In [1]:
```
# !pip install gensim==3.8.3
# !pip install pyLDAvis==3.3.1
```

In [2]:
```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from IPython.display import HTML, display
import tabulate
import pandas as pd
import numpy as np
from PIL import Image
from wordcloud import WordCloud
import seaborn as sns
from gensim.models.coherencemodel import CoherenceModel
import pyLDAvis

# from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
# from nltk.stem.porter import *

import gensim
from gensim.models import Phrases
#Prepare objects for LDA gensim implementation
from gensim import corpora
#Running LDA
from gensim import models

import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
import re
%matplotlib inline
```

```
C:\Users\devas\anaconda3\lib\site-packages\gensim\similarities\__init__.py:15: UserWarning: The gensim.similarities.levenshtein submodule is disabled, because the optional L
evenshtein package <https://pypi.org/project/python-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
```

In [3]:
```
data=pd.read_csv('C:/Users/devas/Documents/Pattern/omicron_tweets.csv', engine='python')
# dropping empty rows
data = data.dropna(subset=['text'])
# dropping duplicates
data = data.drop_duplicates()
print('Rows: {}, columns: {}'.format(data.shape[0], data.shape[1]))
data.head(2)
```

```
Rows: 8066, columns: 13
```

Out[3]:

	tweet_id	date	text	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	hashtags	source	is_retweet
0	1465162850457071621	2021-11-29 03:36:59+00:00	Will Boris Johnson ever learn his lesson after...	James robertson	NaN	NaN	2013-04-22 19:41:30+00:00	303.0	188	84059	NaN	Twitter for Android	False
1	1465162841665769474	2021-11-29 03:36:57+00:00	#Omicron so hot; it's being rush shipped in De...	Gene Bulmer	East Coast	#ChristFollower, #Leader, Husband, Father, #Ru...	2014-11-10 22:43:27+00:00	99.0	162	745	['Omicron']	Twitter for iPhone	False

