# Topic_Modeling

April 20, 2021

#

Topic modeling

#### 0.0.1 In this notebook, we are going to use LDA (Latent Dirichlet Allocation) to answer the following questions:

**Q1: Given the titles that involve an entity (e.g. United States, China, France … etc), what are the topics covered by the given dataset involving that entity?**

```
Examples of titles related to China:
- Russia, China challenge US with proposal to ban space weapons!
- China considers ending one-child policy!
- Chinese Troops Encriple Tibetan Monestaries.   So Hussein was evil for invading Kuweit, but.
- Chinese troops surround monasteries in Tibet!
- Riots break out in Tibetan capital!
```

**Q2: Given a "title" T, what are the most similar titles to T in the dataset?**

[1]:
```
import sys, os, re, pprint, time, multiprocessing
import pandas as pd
import numpy as np

import nltk
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
from collections import defaultdict

from gensim import models, corpora, similarities, matutils
from gensim.models import Phrases
from gensim.corpora import Dictionary
from gensim.models import AuthorTopicModel
from gensim.models import atmodel
from gensim.similarities import MatrixSimilarity


pd.set_option("display.precision", 2)
pd.set_option('display.max_columns', None)
```

```python
pd.set_option('display.max_colwidth', None)
```

## 0.1 Loading data

[2]:
```python
data_path = 'Eluvio_DS_Challenge.csv'
df = pd.read_csv(data_path)

print(df.shape[0])
```

509236

## 0.2 Preprocessing data/titles, in this step we will perform a series of preprocessing tasks on the title attribute to prepare it for the next steps.

## 0.3 Preprocessing steps:

### 0.3.1 1. `clean_and_fix`: in this step, we convert each title to its lower case, and do some cleaning steps, such as:

#### a. convert abbreviations to their normal form, for exampe 'U.S.' would be replaced with 'united states' #### b. replace characters as '-' and '_' with space ' ' ### 2. Remove numbers, symbols ... etc, and only keep English alphbets. This step will filter out articles written in non-English languages ### 3. Tokenizing each title into a list of words ### 4. Removing stop words ### 5. Lemmatize all words ### 6. Remove short words, where size(word) < 3

[3]:
```python
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tokenize import WordPunctTokenizer
# for bigrams and trigrams
from gensim.models import Phrases

lemma = nltk.wordnet.WordNetLemmatizer()
stemmer = SnowballStemmer('english')
stop_words = nltk.corpus.stopwords.words('english')


def tokenize_removeStopWords_stem(text):
    # tokenize
    words = WordPunctTokenizer().tokenize(text)
    # remove stop words
    words = [word for word in words if word not in stopwords.words('english')]
    # stem each word
    words = [stemmer.stem(word) for word in words]
    return words
```

```python
def tokenize_removeStopWords_lemmatize(text):
    # tokenize
    words = WordPunctTokenizer().tokenize(text)
    # remove stop words
    words = [word for word in words if word not in stopwords.words('english')]
    # stem each word
    words = [lemma.lemmatize(word) for word in words]
    return words

def to_lower(txt):
    return txt.lower()

def remove_non_alphabetic(txt):
    regex = re.compile('[^a-zA-Z ]+')
    return regex.sub('', txt)

def de_abbreviate(txt):
    txt = txt.lower()

    txt = txt.replace('-', ' ')
    txt = txt.replace('_', ' ')

    if txt.startswith('us '):
        txt = 'united states ' + txt[2:]
    if txt.startswith('uk '):
        txt = 'united kindom ' + txt[2:]
    if txt.startswith('un '):
        txt = 'united nations ' + txt[2:]
    if txt.startswith('eu '):
        txt = 'europe ' + txt[2:]

    if txt.endswith(' us'):
        txt = txt[0:len(txt) - 2] + ' united states'
    if txt.endswith(' uk'):
        txt = txt[0:len(txt) - 2] + ' united kindom'
    if txt.endswith(' un'):
        txt = txt[0:len(txt) - 2] + ' united nations'
    if txt.endswith(' eu'):
        txt = txt[0:len(txt) - 2] + ' europe'

    txt = txt.replace('u.s.', 'united states')
    txt = txt.replace('u.k.', 'united kingdom')
    txt = txt.replace('u.n.', 'united nations')
    txt = txt.replace('e.u.', 'europe')

    txt = txt.replace(' us ', 'united states')
```

```
    txt = txt.replace(' uk ', 'united kingdom')
    txt = txt.replace(' un ', 'united nations')
    txt = txt.replace(' eu ', 'europe')

    return txt

def remove_short_words(title):
    return [word for word in title if len(word) >= 3]

def preprocess(titles):
    start = time.time()
    pool = multiprocessing.Pool()

    # remove abbreviations, such as U.S., EU ... etc and replace with the full␣
↪names
    # note that this function (de_abbreviate) will convert to lower case too
    titles = list(pool.map( de_abbreviate, titles ))
    # remove non-alphabet charaters, yet keep white spaces
    titles = list(pool.map( remove_non_alphabetic, titles ))
    # tokenize, remove stop words, and stem
    titles = list(pool.map( tokenize_removeStopWords_lemmatize, titles ))

    titles = list(pool.map( remove_short_words, titles ))
    end = time.time()
    print(f"Runtime of the preprocess function is {end - start}")

    pool.close()
    return (titles)

preprocessed_titles = preprocess(df['title'].to_list())

# compute bigrams
bigram = Phrases(preprocessed_titles, min_count=2, threshold=5)

# Now, append bigrams preprocessed titles
for index in range(len(preprocessed_titles)):
    for token in bigram[preprocessed_titles[index]]:
        if '_' in token:
            # Token is a bigram, add to document.
            preprocessed_titles[index].append(token)
```

```
Runtime of the preprocess function is 81.39524817466736
```

[4]:
```
df['preprocessed_title'] = preprocessed_titles
print(df[['title', 'preprocessed_title']][0:5])
```

```
                                              title  \
```

```
0                 Scores killed in Pakistan clashes
1                  Japan resumes refuelling mission
2                   US presses Egypt on Gaza border
3      Jump-start economy: Give health care to all
4   Council of Europe bashes EU&UN terror blacklist

preprocessed_title
0                                       [score, killed, pakistan, clash,
score_killed]
1                                                 [japan, resume, refuelling,
mission]
2               [united, state, press, egypt, gaza, border, united_state,
gaza_border]
3                  [jump, start, economy, give, health, care, jump_start,
health_care]
4   [council, europe, bash, euun, terror, blacklist, council_europe,
terror_blacklist]
```

# 1 Next, we build the dictionary, corpus and the mapping from document to author that we will use to train our *AuthorTopicModel* model in gensim

# 2 The function `build_dictioanry_and_corpus()` builds and returns:

## 2.1 - Dictionary

## 2.2 - Corpus

[5]:
```python
def build_dictioanry_and_corpus(df):
    titles = df['preprocessed_title'].to_list()
    dictionary = corpora.Dictionary(titles)
    # filter extremes: we keep all topkens/words in our dictionary regardless␣
 ↪of their frequencies
    dictionary.filter_extremes(no_below=1, no_above=1)
    corpus = [dictionary.doc2bow(title_words) for title_words in titles]
    doc2author = dict( zip( range(df.shape[0]), list(df.author.values) ) )
    return dictionary, corpus, doc2author
```

[6]:
```python
dictionary, corpus, doc2author = build_dictioanry_and_corpus(df)
```

