (3,
  '0.019*"space" + 0.018*"information" + 0.016*"use" + 0.013*"include" + '
  '0.012*"also" + 0.012*"program" + 0.012*"system" + 0.012*"new" + '
  '0.011*"available" + 0.010*"provide"'),
 (4,
  '0.133*"bike" + 0.057*"ride" + 0.047*"motorcycle" + 0.021*"rid" + '
  '0.016*"rider" + 0.012*"safely" + 0.012*"steer" + 0.012*"riding" + '
  '0.009*"countersteere" + 0.006*"biker"'),
 (5,
  '0.836*"ax" + 0.059*"max" + 0.008*"mac" + 0.007*"mb" + 0.005*"scsi" + '
  '0.004*"meg" + 0.003*"serial" + 0.002*"kit" + 0.002*"db" + 0.002*"roman"'),
 (6,
  '0.036*"line" + 0.033*"organization" + 0.027*"write" + 0.024*"not" + '
  '0.022*"article" + 0.019*"get" + 0.018*"would" + 0.018*"be" + 0.017*"do" + '
  '0.015*"university"'),
 (7,
  '0.054*"image" + 0.040*"server" + 0.036*"memory" + 0.033*"sun" + '
  '0.031*"hour" + 0.031*"package" + 0.026*"normal" + 0.026*"blue" + '
  '0.021*"corporation" + 0.020*"burn"'),
 (8,
  '0.049*"water" + 0.048*"limit" + 0.045*"girl" + 0.039*"band" + '
  '0.036*"stupid" + 0.035*"husband" + 0.027*"marry" + 0.025*"jack" + '
  '0.024*"md" + 0.021*"music"'),
 (9,
  '0.017*"would" + 0.017*"people" + 0.013*"god" + 0.013*"believe" + '
  '0.011*"say" + 0.011*"christian" + 0.009*"mean" + 0.009*"life" + '
  '0.009*"make" + 0.009*"fact"'),
 (10,
  '0.035*"not" + 0.025*"say" + 0.023*"go" + 0.021*"do" + 0.018*"time" + '
  '0.016*"year" + 0.016*"come" + 0.015*"s" + 0.015*"see" + 0.012*"take"'),
 (11,
  '0.046*"government" + 0.037*"gun" + 0.027*"state" + 0.024*"law" + '
  '0.024*"american" + 0.022*"right" + 0.019*"country" + 0.018*"public" + '
  '0.016*"police" + 0.016*"protect"'),
 (12,
  '0.106*"key" + 0.087*"standard" + 0.039*"encryption" + 0.027*"clipper" + '