Data Manipulation

In [4]:

```
data['processed'] = ''
nltk.download('wordnet')
nltk.download('stopwords')

stop_words = stopwords.words('english')

stop_words.extend(['https'])
# initalizing the werdnet lemmatizer
lm = WordNetLemmatizer()

def processing(content):

    content = content.replace('\n', ' ').split(' ')
#  removing these punctuations from tokens like it will convert the word mode? into mode
    rx = re.compile('([&#.:?!-()])*')
    content = [rx.sub('', word) for word in content]

#  removing stopwords
    content = [word.strip().lower() for word in content if word.strip().lower() not in stop_words]
  #  remove words whose length is greater than 1 and or alphabetics only
    content = [word for word in content if len(word)>1 and word.isalpha()]
    # lemmatizing the words to their basic form
    content = [lm.lemmatize(word) for word in content]

    return ' '.join(content)

for k in range(len(data)):
  data.iloc[k,-1] = processing(data.iloc[k,2])
# processed data
data.head()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\devas\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\devas\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

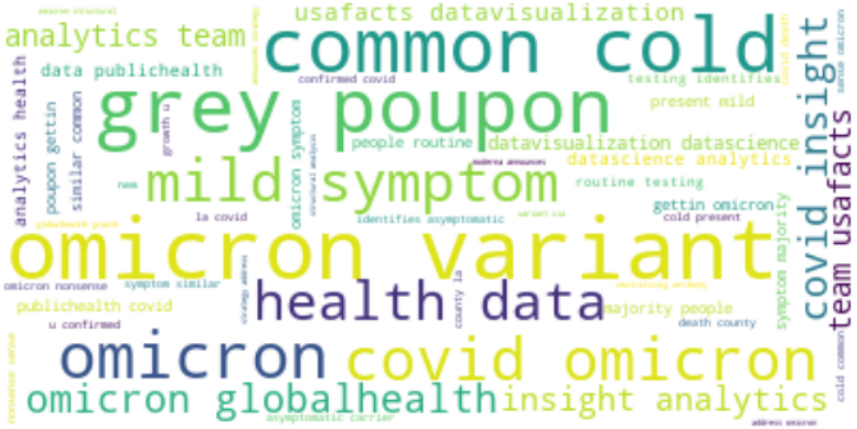
Out[4]:

	tweet_id	date	text	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	hashtags	source	is_retweet
0	1465162850457071621	2021-11-29 03:36:59+00:00	Will Boris Johnson ever learn his lesson after...	James robertson	NaN	NaN	2013-04-22 19:41:30+00:00	303.0	188	84059	NaN	Twitter for Android	False
1	1465162841665769474	2021-11-29 03:36:57+00:00	#Omicron so hot; it's being rush shipped in De...	Gene Bulmer	East Coast	#ChristFollower, #Leader, Husband, Father, #Ru...	2014-11-10 22:43:27+00:00	99.0	162	745	['Omicron']	Twitter for iPhone	False
2	1465162837047726094	2021-11-29 03:36:56+00:00	@JoeBiden How To Stop The Spread of Coronaviru...	Andrew Arcie Galendez (ᖴᔕᗰᖇᑭᔕ)	NaN	I AM PURE FILIPINO; BOTH MY PARENTS ARE FILIPI...	2021-01-28 11:31:59+00:00	114.0	3223	4830	NaN	Twitter for Android	False
3	1465162836498432005	2021-11-29 03:36:56+00:00	Gold Coast leaders urge against overcautious r...	myGC.com.au	Gold Coast, Australia	The Gold Coast's best news, local events, weat...	2009-03-30 03:54:34+00:00	12232.0	3191	271	['GoldCoastNews']	Zapier.com	False
4	1465162831444119562	2021-11-29 03:36:55+00:00	What Is Omicron Radiation...\nhttps://t.co/QfK...	Ghost Of Gus Hall	Poughkeepsie Peoples Republic	Freelancer who knows a name and is lying to po...	2011-10-17 03:25:39+00:00	603.0	960	819	NaN	Twitter for Android	False

Word Cloud

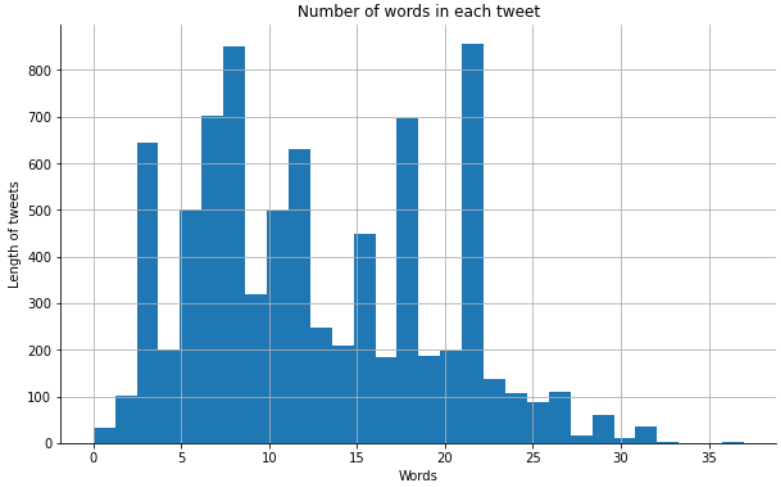
In [5]:

```
b = data['processed'].tolist()
b = ' '.join(map(str, b))
# b = b.replace(',', ' ').lower().replace('-', '')

wordcloud = WordCloud(max_font_size=40, max_words=1000, background_color="white", random_state=100,
                      prefer_horizontal=0.60).generate(b.lower())
# max_font_size=40, max_words=1000, background_color="white",
                      # random_state=100, prefer_horizontal=0.50
plt.figure(figsize=(12,8))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
# plt.show()
plt.savefig('word_cloud.jpg')
```



In [6]:

```
lengths = [len(sentence.split(' ')) if len(sentence)>0 else 0 for sentence in data['processed']]
data['Lengths'] = lengths
plt.figure(figsize=(10,6))
data['Lengths'].hist(bins=30)
sns.despine(top=True, right=True, left=False, bottom=False)
plt.title('Number of words in each tweet')
plt.xlabel('Words')
plt.ylabel('Length of tweets')
plt.savefig('length_tweets.jpg')
plt.show()
```



In [7]:

```
import gensim.corpora as corpora

#decomposing sentences into tokens
tokens = [sentence.split(' ') for sentence in data['processed'] ]
# training a bi gram model in order to include those bigrams as tokens who occured at least 6 times
# in the whole dataset
bigram = gensim.models.Phrases(tokens, min_count=2, threshold=100)
bigram_mod = gensim.models.phrases.Phraser(bigram)

# including bigrams as tokens
sents = [ bigram_mod[token] for token in tokens]

# Create Dictionary to keep track of vocab
dct = corpora.Dictionary(tokens)

print('Unique words before filtering/after pre-processing', len(dct))
# no_below= 30
# filter the words that occure in less than 3 documents and in more the 60% of documents
dct.filter_extremes(no_below= 3, no_above=0.60 )
print('Unique words after filtering', len(dct))

# Create Corpus
corpus = [dct.doc2bow(sent) for sent in sents]

tfidf = gensim.models.TfidfModel(corpus)
corpus_tfidf = tfidf[corpus]
```

```
Unique words before filtering/after pre-processing 9834
Unique words after filtering 3263
```

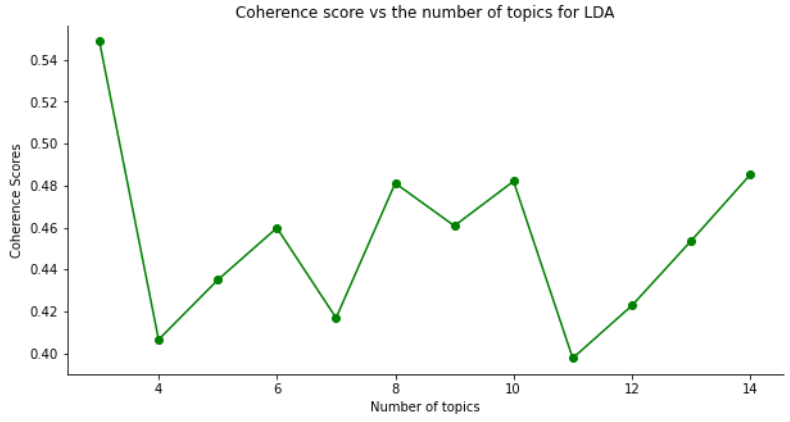
In [8]:

```
%%time
from gensim.models import CoherenceModel
import time
import os

scores = []
for k in range(3,15):
    # LDA model
    lda_model = gensim.models.LdaModel(  corpus=corpus_tfidf, num_topics=k,
                                         id2word=dct, random_state=12)
    # to calculate score for coherence
    coherence_model_lda = CoherenceModel(model=lda_model, texts=sents, dictionary=dct, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    print(k, coherence_lda)
    scores.append(coherence_lda)
```