# 3 Now, let's look at our dictionary and corpus, and check if there are any empty documents due to preprocessing steps

[7]:
```
print('Number of authors: ', len(df.author.unique()))
print('Number of unique words: ', len(dictionary))
print('Number of documents: ', len(corpus))
print('Number of doc2author: ', len(doc2author))

items = list(doc2author.items())#[0:10]

count_empty = 0
empty_docs = list()
for key, value in items:
#     print(key, value)
    corp = corpus[key]
    if not any(corp):
        count_empty += 1
        print('Document # ', key, ' is empty')
        print(corpus[key])
        print('Title: ', df['title'][key])
        print()
        empty_docs.append(key)

print('Empty documents/bag-of-words = ', count_empty)
print(empty_docs)
```

```
Number of authors:  85838
Number of unique words:  100000
Number of documents:  509236
Number of doc2author:  509236
Document #  577  is empty
[]
Title:  CHCNK OUT

Document #  1226  is empty
[]
Title:  Najnowsze wydarzenia, fakty i informacje - HotInfos.org

Document #  1368  is empty
[]
Title:          ..

Document #  1862  is empty
[]
Title:       :

Document #  3690  is empty
```

[]
Title:  The Touchstone

Document #  4322  is empty
[]
Title:

Document #  6281  is empty
[]
Title:  Irena Sendler

Document #  6324  is empty
[]
Title:  Mmiss universo - 1952 - 2005

Document #  7368  is empty
[]
Title:  Thinspiration

Document #  8405  is empty
[]
Title:  69, 127

Document #  8873  is empty
[]
Title:  duthel.org

Document #  8986  is empty
[]
Title:  Fiere Belg op zijn einde?

Document #  9116  is empty
[]
Title:   ANGOLO DI MARKINO : UOMINI LIBERI

Document #  9267  is empty
[]
Title:  Elefantiasi

Document #  9538  is empty
[]
Title:  Cyd Charisse dies in LA

Document #  10113  is empty
[]
Title:  FRONTLINE/WORLD | PBS

Document #  10584  is empty

```
[]
Title:

Document #  12255  is empty
[]
Title:  Burma:+Buddhist+migrants+pressured+to+convert+to+Christianity

Document #  12740  is empty
[]
Title:  www.alltop.com

Document #  12998  is empty
[]
Title:  Königssee

Document #  18836  is empty
[]
Title:  doctorsgirlsbymiminajh9 : My[confined]Space

Document #  19060  is empty
[]
Title:  International+Condemnation;+Continuity+in+Burma

Document #  19135  is empty
[]
Title:  Baghdadophobia

Document #  19681  is empty
[]
Title:  ?

Document #  19909  is empty
[]
Title:  StopPovertyNow.org

Document #  21004  is empty
[]
Title:       24

Document #  21545  is empty
[]
Title:  Motorola,Proxim,redline

Document #  21618  is empty
[]
Title:  PCDJ VJ 5.2

Document #  21766  is empty
```

```
[]
Title:  Just Testting

Document #  22047  is empty
[]
Title:      19

Document #  23450  is empty
[]
Title:  meca

Document #  24444  is empty
[]
Title:  www.futureme.org

Document #  24560  is empty
[]
Title:  Whoopysweb

Document #  25111  is empty
[]
Title:  666

Document #  25498  is empty
[]
Title:  RomboloDjRadioDragon

Document #  25531  is empty
[]
Title:  veggie/vegan

Document #  26070  is empty
[]
Title:  Ironnnnnnnny

Document #  26127  is empty
[]
Title:  DeutschAkademie in Weihnachtsstimmung

Document #  26226  is empty
[]
Title:  Sangkar Dewi Rembulan

Document #  26319  is empty
[]
Title:  Shtypi

Document #  26405  is empty
```

```
[]
Title:  hmu111 on Technorati

Document #  26483  is empty
[]
Title:  newageislam

Document #  26874  is empty
[]
Title:  http://www.comcast.net/data/fan/html/popup.html?v=1030143489&pl=Comcast/
1030032788.xml&launchpoint=Cover&cid=fancover&attr=default_headline&config=/conf
ig/common/fan/default.xml

Document #  27067  is empty
[]
Title:  Arbeitsplatzverlust

Document #  27070  is empty
[]
Title:  Auslandskrankenversicherungen

Document #  27250  is empty
[]
Title:  mashiko.beridze@yahoo.com

Document #  27453  is empty
[]
Title:  Hypnotherapist

Document #  27532  is empty
[]
Title:  SpotXchange

Document #  27699  is empty
[]
Title:  Freemusicwebsites

Document #  28105  is empty
[]
Title:  Privat Ferienhäuser Toskana

Document #  28293  is empty
[]
Title:  Republikáni kritizují obrovský Obamův rozpočet - v hodnotě 3,5 bilionu
dolarů

Document #  28751  is empty
[]
```

Title:  -  |

Document # 28783 is empty
[]
Title: The mistery of Fr. Giussani

Document # 29014 is empty
[]
Title: ΔΩΡΕΑΝ ΠΡΟΓΝΩΣΤΙΚΑ ΣΤΟΙΧΗΜΑΤΟΣ + ΖΩΝΤΑΝΑ ΣΚΟΡ

Document # 29878 is empty
[]
Title: I preservativi aumentano i problemi

Document # 29901 is empty
[]
Title: Clube de Criação do Paraná

Document # 30092 is empty
[]
Title: Zwrot podatku z zagranicy

Document # 30268 is empty
[]
Title: Sangue cordonale: donazione cordone ombelicale

Document # 30588 is empty
[]
Title: Doften av vårsnö I Doroea

Document # 30747 is empty
[]
Title: Wochenzeitung

Document # 30750 is empty
[]
Title: Umfrage be2

Document # 31570 is empty
[]
Title: >Emprestimo Pessoal, Crédito Pessoal, Financiamento e Empréstimos

Document # 31687 is empty
[]
Title: Ratenkredit

Document # 31952 is empty
[]

Title:  skrivadur tikt & klikt

Document #  32529  is empty
[]
Title:  MYCHANNEL

Document #  32968  is empty
[]
Title:

Document #  33664  is empty
[]
Title:  Despre independenta statelor, hegelianism si societatea deschisa

Document #  34074  is empty
[]
Title:  Wolverine, lupul fara coada si surprize

Document #  34106  is empty
[]
Title:  Il sogno americano di Gianluca - spettacolo -Tgcom - pagina 1

Document #  34222  is empty
[]
Title:  EdmundoTV.ru:

Document #  34323  is empty
[]
Title:  Currahee

Document #  34506  is empty
[]
Title:  sellgoldjewellry

Document #  35129  is empty
[]
Title:  Conditia inumana 24 - conditii de munca imposibile

Document #  35168  is empty
[]
Title:  Hemoroizi. Fisuri anale - Tratament

Document #  35662  is empty
[]
Title:  Webinars

Document #  36451  is empty
[]

```
Title:    -

Document #  36588  is empty
[]
Title:          ,              ,,,,

Document #  36637  is empty
[]
Title:  Hopa da Feci Kaza

Document #  38071  is empty
[]
Title:  Chantel McNulty

Document #  38609  is empty
[]
Title:  Sprichwörter

Document #  38727  is empty
[]
Title:  Oyun haberleri

Document #  38846  is empty
[]
Title:  KECURANGAN PILPRES 2009

Document #  38865  is empty
[]
Title:  Pchocasi.Com Bilgi Paylasim Platformu

Document #  38884  is empty
[]
Title:  Interwetten Testsieger beim Sportwetten Vergleichstest

Document #  39164  is empty
[]
Title:  Φ      Γ

Document #  39501  is empty
[]
Title:  Gambaran Skenario Kecurangan Pilpres

Document #  39576  is empty
[]
Title:  Mengembalikan Jati Diri Bangsa

Document #  39593  is empty
[]
```

Title: Udaipur

Document # 40436 is empty
[]
Title: MR

Document # 41690 is empty
[]
Title: Selbstcoaching Tipps gegen Prokrastination / Aufschieberitis

Document # 41723 is empty
[]
Title: ulkoporealtaat | poreammeet

Document # 43391 is empty
[]
Title: Aktuality Zprávy Události Komentáře Blogy

Document # 44707 is empty
[]
Title: Hahahahahahaha…(pause)…Hahahahahahaha…

Document # 45126 is empty
[]
Title: Frases de caminhoneiros e pára-choque de caminhão

Document # 45949 is empty
[]
Title:

Document # 49207 is empty
[]
Title: Pashtunwali

Document # 50832 is empty
[]
Title: TILAK

Document # 52282 is empty
[]
Title: Ferienwohnung in Wasserburg am Bodensee

Document # 52521 is empty
[]
Title: Avrupa nın En Büyük Ahşap Yapısı!