  '0.027*"clipper_chip" + 0.025*"hd" + 0.023*"bit" + 0.023*"algorithm" + '
  '0.022*"security" + 0.019*"secure"'),
 (13,
  '0.174*"game" + 0.165*"team" + 0.077*"play" + 0.064*"season" + 0.044*"goal" '
  '+ 0.042*"score" + 0.029*"canadian" + 0.025*"pin" + 0.024*"playoff" + '
  '0.020*"draft"'),
 (14,
  '0.104*"hockey" + 0.057*"fan" + 0.037*"radio" + 0.034*"andrew" + '
  '0.034*"morning" + 0.031*"hall" + 0.028*"station" + 0.026*"baseball" + '
  '0.025*"coverage" + 0.024*"sport"'),
 (15,
  '0.071*"contact" + 0.047*"newsgroup" + 0.046*"pittsburgh" + 0.041*"medium" + '
  '0.024*"islam" + 0.024*"excellent" + 0.023*"surface" + 0.019*"gordon_bank" + '
  '0.017*"edition" + 0.015*"docs"'),
 (16,
  '0.092*"israel" + 0.068*"israeli" + 0.051*"display" + 0.046*"arab" + '
  '0.028*"mhz" + 0.028*"internal" + 0.025*"output" + 0.025*"input" + '
  '0.018*"shipping" + 0.018*"multiple"'),
 (17,
  '0.124*"president" + 0.077*"job" + 0.070*"steve" + 0.047*"mike" + '
  '0.037*"dave" + 0.031*"brother" + 0.025*"palestinian" + 0.023*"success" + '
  '0.022*"newspaper" + 0.020*"gas"'),
 (18,
  '0.053*"chicago" + 0.049*"hawk" + 0.049*"april" + 0.039*"ibm" + '
  '0.030*"period" + 0.028*"reader" + 0.025*"steven" + 0.021*"yesterday" + '
  '0.020*"status" + 0.019*"carnegie_mellon"'),
 (19,
  '0.206*"chip" + 0.029*"dan" + 0.022*"serial_number" + 0.017*"environmental" '
  '+ 0.016*"exe" + 0.015*"integrate" + 0.008*"moore" + 0.005*"stack" + '
  '0.004*"suspicious" + 0.002*"western_australia"')]
```

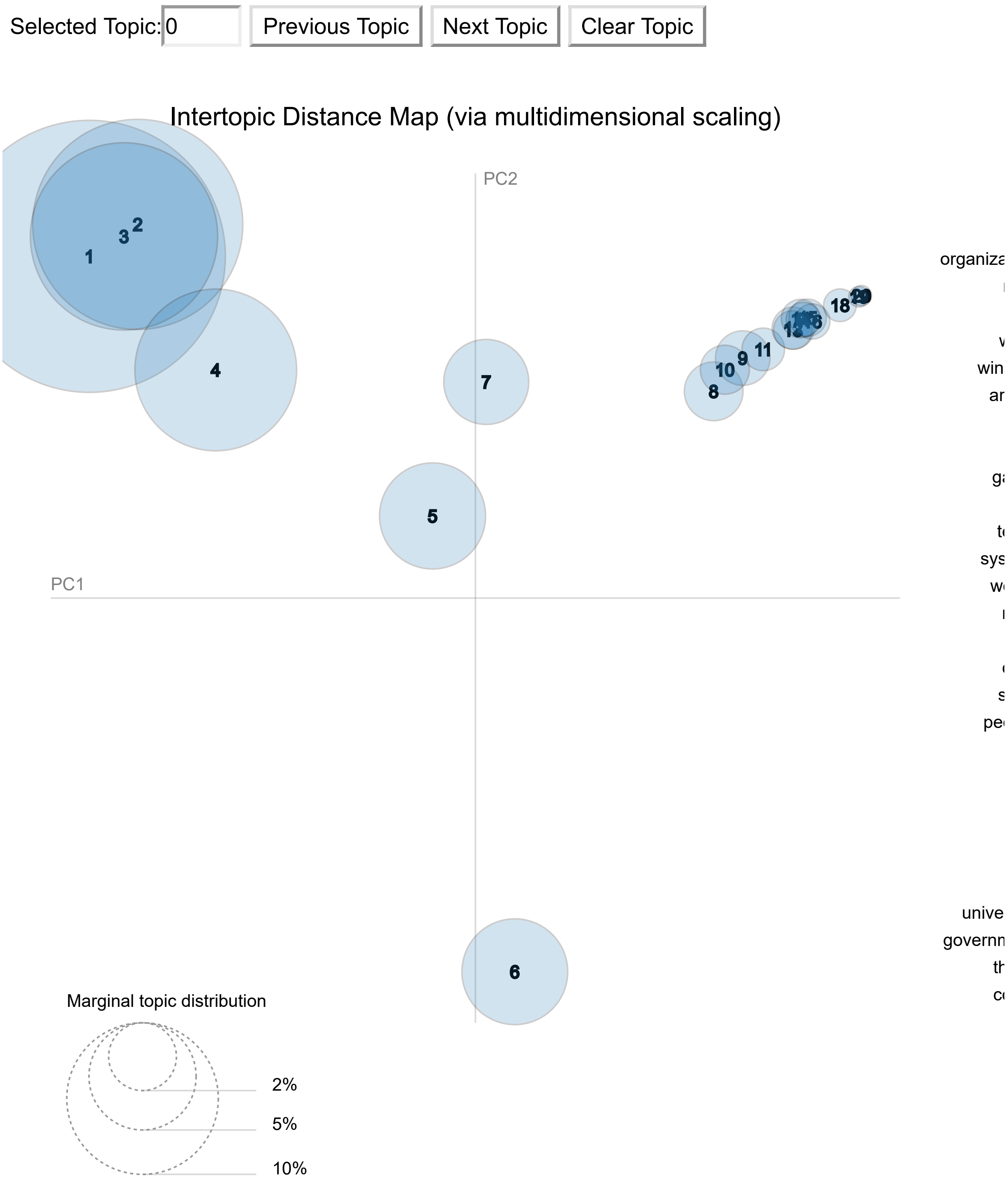
In [21]:

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

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