```
3 0.5488264668187827
4 0.4065263368076506
5 0.43498981193412023
6 0.4598313128230022
7 0.41688548620270843
8 0.48115228053009207
9 0.4608086515564191
10 0.4820849819803641
11 0.3976712431810696
12 0.4227588296332713
13 0.45360036270778586
14 0.48519719309739046
Wall time: 1min 17s
```

In [9]:

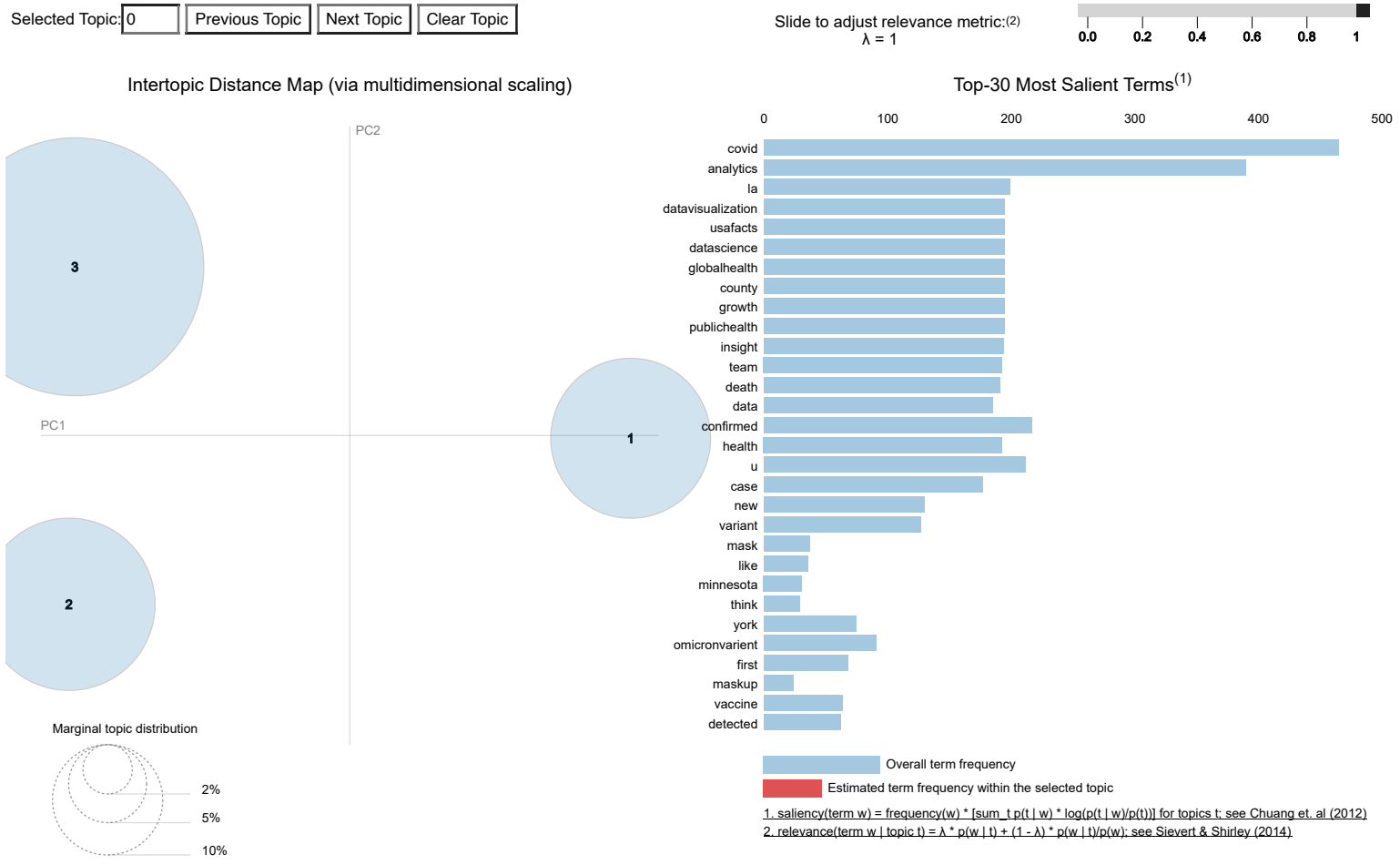
```
selected_topics = np.argmax(scores)+3

plt.figure(figsize=(10, 5))
plt.plot(list(range(3,15)), scores, marker='o', color='green')
sns.despine(top=True, right=True, left=False, bottom=False)

plt.locator_params(integer=True)
plt.title('Coherence score vs the number of topics for LDA')
plt.xlabel('Number of topics')
plt.ylabel('Coherence Scores')
plt.savefig('lda_scores.jpg')
plt.show()
```



In [10]:

```
import pyLDAvis.gensim_models

lda_model = gensim.models.LdaModel(corpus=corpus_tfidf, id2word=dct, num_topics=selected_topics,
                                   random_state=12, chunksize=128, passes=10 )