Document # 55477 is empty
[]

Title: { _ }

Document # 56854 is empty
[]
Title: Here we go again…

Document # 59409 is empty
[]
Title: Komunistyczna cenzura na Facebooku

Document # 62302 is empty
[]
Title: Neuer Impfstoff schützt auch vor Schweinegrippe

Document # 62965 is empty
[]
Title: Feedreports

Document # 64095 is empty
[]
Title: The Other 9/11

Document # 64745 is empty
[]
Title: Amostras de lactacyd grátis

Document # 67886 is empty
[]
Title: Morto e condannato

Document # 71453 is empty
[]
Title: Alhamdulillah

Document # 71716 is empty
[]
Title: ibuyeco

Document # 71896 is empty
[]
Title: IL NUCLEARE ED I GIOCHI SOSPETTI

Document # 71994 is empty
[]
Title: Il testamento di Shaban

Document # 72840 is empty
[]

Title:  abhinav.com

Document #  77510  is empty
[]
Title:

Document #  78252  is empty
[]
Title:  _ … no. HA HA HA HA HA !!!

Document #  80929  is empty
[]
Title:  http://www.atimes.com/atimes/Middle_East/MC19Ak04.html

Document #  85327  is empty
[]
Title:  Locum Tenens

Document #  87740  is empty
[]
Title:  Låt skorna bestämma

Document #  88491  is empty
[]
Title:  Farhan and Ambreen

Document #  92065  is empty
[]
Title:  Exatidão Científica

Document #  93479  is empty
[]
Title:  bisi sant

Document #  95352  is empty
[]
Title:  IMMIGRATI RIBELLI ED ESASPERATI

Document #  110514  is empty
[]
Title:  ps3 Jb2 - ps3 Jb2

Document #  145546  is empty
[]
Title:

Document #  147351  is empty
[]

```
Title:  Blair: UK and EU need each other

Document #  149534  is empty
[]
Title:  Prunes are not a laxative, EU rules

Document #  155364  is empty
[]
Title:  BALOTU-BULANIK (MUŞ) Depremi, 19 Ocak 2013 Cumartesi - 01:04

Document #  163052  is empty
[]
Title:

Document #  163069  is empty
[]
Title:

Document #  167276  is empty
[]
Title:   Marmageddon!

Document #  173527  is empty
[]
Title:  Bitcointyler

Document #  173976  is empty
[]
Title:  middleeastnews

Document #  175989  is empty
[]
Title:  Autism=atheism

Document #  176340  is empty
[]
Title:

Document #  176962  is empty
[]
Title:       allegro

Document #  178880  is empty
[]
Title:  RabatPress

Document #  179349  is empty
[]
```

Title:  Curaçaose politicus Helmin Wiels doodgeschoten

Document #  179780  is empty
[]
Title:

Document #  180893  is empty
[]
Title:

Document #  184330  is empty
[]
Title:  .

Document #  186068  is empty
[]
Title:  Mrs. Ingle

Document #  186916  is empty
[]
Title:  Chemiewaffen in Syrien

Document #  209575  is empty
[]
Title:  This just in from TheBloodyObvious.com

Document #  217144  is empty
[]
Title:  Huronia

Document #  217805  is empty
[]
Title:

Document #  246641  is empty
[]
Title:  Copsicles

Document #  248030  is empty
[]
Title:  Cara Alami Menghilangkan Bulu Ketiak

Document #  257186  is empty
[]
Title:  Explosionen in Doha/Qatar

Document #  335912  is empty
[]

Title:   Biji Serok Apo  sloganına ifade özgürlüğü beraatı

Document #  341597  is empty
[]
Title:

Document #  352127  is empty
[]
Title:  Vaillant Servis

Document #  396846  is empty
[]
Title:  Untitled

Empty documents/bag-of-words =  151
[577, 1226, 1368, 1862, 3690, 4322, 6281, 6324, 7368, 8405, 8873, 8986, 9116, 9267, 9538, 10113, 10584, 12255, 12740, 12998, 18836, 19060, 19135, 19681, 19909, 21004, 21545, 21618, 21766, 22047, 23450, 24444, 24560, 25111, 25498, 25531, 26070, 26127, 26226, 26319, 26405, 26483, 26874, 27067, 27070, 27250, 27453, 27532, 27699, 28105, 28293, 28751, 28783, 29014, 29878, 29901, 30092, 30268, 30588, 30747, 30750, 31570, 31687, 31952, 32529, 32968, 33664, 34074, 34106, 34222, 34323, 34506, 35129, 35168, 35662, 36451, 36588, 36637, 38071, 38609, 38727, 38846, 38865, 38884, 39164, 39501, 39576, 39593, 40436, 41690, 41723, 43391, 44707, 45126, 45949, 49207, 50832, 52282, 52521, 55477, 56854, 59409, 62302, 62965, 64095, 64745, 67886, 71453, 71716, 71896, 71994, 72840, 77510, 78252, 80929, 85327, 87740, 88491, 92065, 93479, 95352, 110514, 145546, 147351, 149534, 155364, 163052, 163069, 167276, 173527, 173976, 175989, 176340, 176962, 178880, 179349, 179780, 180893, 184330, 186068, 186916, 209575, 217144, 217805, 246641, 248030, 257186, 335912, 341597, 352127, 396846]

4 We discover that there are 151 documents/rows with empty bag-of-words and/or bigrams, where their titles were filtered out and emptied during the preprocessing.

5 For example, at row with key = 1368, the title is in Arabic language which was filtered during the preprocessing step `remove_non_alphabetic()`.

6 What we will do now is removing those 151 rows from the dataset, and rebuild the dictionary and corpus

[8]:
```
# 1. First, remove those rows that will result in empty bags-of-words
df.drop(df.index[empty_docs], axis=0, inplace=True)
# reset the index
df.index = range(df.shape[0])
print('New dataframe size is ', df.shape[0])
```

```
New dataframe size is  509085
```

[9]:
```
# 2. Next, rebuild the dictionary, corpus, and the document-to-author mapping
dictionary, corpus, doc2author = build_dictioanry_and_corpus(df)

print('Number of authors: ', len(df.author.unique()))
print('Number of unique words: ', len(dictionary))
print('Number of documents: ', len(corpus))
print('Number of doc2author: ', len(doc2author))
```

```
Number of authors:  85745
Number of unique words:  100000
Number of documents:  509085
Number of doc2author:  509085
```

#

LDA model training

# 7 After cleaning and preprocessing our dataset, let's train an LDA model capable of discovering topics in the dataset

# 8 Model parameters:

### 8.1 1. *alpha='auto' and eta='auto'*, 'auto' enables automatic fine tuning the alpha and beta parameters of the LDA model from the dataset

### 8.2 2. *passes = 20*, is the number of passes through the corpus during training (the higher the better)

### 8.3 3. *iterations = 40*, is the maximum number of iterations through the corpus when inferring the topic distribution of a corpus. (the higher the better)