```
Perplexity:  -14.708910121024964

Coherence Score:  0.47647829731491387
```

In [22]:
```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

Out[22]:



In [34]:
```
# Download File: http://mallet.cs.umass.edu/dist/mallet-2.0.8.zip
import os
from gensim.models.wrappers import LdaMallet

os.environ['MALLET_HOME'] = 'C:\\Mallet'
mallet_path = 'C:\\Mallet\\bin\\mallet'
#mallet_path = 'C:/Mallet' # update this path
ldamallet = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_topi
```

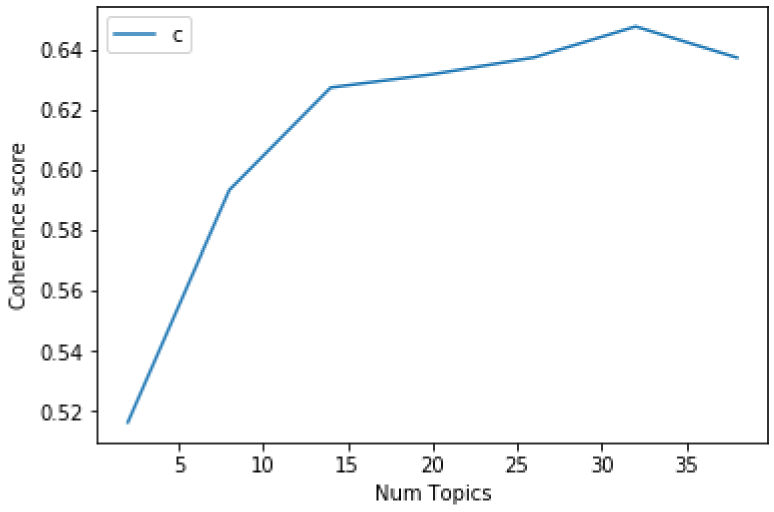
In [35]:
```
# Show Topics
pprint(ldamallet.show_topics(formatted=False))

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

```
   ('earth', 0.00666948475881828),
   ('center', 0.006559892286255539),
   ('year', 0.006450299813692797),

   ('research', 0.006325051273621092),
   ('nasa', 0.006090210260986645),
   ('cost', 0.006074554193477682)]),
 (16,
  [('line', 0.045045860487569395),
   ('organization', 0.04474414675356022),
   ('_', 0.02271904417089066),
   ('ca', 0.016624426743905383),
   ('md', 0.006034274680183442),
   ('air', 0.005642046825971518),
   ('te', 0.005129133478155926),
   ('ed', 0.005098962104755009),
   ('mv', 0.004344677769732078),
   ('reply', 0.0038921071687183202)])]
```

In [36]:
```
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 respe
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dict
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

In [37]:
```
# Can take a long time to run.
model_list, coherence_values = compute_coherence_values(dictionary=id2word, corpu
```