pyLDAvis.enable_notebook()
results = pyLDAvis.gensim_models.prepare(lda_model, corpus_tfidf, dct, sort_topics=False)
pyLDAvis.save_html(results, 'ldavis_english' +'.html')
results
```

Out[10]:



In [11]:
```
# top words in each topic
lda_model.print_topics()
```

Out[11]:
```
[(0,
  '0.106*"covid" + 0.095*"analytics" + 0.049*"la" + 0.048*"usafacts" + 0.048*"datascience" + 0.048*"datavisualization" + 0.048*"globalhealth" + 0.048*"county" + 0.048*"growt
h" + 0.048*"publichealth"'),
 (1,
  '0.008*"mask" + 0.008*"like" + 0.008*"omicronvarient" + 0.006*"minnesota" + 0.006*"omicronvariant" + 0.006*"think" + 0.005*"maskup" + 0.005*"anyone" + 0.004*"auspol" + 0.0
04*"biden"'),
 (2,
  '0.017*"case" + 0.012*"new" + 0.012*"variant" + 0.007*"york" + 0.006*"first" + 0.006*"vaccine" + 0.006*"state" + 0.006*"detected" + 0.006*"coronavirus" + 0.005*"omicronvar
ient"')]
```

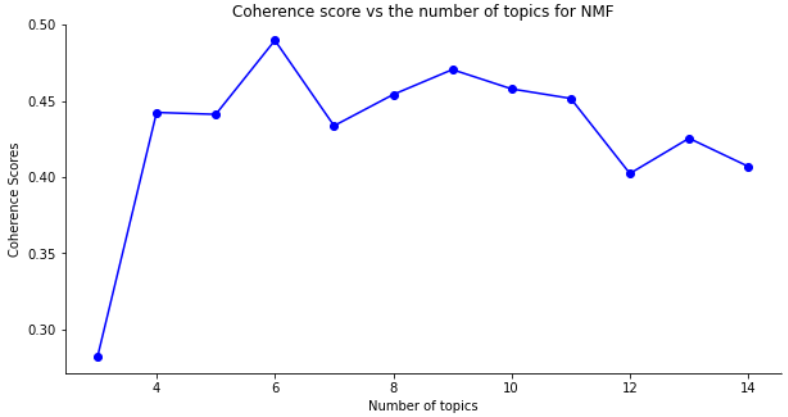
Non-Negative Matrix Factorization (NMF)

In [12]:
```
from gensim.models.nmf import Nmf

scores_nmf = []
for k in range(3,15):
    # lda mallet model
    nmf_model = Nmf(corpus_tfidf, num_topics=k, \
                                  id2word=dct, \
                                  passes=10)
    # to calculate score for coherence
    coherence_model_lda = CoherenceModel(model=nmf_model, texts=sents, dictionary=dct, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    print(k, coherence_lda)
    scores_nmf.append(coherence_lda)
```

```
3 0.28168517274532157
4 0.442362387904111175
5 0.4411030317965917
6 0.4897078960989399
7 0.43361447763211913
8 0.45412241563139766
9 0.4705144598195836
10 0.45781569456240473
11 0.45155744281011023
12 0.40227751202378337
13 0.4254008453564542
14 0.40692994154704376
```

In [13]:

```
plt.figure(figsize=(10, 5))
plt.plot(list(range(3,15)), scores_nmf, marker='o', color='blue')
sns.despine(top=True, right=True, left=False, bottom=False)

plt.locator_params(integer=True)
plt.title('Coherence score vs the number of topics for NMF')
plt.xlabel('Number of topics')
plt.ylabel('Coherence Scores')
plt.savefig('lda_scores.jpg')
plt.show()
```



In [14]:
```
selected_topics_nmf=np.argmax(scores_nmf)+3
nmf_model = Nmf(corpus=corpus_tfidf, id2word=dct, num_topics=selected_topics_nmf,
                                          random_state=12, chunksize=128, passes=10 )