### 8.4 4. num_topics is the number of topics to be extracted from the training corpus

# 9 Since LDA requires the num_topics hyperparameter beforehand, we will search through different values of num_topics for the optimal number of topics that will result in the best topic coherence (check /https://www.aclweb.org/anthology/D12-1087.pdf for more details on topic coherence)

# 10 In order to speed-up the training process, we parallelize the training, where the model for each number-of-topics is trained on a separate core

[10]:
```
from gensim.models import CoherenceModel

model_list = []
passes = 20
iterations = 40

# this function filters the dataset by a set of kewords that exist in the␣
↪titles column
def filter_by_entity(df, keywords):
    keys_concat = '|'.join(keywords)
    p = re.compile(keys_concat, flags=re.IGNORECASE)
    new_df = df[[bool(p.search(x)) for x in df['title']]]
    return new_df

def train_model(num_topics):
```

```
    start = time.time()
    model = models.LdaModel(corpus=corpus, num_topics=num_topics, \
                            id2word=dictionary, \
                            passes=passes, iterations=iterations,\
                            alpha='auto', eta='auto')

    end = time.time()
    print(f'Runtime of model training = ', end - start, ', where num_topics =␣
↪', num_topics)

    return (model, num_topics)
```

# 11 Now lets filter all the titles related to China using a list of Chine related keywords `china_keywords = list(['china', 'chinese', 'shanghai', 'tibet'])`

# 12 The filtered data frame `filtered_df` contains all related entries to China based on the list above. This dataframe will be used to build new dictionary and corpus. and train LDA model

# 13 We will try different values for the `num_topics` hyperparameter of the LDA model, and for each value we will store the trained model into a list called `model_list`

[13]:
```
china_keywords = list(['china', 'chinese', 'shanghai', 'tibet'])

filtered_df = filter_by_entity(df, china_keywords)
dictionary, corpus, doc2author = build_dictioanry_and_corpus(filtered_df)

print('Topic modeling on China related articles:')
print('Number of authors: ', len(filtered_df.author.unique()))
print('Number of unique words: ', len(dictionary))
print('Number of documents: ', len(corpus))
print()
print('First 10 China related articles:')
print(filtered_df['title'].head(10))
print()

# num_topics
num_topics = list(range(2,51, 2))
pool = multiprocessing.Pool()
model_list = pool.map(train_model, num_topics)
```

```python
pool.close()
pool.join()
print('Finished!')
```

```
Topic modeling on China related articles:
Number of authors:  9476
Number of unique words:  39318
Number of documents:  31413

First 10 China related articles:
54
Hyperurbanization in China
122                                                          U.S. official, up
to 4 Chinese face spy charges - Security- msnbc.com
140                                                                  Russia,
China challenge US with proposal to ban space weapons
184
Russia, China Challenge US Space Arms
219                                                                  A $40
Million "Comedic Gold" Pissing Contest with the Chinese
307
Mix-Up Blamed for FDA Failure on China Heparin Plant
385    China asks nicely that Web sites cut out porn, violence, horror and all
backward, decadent thoughts.  Good luck with that.
415                                                                    China
wants online boycott of  decadent, backward thoughts
497
Rice Urges China to Use Influence on North Korea
557
US-India defence deal  to counter China  - Telegraph
Name: title, dtype: object

Runtime of model training =  226.33795928955078 , where num_topics =  8
Runtime of model training =  233.21868801116943 , where num_topics =  6
Runtime of model training =  237.51599287986755 , where num_topics =  12
Runtime of model training =  254.79378008842468 , where num_topics =  4
Runtime of model training =  255.31191968917847 , where num_topics =  10
Runtime of model training =  256.4270384311676 , where num_topics =  14
Runtime of model training =  269.99749398231506 , where num_topics =  18
Runtime of model training =  274.5581388473511 , where num_topics =  16
Runtime of model training =  279.3180994987488 , where num_topics =  20
Runtime of model training =  282.5347225666046 , where num_topics =  2
Runtime of model training =  291.38371229171753 , where num_topics =  28
Runtime of model training =  301.6735427379608 , where num_topics =  32
Runtime of model training =  301.7532317638397 , where num_topics =  24
Runtime of model training =  302.3709409236908 , where num_topics =  30
```

```
Runtime of model training =  303.7362277507782 , where num_topics =  22
Runtime of model training =  307.5811445713043 , where num_topics =  26
Runtime of model training =  240.2774817943573 , where num_topics =  34
Runtime of model training =  237.8790192604065 , where num_topics =  38
Runtime of model training =  246.22257161140442 , where num_topics =  36
Runtime of model training =  250.864328622818 , where num_topics =  40
Runtime of model training =  256.3949615955353 , where num_topics =  42
Runtime of model training =  259.3895752429962 , where num_topics =  44
Runtime of model training =  267.78936791419983 , where num_topics =  46
Runtime of model training =  272.73027658462524 , where num_topics =  48
Runtime of model training =  272.12229347229004 , where num_topics =  50
Finished!
```

## 14 Now lets calculate the c_v topics coherence score for each of the trained models

[14]:
```
# calculate topics coherence
def calculate_coherence(models):
    coherence_list = list()
    for (model, num_topics) in models:
        # Compute Coherence Score using c_v
        coherence_model = CoherenceModel(model=model,␣
 ↪texts=filtered_df['preprocessed_title'], \
                                             dictionary=dictionary,␣
 ↪coherence='c_v')
        coherence_score_cv = coherence_model.get_coherence()
        print('\nCoherence Score (c_v): ', coherence_score_cv, ' and num_topics␣
 ↪= ', num_topics)
        coherence_list.append((num_topics, coherence_score_cv))
    return coherence_list

coh_list = calculate_coherence(model_list)
print(coh_list)
```

```
Coherence Score (c_v):  0.3411195139370691  and num_topics =  2

Coherence Score (c_v):  0.41917542001924146  and num_topics =  4

Coherence Score (c_v):  0.3824197166131163  and num_topics =  6

Coherence Score (c_v):  0.3715623969880748  and num_topics =  8

Coherence Score (c_v):  0.3696895609378745  and num_topics =  10

Coherence Score (c_v):  0.38625148668212445  and num_topics =  12
```

```
Coherence Score (c_v):  0.3642281463397901  and num_topics =  14

Coherence Score (c_v):  0.3863931702545743  and num_topics =  16

Coherence Score (c_v):  0.3553958954022378  and num_topics =  18

Coherence Score (c_v):  0.3588115521364913  and num_topics =  20

Coherence Score (c_v):  0.36661506786223863  and num_topics =  22

Coherence Score (c_v):  0.3830928066804408  and num_topics =  24

Coherence Score (c_v):  0.34244625855063837  and num_topics =  26

Coherence Score (c_v):  0.3801590596618899  and num_topics =  28

Coherence Score (c_v):  0.37734781715828364  and num_topics =  30

Coherence Score (c_v):  0.3944970263134258  and num_topics =  32

Coherence Score (c_v):  0.3881869198280844  and num_topics =  34

Coherence Score (c_v):  0.4037678503274867  and num_topics =  36

Coherence Score (c_v):  0.39772652293984034  and num_topics =  38

Coherence Score (c_v):  0.4006394061977934  and num_topics =  40

Coherence Score (c_v):  0.4187329218216884  and num_topics =  42

Coherence Score (c_v):  0.4237049948098862  and num_topics =  44

Coherence Score (c_v):  0.4069376233405508  and num_topics =  46

Coherence Score (c_v):  0.38318755261501325  and num_topics =  48

Coherence Score (c_v):  0.44014468388615013  and num_topics =  50
[(2, 0.3411195139370691), (4, 0.41917542001924146), (6, 0.3824197166131163), (8,
0.3715623969880748), (10, 0.3696895609378745), (12, 0.38625148668212445), (14,
0.3642281463397901), (16, 0.3863931702545743), (18, 0.3553958954022378), (20,
0.3588115521364913), (22, 0.36661506786223863), (24, 0.3830928066804408), (26,
0.34244625855063837), (28, 0.3801590596618899), (30, 0.37734781715828364), (32,
0.3944970263134258), (34, 0.3881869198280844), (36, 0.4037678503274867), (38,
0.39772652293984034), (40, 0.4006394061977934), (42, 0.4187329218216884), (44,
0.4237049948098862), (46, 0.4069376233405508), (48, 0.38318755261501325), (50,
0.44014468388615013)]
```