In [38]:
```
# 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 [39]:
```
# 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 0.5162
Num Topics = 8  has Coherence Value of 0.5933
Num Topics = 14  has Coherence Value of 0.6274
Num Topics = 20  has Coherence Value of 0.6318
Num Topics = 26  has Coherence Value of 0.6374
Num Topics = 32  has Coherence Value of 0.6477
Num Topics = 38  has Coherence Value of 0.6373
```

In [40]:
```
# 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,
  '0.029*"god" + 0.018*"christian" + 0.010*"people" + 0.009*"bible" + '
  '0.009*"religion" + 0.007*"church" + 0.007*"faith" + 0.007*"man" + '
  '0.007*"word" + 0.007*"law"'),
 (1,
  '0.033*"file" + 0.033*"window" + 0.020*"line" + 0.017*"program" + '
  '0.016*"problem" + 0.010*"write" + 0.010*"set" + 0.010*"read" + '
  '0.010*"change" + 0.009*"error"'),
 (2,
  '0.118*"write" + 0.099*"article" + 0.080*"organization" + 0.078*"line" + '
  '0.036*"host" + 0.025*"university" + 0.024*"nntp_post" + 0.021*"reply" + '
  '0.021*"nntp_posting" + 0.011*"hear"'),
 (3,
  '0.018*"gun" + 0.012*"people" + 0.011*"state" + 0.011*"armenian" + '
  '0.011*"law" + 0.007*"government" + 0.007*"crime" + 0.006*"weapon" + '
  '0.006*"kill" + 0.006*"turkish"'),
 (4,
  '0.025*"drive" + 0.019*"card" + 0.016*"system" + 0.015*"problem" + '
  '0.013*"driver" + 0.013*"scsi" + 0.012*"work" + 0.011*"bit" + 0.010*"mac" + '
  '0.009*"monitor"'),
 (5,
  '0.038*"line" + 0.027*"organization" + 0.023*"_" + 0.018*"ca" + '
  '0.006*"reply" + 0.006*"air" + 0.006*"ed" + 0.006*"md" + 0.005*"mv" + '
  '0.004*"te"'),
 (6,
  '0.009*"drug" + 0.008*"problem" + 0.007*"study" + 0.007*"food" + '
  '0.006*"doctor" + 0.006*"day" + 0.006*"effect" + 0.006*"time" + '
  '0.006*"research" + 0.006*"medical"'),
 (7,
  '0.031*"key" + 0.013*"system" + 0.011*"encryption" + 0.010*"bit" + '
  '0.009*"security" + 0.009*"government" + 0.008*"technology" + 0.008*"chip" + '
  '0.008*"public" + 0.007*"message"'),
 (8,
  '0.024*"year" + 0.019*"good" + 0.013*"game" + 0.013*"organization" + '
  '0.012*"run" + 0.012*"line" + 0.010*"hit" + 0.009*"win" + 0.008*"lose" + '
  '0.007*"team"'),
 (9,
  '0.035*"write" + 0.028*"good" + 0.025*"people" + 0.025*"make" + '
  '0.024*"thing" + 0.019*"article" + 0.015*"organization" + 0.014*"line" + '
  '0.011*"bad" + 0.011*"opinion"'),
 (10,
  '0.069*"organization" + 0.065*"line" + 0.050*"university" + 0.039*"host" + '
  '0.021*"nntp_post" + 0.021*"nntp_posting" + 0.016*"sale" + 0.015*"reply" + '
  '0.013*"price" + 0.012*"mail"'),
 (11,
  '0.012*"people" + 0.012*"israel" + 0.011*"kill" + 0.009*"leave" + '
  '0.009*"israeli" + 0.008*"jew" + 0.008*"time" + 0.007*"call" + 0.007*"arab" '
  '+ 0.007*"live"'),
 (12,
  '0.019*"post" + 0.017*"information" + 0.016*"send" + 0.015*"mail" + '
  '0.015*"list" + 0.014*"group" + 0.013*"book" + 0.012*"address" + '
  '0.010*"internet" + 0.009*"include"'),
 (13,
  '0.022*"game" + 0.019*"team" + 0.015*"play" + 0.010*"player" + '
  '0.010*"hockey" + 0.008*"year" + 0.008*"win" + 0.007*"goal" + 0.006*"season" '
  '+ 0.006*"line"'),
 (14,
  '0.017*"point" + 0.016*"question" + 0.012*"reason" + 0.011*"make" + '
  '0.010*"claim" + 0.010*"exist" + 0.010*"argument" + 0.009*"evidence" + '
  '0.008*"true" + 0.008*"thing"'),
 (15,
  '0.864*"ax" + 0.059*"max" + 0.002*"qax" + 0.002*"qq" + 0.001*"mb" + '
  '0.001*"giz" + 0.001*"mf" + 0.001*"bs" + 0.001*"mq" + 0.001*"tm"'),
 (16,
  '0.024*"space" + 0.007*"launch" + 0.007*"earth" + 0.007*"system" + '
  '0.006*"center" + 0.006*"nasa" + 0.006*"project" + 0.006*"research" + '
  '0.006*"datum" + 0.006*"satellite"'),
 (17,
  '0.014*"image" + 0.010*"version" + 0.010*"display" + 0.010*"graphic" + '
  '0.009*"server" + 0.009*"software" + 0.009*"color" + 0.009*"program" + '
  '0.008*"window" + 0.008*"application"'),
 (18,
  '0.025*"car" + 0.010*"bike" + 0.006*"back" + 0.006*"light" + 0.006*"drive" + '
  '0.005*"turn" + 0.005*"good" + 0.005*"engine" + 0.005*"dod" + 0.005*"buy"'),
 (19,
  '0.013*"make" + 0.013*"work" + 0.011*"money" + 0.010*"people" + '
  '0.009*"president" + 0.009*"pay" + 0.008*"year" + 0.008*"state" + '
  '0.008*"job" + 0.008*"government"')]
```

In [41]:
```
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 documen
        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_num),
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywo

    # 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, corpus=

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

# Show
df_dominant_topic.head(10)
```

Out[41]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords
0	0	18.0	0.1726	car, bike, back, light, drive, turn, good, eng...
1	1	4.0	0.2143	drive, card, system, problem, driver, scsi, wo...
2	2	18.0	0.2497	car, bike, back, light, drive, turn, good, eng...
3	3	17.0	0.2448	image, version, display, graphic, server, soft...
4	4	4.0	0.2157	drive, card, system, problem, driver, scsi, wo...
5	5	18.0	0.4382	car, bike, back, light, drive, turn, good, eng...
6	6	10.0	0.1806	organization, line, university, host, nntp_pos...
7	7	10.0	0.1301	organization, line, university, host, nntp_pos...
8	8	3.0	0.1699	gun, people, state, armenian, law, government,...
9	9	14.0	0.2727	point, question, reason, make, claim, exist, a...

In [42]:
```
# 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_Contribution'
                                            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", "Keywor

# Show
sent_topics_sorteddf_mallet.head()
```

Out[42]:

	Topic_Num	Topic_Perc_Contrib	Keywords	
0	0.0	0.7765	god, christian, people, bible, religion, churc...	From: (Robert Weiss
1	1.0	0.9691	file, window, line, program, problem, write, s...	From: (Landon C. Nol
2	2.0	0.5212	write, article, organization, line, host, univ...	From: (Jim De Arras) S
3	3.0	0.7325	gun, people, state, armenian, law, government,...	From: (Serdar Argic)
4	4.0	0.8159	drive, card, system, problem, driver, scsi, wo...	From: (D. Keith Rice

In [43]:
```
# 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_Keywords']