nmf_model.print_topics()
```

Out[14]:
```
[(0,
  '0.285*"poupon" + 0.285*"grey" + 0.247*"like" + 0.142*"gettin" + 0.002*"sound" + 0.001*"transformer" + 0.001*"name" + 0.001*"somewhat" + 0.001*"movie" + 0.001*"look"'),
 (1,
  '0.123*"case" + 0.044*"first" + 0.041*"two" + 0.030*"detected" + 0.025*"variant" + 0.019*"report" + 0.015*"canada" + 0.013*"york" + 0.013*"new" + 0.012*"confirmed"'),
 (2,
  '0.023*"variant" + 0.017*"new" + 0.010*"travel" + 0.010*"world" + 0.010*"omicronvariant" + 0.010*"coronavirus" + 0.009*"country" + 0.009*"vaccine" + 0.008*"know" + 0.007
*"say"'),
 (3,
  '0.153*"cold" + 0.152*"common" + 0.144*"symptom" + 0.077*"routine" + 0.077*"majority" + 0.076*"similar" + 0.076*"present" + 0.073*"testing" + 0.073*"mild" + 0.066*"peopl
e"'),
 (4,
  '0.114*"covid" + 0.101*"analytics" + 0.051*"usafacts" + 0.051*"datavisualization" + 0.051*"datascience" + 0.050*"globalhealth" + 0.050*"growth" + 0.050*"publichealth" + 0.
050*"insight" + 0.050*"county"'),
 (5,
  '0.297*"moderna" + 0.153*"candidate" + 0.153*"address" + 0.138*"booster" + 0.137*"via" + 0.077*"variant" + 0.003*"say" + 0.001*"pfizer" + 0.001*"vaccine" + 0.001*"patien
t"')]
```

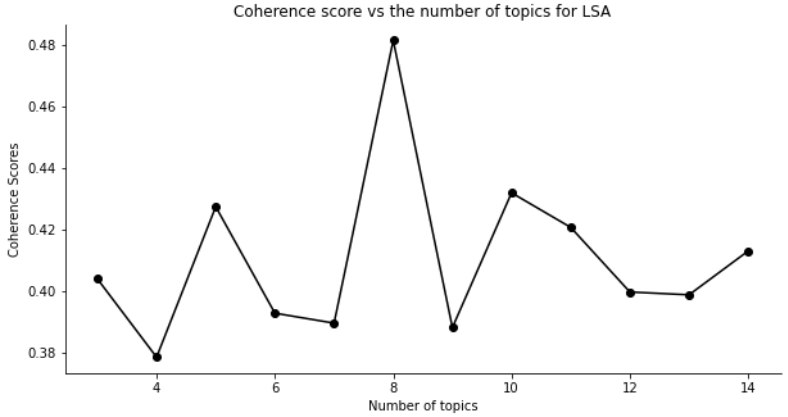
Latent Samentic Analysis

In [15]:
```
from gensim.models.lsimodel import LsiModel

scores_lsi = []
for k in range(3,15):
    # LSI model
    lsi_model = LsiModel(  corpus=corpus_tfidf, num_topics=k,power_iters=250,
                                                id2word=dct)
    # to calculate score for coherence
    coherence_model_lsi = CoherenceModel(model=lsi_model, texts=sents, dictionary=dct, coherence='c_v')
    coherence_lsi = coherence_model_lsi.get_coherence()
    print(k, coherence_lsi)
    scores_lsi.append(coherence_lsi)
```

```
3 0.4039538049985449
4 0.3784976981923267
5 0.42744837035054306
6 0.3928130838248274
7 0.3895709231074262
8 0.48158829107852097
9 0.3881411519633174
10 0.43196943563017276
11 0.4207294653554794
12 0.3997005384465795
13 0.3987879310335635
14 0.41305537942116277
```

In [16]:

```
plt.figure(figsize=(10, 5))
plt.plot(list(range(3,15)), scores_lsi, marker='o', color='black')
sns.despine(top=True, right=True, left=False, bottom=False)