# 15 Now let's find the model that achieved the best topic coherence and save it to disk for latter use

[17]:
```
best_num_topics, best_coherence = max(coh_list, key=lambda x: x[1])
best_model = None
for model, topics in model_list:
    if topics == best_num_topics:
        print(topics)
        best_model = model
        break

print('Best topics coherence: ', best_coherence)
print('Optimal number of topics = ', best_num_topics)

# Now print topics of best model
counter = 1
for topic in best_model.show_topics(num_topics=best_num_topics):
    words = ''
    for word, prob in model.show_topic(topic[0]):
        words += word + ' '
    print('Topic ', counter, ' words: ' + words)
    print()
    counter += 1


# Save the best model.
model_filename = 'china_top_model.atmodel'
best_model.save('./' + model_filename)
# Load model.
best_model = models.LdaModel.load('./' + model_filename)
```

```
50
Best topics coherence:  0.44014468388615013
Optimal number of topics =  50
Topic  1  words: military ahead expert huge even strategic street wall censor
question

Topic  2  words: country said minister corruption part record close case effort
prime

Topic  3  words: ban activist capital crisis debt probe meat food general
interest

Topic  4  words: protest dead rule pacific eastern visa live resident asia
spread

Topic  5  words: international jet fighter western american pledge chinese
```

medium steel bus

Topic 6 words: people attack death killed east three region xinjiang investment buy

Topic 7 words: internet satellite service search peace friday moon behind censorship chinese

Topic 8 words: end fishing chinese boat army land fisherman line offer turn

Topic 9 words: space time russian station defence post put sends trying officer

Topic 10 words: one billion bank shanghai child firm policy yuan currency led

Topic 11 words: reported canadian myanmar agree also based future tank organ slam

Topic 12 words: company dispute iran percent concern rise price arm find stealth

Topic 13 words: authority made home illegal despite chinese reach fake chinese_authority infrastructure

Topic 14 words: global oil hit month biggest gas massive nearly expected problem

Topic 15 words: china_sea see britain terror emission response order body along saudi

Topic 16 words: state united united_state nation security defense obama united_nation summit former

Topic 17 words: japan trade patrol data including fall business secret arrested used

Topic 18 words: high join tibetan canada africa level speed railway self within

Topic 19 words: ruling exercise blast export way west share large rare european

Topic 20 words: warns launch threat cut like territorial site import smog look

Topic 21 words: talk call chief energy tibet urge lama dalai plane dalai_lama

Topic 22 words: top take since start step action industry release decade accuses

Topic 23 words: day coal kill explosion mine senior hundred research ready every

Topic 24 words: open agreement according become arrest charge study bridge released ancient

Topic 25 words: coast factory vessel fight journalist give citizen ever try called

Topic 26 words: hong kong hong_kong australia national seek solar target row third

Topic 27 words: deal power help plant stock online nuclear taking anger network

Topic 28 words: city japanese navy zone meet putin member win ethnic chinese

Topic 29 words: india island could pakistan system face vow area troop bid

Topic 30 words: russia joint build drill move naval second europe hope premier

Topic 31 words: claim back hold found growth growing meeting risk thousand relation

Topic 32 words: use development tuesday collapse river heavy strong continue cruise said

Topic 33 words: government report plan medium may chinese asia project state_medium monday

Topic 34 words: right economic human show human_right sanction begin key great race

Topic 35 words: anti aircraft free carrier public political mission drug suspect pressure

Topic 36 words: amid crackdown old block social six thursday across access history

Topic 37 words: chinese new year official two million party last communist five

Topic 38 words: korea north north_korea missile nuclear south_korea korean week border tension

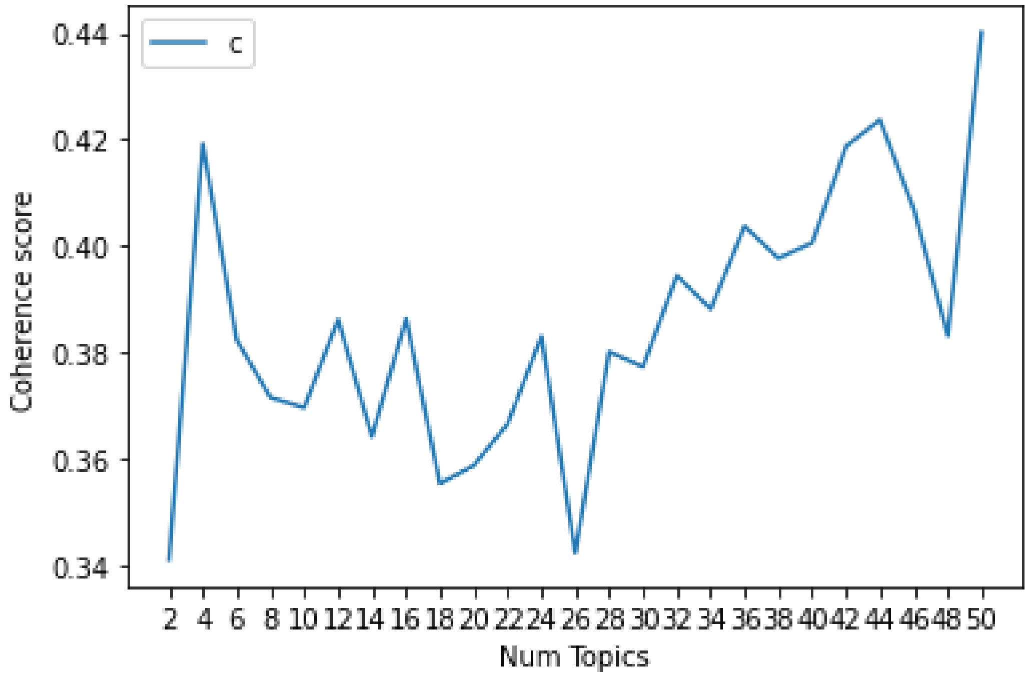
Topic 39 words: ship water chinese want near road australian four work student

```
Topic  40  words: first taiwan news law bbc bbc_news time tourist head
first_time

Topic  41  words: china south sea south_china say beijing war chinese_official
cyber new

Topic  42  words: get worker issue fear control eye stop warning come strike

Topic  43  words: president air visit pollution tell asian flight train another
air_pollution

Topic  44  words: leader man province test village family return vehicle
threatens supply

Topic  45  words: support tiananmen university environmental criticism square
saturday anniversary full production

Topic  46  words: world tie largest economy would indian boost demand
world_largest keep

Topic  47  words: police market force group chinese detained around southern
independence run

Topic  48  words: make woman climate court change sale life xinhua bird
climate_change

Topic  49  words: set fire theunited sign latest typhoon democracy germany
activity wednesday

Topic  50  words: foreign building missing ministry trial still almost chinese
diplomat terrorism
```

## 16 Let's visualize the topics coherence score vs the different numbers of topics

[18]:
```
import matplotlib
import matplotlib.pyplot as plt

num_topics = [topics[0] for topics in coh_list]
coh_scores = [scores[1] for scores in coh_list]
plt.plot(num_topics, coh_scores)
plt.xlabel("Num Topics")
plt.xticks(num_topics)
plt.ylabel("Coherence score")
```

```
plt.legend(("coherence_values"), loc='best')
plt.show()
```