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

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

# Show
df_dominant_topics
```

Out[43]:

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Docume
0	18.0	car, bike, back, light, drive, turn, good, eng...	853.0	0.07
1	4.0	drive, card, system, problem, driver, scsi, wo...	421.0	0.03
2	18.0	car, bike, back, light, drive, turn, good, eng...	501.0	0.04
3	17.0	image, version, display, graphic, server, soft...	550.0	0.04
4	4.0	drive, card, system, problem, driver, scsi, wo...	1328.0	0.11
5	18.0	car, bike, back, light, drive, turn, good, eng...	181.0	0.01
6	10.0	organization, line, university, host, nntp_pos...	510.0	0.04
7	10.0	organization, line, university, host, nntp_pos...	542.0	0.04
8	3.0	gun, people, state, armenian, law, government,...	579.0	0.05
9	14.0	point, question, reason, make, claim, exist, a...	367.0	0.03
10	7.0	key, system, encryption, bit, security, govern...	1061.0	0.09
11	8.0	year, good, game, organization, run, line, hit...	372.0	0.03
12	9.0	write, good, people, make, thing, article, org...	307.0	0.02
13	13.0	game, team, play, player, hockey, year, win, g...	605.0	0.05
14	7.0	key, system, encryption, bit, security, govern...	324.0	0.02
15	18.0	car, bike, back, light, drive, turn, good, eng...	10.0	0.00
16	16.0	space, launch, earth, system, center, nasa, pr...	598.0	0.05
17	3.0	gun, people, state, armenian, law, government,...	765.0	0.06
18	4.0	drive, card, system, problem, driver, scsi, wo...	1143.0	0.10
19	16.0	space, launch, earth, system, center, nasa, pr...	297.0	0.02
20	0.0	god, christian, people, bible, religion, churc...	NaN	N
21	6.0	drug, problem, study, food, doctor, day, effec...	NaN	N
22	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N
23	11.0	people, israel, kill, leave, israeli, jew, tim...	NaN	N

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Docume
24	18.0	car, bike, back, light, drive, turn, good, eng...	NaN	N
25	0.0	god, christian, people, bible, religion, churc...	NaN	N
26	19.0	make, work, money, people, president, pay, yea...	NaN	N
27	9.0	write, good, people, make, thing, article, org...	NaN	N
28	8.0	year, good, game, organization, run, line, hit...	NaN	N
29	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N
...	...	...	...	
11284	2.0	write, article, organization, line, host, univ...	NaN	N
11285	8.0	year, good, game, organization, run, line, hit...	NaN	N
11286	11.0	people, israel, kill, leave, israeli, jew, tim...	NaN	N
11287	7.0	key, system, encryption, bit, security, govern...	NaN	N
11288	1.0	file, window, line, program, problem, write, s...	NaN	N
11289	3.0	gun, people, state, armenian, law, government,...	NaN	N
11290	0.0	god, christian, people, bible, religion, churc...	NaN	N
11291	17.0	image, version, display, graphic, server, soft...	NaN	N
11292	6.0	drug, problem, study, food, doctor, day, effec...	NaN	N
11293	2.0	write, article, organization, line, host, univ...	NaN	N
11294	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N
11295	10.0	organization, line, university, host, nntp_pos...	NaN	N
11296	10.0	organization, line, university, host, nntp_pos...	NaN	N
11297	9.0	write, good, people, make, thing, article, org...	NaN	N
11298	0.0	god, christian, people, bible, religion, churc...	NaN	N
11299	12.0	post, information, send, mail, list, group, bo...	NaN	N
11300	6.0	drug, problem, study, food, doctor, day, effec...	NaN	N
11301	8.0	year, good, game, organization, run, line, hit...	NaN	N
11302	18.0	car, bike, back, light, drive, turn, good, eng...	NaN	N
11303	9.0	write, good, people, make, thing, article, org...	NaN	N
11304	10.0	organization, line, university, host, nntp_pos...	NaN	N
11305	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N
11306	5.0	line, organization, _, ca, reply, air, ed, md,...	NaN	N
11307	0.0	god, christian, people, bible, religion, churc...	NaN	N
11308	7.0	key, system, encryption, bit, security, govern...	NaN	N
11309	18.0	car, bike, back, light, drive, turn, good, eng...	NaN	N
11310	18.0	car, bike, back, light, drive, turn, good, eng...	NaN	N
11311	13.0	game, team, play, player, hockey, year, win, g...	NaN	N
11312	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Docume
11313	4.0	drive, card, system, problem, driver, scsi, wo...	NaN	N

11314 rows × 4 columns

```
In [ ]:
```