plt.locator_params(integer=True)
plt.title('Coherence score vs the number of topics for LSA')
plt.xlabel('Number of topics')
plt.ylabel('Coherence Scores')
plt.savefig('lda_scores.jpg')
plt.show()
```



In [17]:
```
selected_topics_lsi = np.nanargmax(scores_lsi)+3
lsi_model = LsiModel(  corpus=corpus_tfidf, num_topics=selected_topics_lsi,
                                          id2word=dct)
lsi_model.print_topics()
```

Out[17]:
```
[(0,
  '-0.577*"poupon" + -0.577*"grey" + -0.500*"like" + -0.289*"gettin" + -0.011*"cold" + -0.011*"common" + -0.010*"symptom" + -0.006*"covid" + -0.005*"routine" + -0.005*"major
ity"'),
 (1,
  '-0.469*"cold" + -0.468*"common" + -0.444*"symptom" + -0.236*"routine" + -0.235*"majority" + -0.235*"similar" + -0.232*"present" + -0.225*"testing" + -0.224*"mild" + -0.20
4*"people"'),
 (2,
  '-0.473*"covid" + -0.418*"analytics" + -0.209*"usafacts" + -0.209*"datavisualization" + -0.209*"datascience" + -0.209*"globalhealth" + -0.208*"growth" + -0.208*"publicheal
th" + -0.208*"insight" + -0.207*"county"'),
 (3,
  '0.700*"moderna" + 0.360*"candidate" + 0.360*"address" + 0.327*"booster" + 0.325*"via" + 0.194*"variant" + 0.008*"case" + 0.007*"new" + 0.006*"say" + 0.006*"vaccine"'),
 (4,
  '-0.492*"case" + -0.398*"variant" + -0.307*"new" + -0.198*"first" + -0.170*"coronavirus" + -0.167*"two" + -0.165*"detected" + -0.146*"omicronvariant" + -0.136*"say" + -0.1
23*"travel"'),
 (5,
  '-0.679*"eth" + -0.362*"think" + -0.360*"mask" + -0.349*"luck" + -0.335*"dont" + 0.122*"case" + -0.081*"omicronvariant" + -0.047*"know" + 0.043*"first" + 0.040*"two"'),
 (6,
  '-0.543*"omicronvariant" + 0.494*"case" + -0.396*"cake" + 0.156*"eth" + 0.147*"first" + 0.146*"two" + -0.133*"vaccine" + -0.109*"know" + -0.095*"travel" + -0.092*"new"'),
 (7,
  '-0.537*"omicronvariant" + -0.445*"cake" + -0.379*"case" + 0.207*"new" + 0.191*"variant" + 0.154*"travel" + 0.143*"know" + -0.140*"first" + 0.129*"vaccine" + -0.119*"tw
o"')]
```

Best Model Analysis

On the basis of coherence scores lda model have been chosen as the best model for the current data set

In [18]:
```
# judged manually from pyldavis
topics_name = ['Topic 1', 'Topic 2', 'Topic 3']

# getting the most dominant topics from a trained model
predicted_topics = lda_model[corpus_tfidf]

probs, topics = [], []
for k in predicted_topics:
  # sorting the probabilites
  k.sort(key=lambda x:x[1])
  # selecting the topic with greates probability
  topics.append(topics_name[ k[0][0] ] )