# 17 Finally , we can visualize the topics mined by the best LDA model using pyLDAvis

[19]:
```
import pickle
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

# Visualize the topics
pyLDAvis.enable_notebook()
LDAvis_prepared = gensimvis.prepare(best_model, corpus, dictionary)
LDAvis_prepared
```

[19]:
```
PreparedData(topic_coordinates=              x         y  topics  cluster   Freq
topic
40     -1.17e-01 -4.07e-01       1         1  14.34
36     -3.30e-01  3.30e-02       2         1   5.38
15     -5.85e-02 -2.05e-01       3         1   3.81
32     -1.75e-01  3.15e-02       4         1   2.70
5      -8.40e-02 -2.62e-02       5         1   2.61
```

```
9     -3.95e-02 -5.35e-02       6        1   2.45
39    -5.85e-02 -1.14e-02       7        1   2.09
45     3.56e-02 -7.18e-02       8        1   2.06
1      4.17e-02 -9.64e-03       9        1   2.03
16     3.64e-02 -3.75e-02      10        1   1.97
28     4.39e-02 -5.66e-02      11        1   1.92
37     3.81e-02 -1.03e-01      12        1   1.92
38    -1.51e-01  6.91e-02      13        1   1.89
29     4.77e-02 -5.89e-02      14        1   1.84
46    -1.26e-01  6.85e-02      15        1   1.84
11    -1.59e-02  1.59e-02      16        1   1.82
48     5.50e-02 -2.43e-02      17        1   1.80
42     2.64e-03 -1.08e-02      18        1   1.75
26     6.05e-02 -4.63e-02      19        1   1.71
25    -4.94e-02 -3.68e-03      20        1   1.70
20     5.19e-02 -3.01e-02      21        1   1.68
41    -2.58e-02  3.94e-02      22        1   1.68
13     6.06e-02 -6.63e-03      23        1   1.65
47     3.46e-02 -6.30e-03      24        1   1.62
34     4.33e-02 -6.48e-03      25        1   1.62
33     6.71e-02 -2.17e-02      26        1   1.58
6     -4.73e-02  5.77e-02      27        1   1.58
0      5.60e-02 -2.69e-03      28        1   1.57
49    -5.28e-02  6.26e-02      29        1   1.52
2      5.01e-02  1.34e-02      30        1   1.52
7     -1.15e-01  7.75e-02      31        1   1.49
30     6.86e-02  1.29e-02      32        1   1.48
19     5.75e-02 -2.41e-03      33        1   1.46
35     5.60e-02  1.02e-02      34        1   1.46
17     6.70e-02  1.94e-02      35        1   1.44
12    -5.28e-02  8.02e-02      36        1   1.42
22     6.15e-02 -3.62e-04      37        1   1.42
4     -8.36e-02  7.79e-02      38        1   1.34
27    -2.24e-02  6.52e-02      39        1   1.33
18     7.09e-02  1.98e-02      40        1   1.30
23     7.65e-02  2.52e-02      41        1   1.29
8      4.04e-02  3.31e-02      42        1   1.28
21     7.82e-02  3.89e-02      43        1   1.27
3      3.01e-02  5.64e-02      44        1   1.26
24    -5.31e-02  8.83e-02      45        1   1.25
14     7.70e-02  2.37e-02      46        1   1.24
43     4.28e-02  4.20e-02      47        1   1.22
44     4.60e-02  5.92e-02      48        1   1.17
10     7.76e-02  3.71e-02      49        1   1.12
31     8.29e-02  4.39e-02      50        1   1.10, topic_info=              Term
Freq      Total Category  logprob  loglift
0            china  35297.00  35297.00  Default    30.00    30.00
```

```
3          chinese  10993.00  10993.00  Default    29.00    29.00
9            state   3679.00   3679.00  Default    28.00    28.00
10          united   3295.00   3295.00  Default    27.00    27.00
11    united_state   2859.00   2859.00  Default    26.00    26.00
…              …         …       …        …        …       …
3437    television     41.67     42.90  Topic50    -4.66     4.48
27          blamed     40.72     41.95  Topic50    -4.68     4.48
5027    plan_build     39.04     40.27  Topic50    -4.72     4.47
2813          able     36.46     37.69  Topic50    -4.79     4.47
959           said     84.68   1153.13  Topic50    -3.95     1.89

[1640 rows x 6 columns], token_table=      Topic  Freq    Term
term
2813      50  0.96    able
1808      30  0.98  abroad
3279      30  0.98   abuse
5300      35  0.96  accept
505       34  0.99  access
…     …    …      …
2876       4  0.98   young
5553       6  1.00    yuan
1539      33  0.98    zhou
1949      39  1.00    zone
1192      39  0.97     zoo

[1722 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1',
'ylab': 'PC2'}, topic_order=[41, 37, 16, 33, 6, 10, 40, 46, 2, 17, 29, 38, 39,
30, 47, 12, 49, 43, 27, 26, 21, 42, 14, 48, 35, 34, 7, 1, 50, 3, 8, 31, 20, 36,
18, 13, 23, 5, 28, 19, 24, 9, 22, 4, 25, 15, 44, 45, 11, 32])
```

#

Discussion of China related articles

## We found the best number of topics = 50, and as we can see from the visualization in the previous cell: ## 1. Topic 1 is about the tensions in the south China sea ## 2. Topic 2 is about statements from chinese officials ## 3. Topic 12 is about China's involvement in the North Korea and South Korea tensions ## 4. Topic 39 is about tensions between China, Russia, and Japan # Based on the overlap between a set of topics (e.g., topics 33, 37, 40, 43, 46, … etc), I believe that there is room for improvement to achieve more accurate clustering of articles into topics

[ ]:

18 Now lets filter all the titles related to Middle East using a list of related keywords `middle_east_keywords = list(['egypt', 'palestine', 'israel', 'iraq', 'iran', 'arab', 'saudi', 'gaza', 'turk'])`

19 The filtered data frame `filtered_df` contains all related entries to China based on the list above. This dataframe will be used to build new dictionary and corpus

20 We will try different values for the `num_topics` hyperparameter of the LDA model, and for each value we will store the trained model into `model_list`

[21]:
```
middle_east_keywords = list(['egypt', 'palestine', 'israel', 'iraq', 'iran',␣
↪'arab', \
                          'saudi', 'gaza', 'turk'])

filtered_df = filter_by_entity(df, middle_east_keywords)
dictionary, corpus, doc2author = build_dictioanry_and_corpus(filtered_df)

print('Topic modeling on China related articles:')
print('Number of authors: ', len(filtered_df.author.unique()))
print('Number of unique words: ', len(dictionary))
print('Number of documents: ', len(corpus))
print()
print('First 10 Middle East related articles:')
print(filtered_df['title'].head(10))
print()

# num_topics
num_topics = list(range(2,51, 2))
pool = multiprocessing.Pool()
model_list = pool.map(train_model, num_topics)

pool.close()
pool.join()
print('Finished!')
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
```

```
  and should_run_async(code)

Topic modeling on China related articles:
Number of authors:  18687
Number of unique words:  63361
Number of documents:  82675

First 10 Middle East related articles:
2                                                                              US
presses Egypt on Gaza border
15                      Nicolas Sarkozy, Angela Merkel confirm their opposition
to Turkey being EU membership
22                                                     Merkel to meet leaders of
Turkey, United Arab Emirates
42                                                               Six killed in
Israeli airstrike on Hamas base
48           Russia says an Iranian missile test this week raised suspicions
over its true nuclear ambitions.
49                                                                    US says
al-Qaida in Iraq using children
50    If Iran has 100% packet loss why does this website still load?  Also some
interesting stories on there.
55
Israel plans Egypt border fence
74                                                      U.S.-Backed Russian
Institutes Help Iran Build Reactor
85                                                                      Iran
starts second atomic power plant
Name: title, dtype: object

Runtime of model training =  425.42524003982544 , where num_topics =  4
Runtime of model training =  429.04105138778687 , where num_topics =  6
Runtime of model training =  445.7880780696869 , where num_topics =  8
Runtime of model training =  463.81572937965393 , where num_topics =  10
Runtime of model training =  466.74732756614685 , where num_topics =  12
Runtime of model training =  481.3924162387848 , where num_topics =  14
Runtime of model training =  489.4758355617523 , where num_topics =  2
Runtime of model training =  497.50837206840515 , where num_topics =  16
Runtime of model training =  520.74387550354 , where num_topics =  18
Runtime of model training =  546.96453166008 , where num_topics =  20
Runtime of model training =  565.7443571090698 , where num_topics =  22
Runtime of model training =  585.6039493083954 , where num_topics =  24
Runtime of model training =  603.6753454208374 , where num_topics =  26
Runtime of model training =  630.6157319545746 , where num_topics =  28
Runtime of model training =  657.178985118866 , where num_topics =  30
Runtime of model training =  680.2398056983948 , where num_topics =  32
Runtime of model training =  676.2042279243469 , where num_topics =  34
Runtime of model training =  701.1498019695282 , where num_topics =  36
```

```
Runtime of model training =  724.0148634910583 , where num_topics =  38
Runtime of model training =  744.0228261947632 , where num_topics =  40
Runtime of model training =  768.3842639923096 , where num_topics =  42
Runtime of model training =  789.7741312980652 , where num_topics =  44
Runtime of model training =  811.5759119987488 , where num_topics =  46
Runtime of model training =  844.1068692207336 , where num_topics =  48
Runtime of model training =  866.8519554138184 , where num_topics =  50
Finished!
```

# 21 Now lets calculate the c_v topics coherence score for each of the trained models

[22]:
```
# calculate topics coherence
def calculate_coherence(models):
    coherence_list = list()
    for (model, num_topics) in models:
        # Compute Coherence Score using c_v
        coherence_model = CoherenceModel(model=model,␣
 ↪texts=filtered_df['preprocessed_title'], \
                                             dictionary=dictionary,␣
 ↪coherence='c_v')
        coherence_score_cv = coherence_model.get_coherence()
        print('\nCoherence Score (c_v): ', coherence_score_cv, ' and num_topics␣
 ↪= ', num_topics)
        coherence_list.append((num_topics, coherence_score_cv))
    return coherence_list

coh_list = calculate_coherence(model_list)
print(coh_list)
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)


Coherence Score (c_v):  0.26450601940219287  and num_topics =  2

Coherence Score (c_v):  0.39523294020773014  and num_topics =  4

Coherence Score (c_v):  0.4121413186991372  and num_topics =  6

Coherence Score (c_v):  0.3652314371508326  and num_topics =  8
```

```
Coherence Score (c_v):  0.3761327410817218  and num_topics =  10

Coherence Score (c_v):  0.3452110343840544  and num_topics =  12

Coherence Score (c_v):  0.3319156292146161  and num_topics =  14

Coherence Score (c_v):  0.3283303580728115  and num_topics =  16

Coherence Score (c_v):  0.3272164236240815  and num_topics =  18

Coherence Score (c_v):  0.3266125643806592  and num_topics =  20

Coherence Score (c_v):  0.3153536672997266  and num_topics =  22

Coherence Score (c_v):  0.3113689746554735  and num_topics =  24

Coherence Score (c_v):  0.31167647185523467  and num_topics =  26

Coherence Score (c_v):  0.316386201381872  and num_topics =  28

Coherence Score (c_v):  0.30889067781188634  and num_topics =  30

Coherence Score (c_v):  0.31463505487772664  and num_topics =  32

Coherence Score (c_v):  0.3208118036648004  and num_topics =  34

Coherence Score (c_v):  0.314205152821205  and num_topics =  36

Coherence Score (c_v):  0.335433212487173  and num_topics =  38

Coherence Score (c_v):  0.36062450380157224  and num_topics =  40

Coherence Score (c_v):  0.33900091325047965  and num_topics =  42

Coherence Score (c_v):  0.3451669808515861  and num_topics =  44

Coherence Score (c_v):  0.3608552901886936  and num_topics =  46

Coherence Score (c_v):  0.3605718223899744  and num_topics =  48

Coherence Score (c_v):  0.3706433930605189  and num_topics =  50
[(2, 0.26450601940219287), (4, 0.39523294020773014), (6, 0.4121413186991372),
(8, 0.3652314371508326), (10, 0.3761327410817218), (12, 0.3452110343840544),
(14, 0.3319156292146161), (16, 0.3283303580728115), (18, 0.3272164236240815),
(20, 0.3266125643806592), (22, 0.3153536672997266), (24, 0.3113689746554735),
(26, 0.31167647185523467), (28, 0.316386201381872), (30, 0.30889067781188634),
(32, 0.31463505487772664), (34, 0.3208118036648004), (36, 0.314205152821205),
(38, 0.335433212487173), (40, 0.36062450380157224), (42, 0.33900091325047965),
```

```
(44, 0.3451669808515861), (46, 0.3608552901886936), (48, 0.3605718223899744),
(50, 0.3706433930605189)]
```

# 22 Now let's find the model that achieved the best topic coherence and save it to disk for latter use

[23]:
```
best_num_topics, best_coherence = max(coh_list, key=lambda x: x[1])
best_model = None
for model, topics in model_list:
    if topics == best_num_topics:
        print(topics)
        best_model = model
        break

print('Best topics coherence: ', best_coherence)
print('Optimal number of topics = ', best_num_topics)

# Now print topics of best model
counter = 1
for topic in best_model.show_topics(num_topics=8):
    words = ''
    for word, prob in model.show_topic(topic[0]):
        words += word + ' '
    print('Topic ', counter, ' words: ' + words)
    counter += 1


# Save the best model.
model_filename = 'middle_east_top_model_' + str(best_num_topics) + '.atmodel'
best_model.save('./' + model_filename)
# Load model.
best_model = models.LdaModel.load('./' + model_filename)
```

```
6
Best topics coherence:  0.4121413186991372
Optimal number of topics =  6
Topic  1  words: israeli palestinian gaza police yemen coup year right soldier
woman
Topic  2  words: egypt president erdogan egyptian air death court journalist
muslim top
Topic  3  words: turkey state israel united say turkish united_state syria
iranian military
Topic  4  words: iraq iraqi isi attack force islamic killed mosul kurdish kill
Topic  5  words: minister syrian israel west bank plan west_bank town settlement
home
Topic  6  words: iran saudi arabia saudi_arabia deal nuclear russia oil world
```

```
netanyahu
```