data['Topics'] = topics
data.head(2)
```

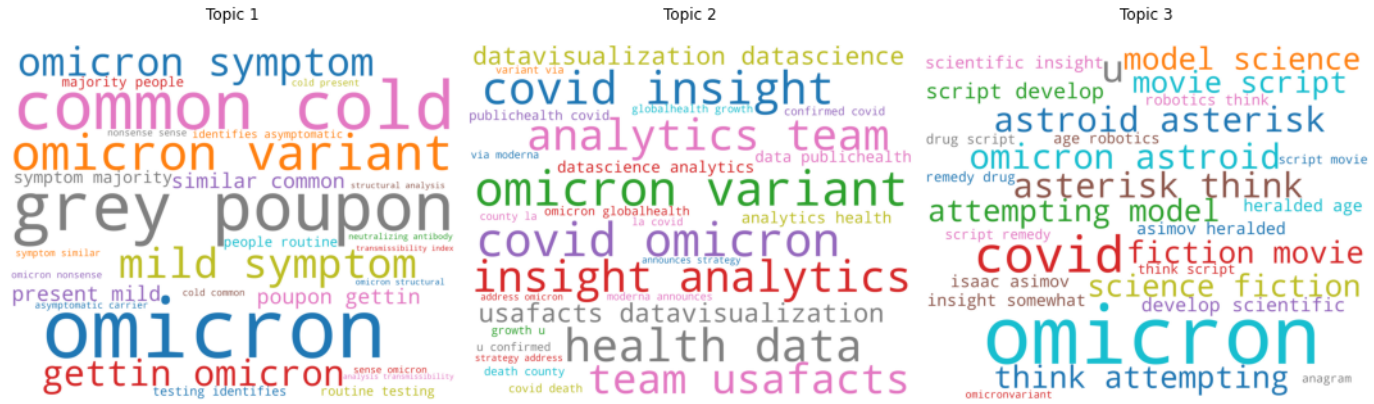
Out[18]:

	tweet_id	date	text	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	hashtags	source	is_retweet	processed	Lengths	Topics
0	1465162850457071621	2021-11-29 03:36:59+00:00	Will Boris Johnson ever learn his lesson after...	James robertson	NaN	NaN	2013-04-22 19:41:30+00:00	303.0	188	84059	NaN	Twitter for Android	False	boris johnson ever learn lesson endless covid ...	10	Topic 1
1	1465162841665769474	2021-11-29 03:36:57+00:00	#Omicron so hot; it's being rush shipped in De...	Gene Bulmer	East Coast	#ChristFollower, #Leader, Husband, Father, #Ru...	2014-11-10 22:43:27+00:00	99.0	162	745	['Omicron']	Twitter for iPhone	False	omicron rush shipped package symptom barely kn...	9	Topic 1

In [19]:

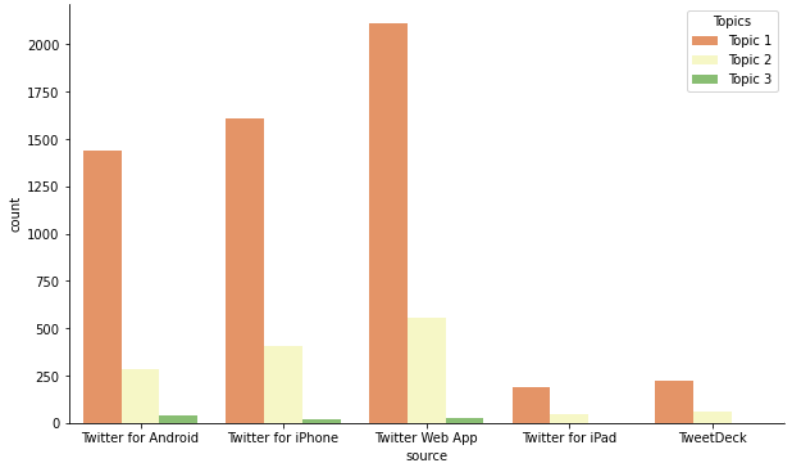
```python
# 1. Wordcloud of Top N words in each topic

from matplotlib import pyplot as plt
from wordcloud import WordCloud

cloud = WordCloud(background_color='white', width=2500, height=2000,
                  max_words=28, colormap='tab10', prefer_horizontal=1.0)

fig, axes = plt.subplots(1, 3, figsize=(15,8), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):

  fig.add_subplot(ax)
  if i>len(topics_name)-1:
    continue
  curr = data[data['Topics']==topics_name[i]]
  print(curr.shape)
  tokens = [tok for d in curr['processed'] for tok in d.split(' ')]
  cloud.generate(' '.join( tokens ))

  plt.gca().imshow(cloud)
  plt.gca().set_title( topics_name[i]+'\n')
  plt.gca().axis('off')

plt.axis('off')
plt.tight_layout()
plt.show()
```

```
(6233, 16)
(1738, 16)
(95, 16)
```



In [20]:

```
def subcategory_plot(df, col):

    plt.figure(figsize=(10,6))
    bar_pub = list( df[col].value_counts().index[:5] )
    temp2 = df[df[col].isin(bar_pub)]

    sns.countplot(x=col, hue='Topics', data=temp2, palette="RdYlGn")
    sns.despine()

    plt.savefig('bars_{}.png'.format(col))
    plt.show()

subcategory_plot(data, 'source')
```



In [21]:

```
# users have been categorized according to their number of followers
data['User type']=pd.cut(data.user_followers, [1,200, 1000, 2000, 10000, 100000000],
                         labels=['Naive', 'Average', 'Popular', 'Micro-Influencer', 'Influencer'])

subcategory_plot(data, 'User type')
```