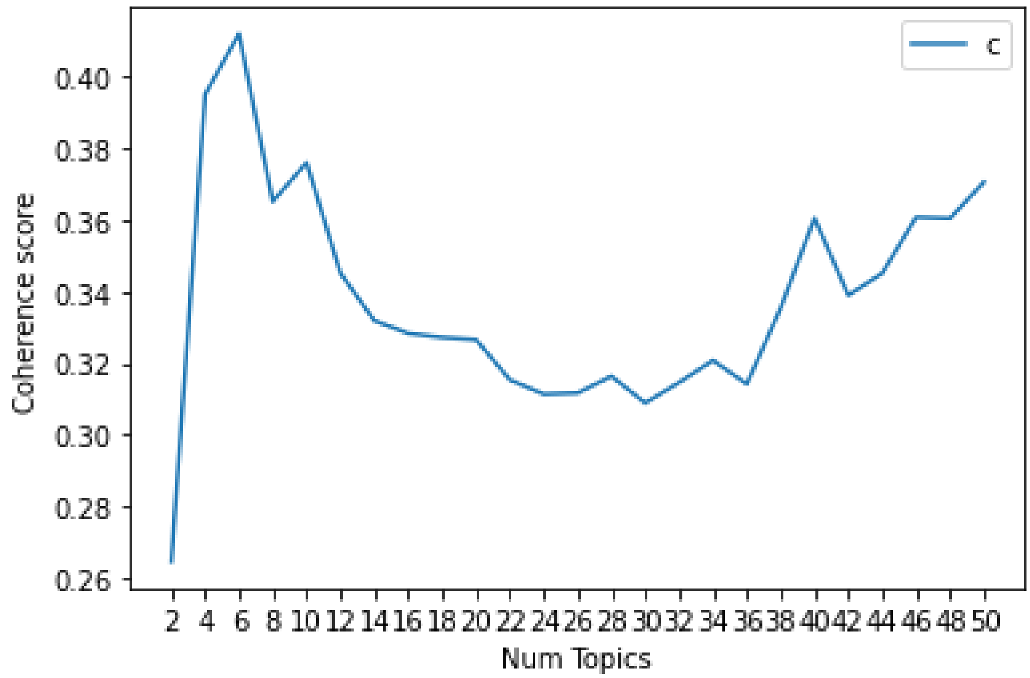
```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

# 23 Let's visualize the topics coherence score vs the different numbers of topics

[24]:

```python
import matplotlib
import matplotlib.pyplot as plt

num_topics = [topics[0] for topics in coh_list]
coh_scores = [scores[1] for scores in coh_list]
plt.plot(num_topics, coh_scores)
plt.xlabel("Num Topics")
plt.xticks(num_topics)
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```




# 24 Finally , we can visualize the topics mined by the best LDA model using pyLDAvis

[25]:
```
import pickle
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

# Visualize the topics
pyLDAvis.enable_notebook()
LDAvis_prepared = gensimvis.prepare(best_model, corpus, dictionary)
LDAvis_prepared
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[25]:
```
PreparedData(topic_coordinates=          x         y  topics  cluster   Freq
topic
2       0.40  1.13e-02       1        1  37.01
```

```
0     -0.02  1.13e-03       2        1  14.95
3     -0.11  3.64e-01       3        1  14.49
5     -0.13 -2.26e-01       4        1  13.64
1     -0.11 -3.89e-02       5        1  10.05
4     -0.02 -1.11e-01       6        1   9.87, topic_info=             Term
Freq     Total Category  logprob  loglift
51           iran  12352.00  12352.00  Default    30.00    30.00
251         saudi  10589.00  10589.00  Default    29.00    29.00
26        israeli   8432.00   8432.00  Default    28.00    28.00
17         turkey  16058.00  16058.00  Default    27.00    27.00
45           iraq   7152.00   7152.00  Default    26.00    26.00
…           …        …       …       …       …       …
473           vow    412.29    413.14   Topic6    -5.49     2.31
363          must    403.06    403.91   Topic6    -5.51     2.31
2727  cooperation    397.66    398.51   Topic6    -5.52     2.31
60         israel   1764.92  11925.75   Topic6    -4.03     0.40
153          news    658.63   1721.06   Topic6    -5.02     1.35

[233 rows x 6 columns], token_table=      Topic  Freq        Term
term
2044      2  1.00    activist
1410      5  1.00         air
3962      5  1.00  air_strike
4985      5  1.00     airport
626       5  1.00  airstrikes
…     …   …         …
344       2  1.00       woman
235       4  1.00       world
988       1  0.36        year
988       2  0.64        year
2039      2  1.00       yemen

[228 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1',
'ylab': 'PC2'}, topic_order=[3, 1, 4, 6, 2, 5])
```

#

Discussion of Middle East related articles

## We achieved the best topics coherence with `num_topics = 6`, and as we can see from the visualization in the previous cell: ## 1. Topic 1 is a big mixed topic, which is mainly about ISIS, U.S.A., Iraq, Iran, Israel and Turkey ## 2. Topic 2 is about the Israeli-Palestinian conflict ## 3. Topic 3 is about Iraq related news ## 4. Topic 4 is about Iran, its nuclear program, and its tension with neighbouring countries (e.g. Saudia Arabia) ## 5. Topic 5 is about Turkey, and its political tensions with Egypt ## 6. Topic 6 is about Israel, its prime minister, and its tensions with Palestine and neibouring countries (e.g., Syria and Iran)

#

Answering the second question

#

Given a title T, what are the most similar articles to T?

### 24.1 First, we compute the similarity matrix to query the model and find the most related articles to a given article

[26]:
```
index = similarities.MatrixSimilarity(best_model[corpus])
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

## 25 First, we list the first 10 articles related to Middle East

## 26 Next, we select one of the shown articles to query its similar articles

## 27 `print_related_documents` takes `article_id` of article T, and shows the `top_k` most related articles to T

[27]:
```
print('First 10 Middle East related articles:')
print(filtered_df['title'].head(10))

def print_related_documents(article_id, top_k):
    vec_lda = best_model[corpus[article_id]]
    print()
    sims = index[vec_lda]
    sims = sorted(enumerate(sims), key=lambda item: -item[1])[0:top_k]
    for doc_position, doc_score in sims:
        print(doc_score, filtered_df['title'].iloc[doc_position])
    return

article_id = 4
top_k = 10

article_title = filtered_df['title'].iloc[article_id]
article_vector = filtered_df['preprocessed_title'].iloc[article_id]
print()
```

```python
print('The', top_k, 'most related articles to: [', article_title, '] are:')
print_related_documents(article_id, top_k)
```

```
First 10 Middle East related articles:
2                                                                             US
presses Egypt on Gaza border
15                      Nicolas Sarkozy, Angela Merkel confirm their opposition
to Turkey being EU membership
22                                                     Merkel to meet leaders of
Turkey, United Arab Emirates
42                                                              Six killed in
Israeli airstrike on Hamas base
48          Russia says an Iranian missile test this week raised suspicions
over its true nuclear ambitions.
49                                                                    US says
al-Qaida in Iraq using children
50    If Iran has 100% packet loss why does this website still load?  Also some
interesting stories on there.
55
Israel plans Egypt border fence
74                                                      U.S.-Backed Russian
Institutes Help Iran Build Reactor
85                                                                      Iran
starts second atomic power plant
Name: title, dtype: object

The 10 most related articles to: [ Russia says an Iranian missile test this week
raised suspicions over its true nuclear ambitions. ] are:

1.0 Russia says an Iranian missile test this week raised suspicions over its
true nuclear ambitions.
0.9997606 The head of the UN nuclear watchdog has said it is close to signing an
accord with Iran after his talks in Tehran
0.9991644 Gaza crisis: Israel claims progress as UN chief set to visit Middle
East
0.99868786 Turkey, EU Seal $3.2 Billion Deal to Stem Migrant Flow - World News -
Israel News - Haaretz Israeli News Source
0.9983866 Would the world blame Israel if Iranian nuclear talks fail?
0.9983326 Iran hints might not reject 10-year partial freeze of nuclear work
0.9983164 Officials: Iran nuke talks to continue in new phase
0.99814856 Iran granted special exemptions over nuclear deal behind close doors.
0.99814516 WikiLeaks: Syria aimed chemical weapons at Israel after Syria s
secret North Korean nuclear plant was destroyed by Israel. Hmmm
0.9979881 Saudi Arabia’s Sabic Considering Shale Gas Investments in U.S.
```

```
/home/ehussein/anaconda3/envs/my_env/lib/python3.9/site-
packages/ipykernel/ipkernel.py:287: DeprecationWarning: `should_run_async` will
not call `transform_cell` automatically in the future. Please pass the result to
```

```
`transformed_cell` argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[ ]:

```

```