In [1]:

```
import sys
#!pip3 install numpy
#!pip3 install pandas
!{sys.executable} -m spacy download en_core_web_sm
import re, numpy as np, pandas as pd
from pprint import pprint
# Gensim
#!pip install gensim
#!python3 -m pip3 install spacy
#!pip install matplot
#!pip install click
#!python3 -m spacy download en
#!python3 -m pip install gensim
#!python3 -m pip install matplot
#!python3 -m pip install nltk
#!pip install python-Levenshtein
!python3 -m pip install nltk.downloader stopwords
import gensim
import spacy
import logging
import warnings
import gensim.corpora as corpora
# from gensim.utils import lemmatize, simple_preprocess
from gensim.utils import  simple_preprocess
from gensim.models import CoherenceModel
import matplotlib.pyplot as plt
```

```
Collecting en-core-web-sm==3.1.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.1.0/en_core_web_sm-3.1.0-py3-none-any.whl (13.6 MB)
     |████████████████████████████████| 13.6 MB 14.8 MB/s eta 0:00:01
Requirement already satisfied: spacy<3.2.0,>=3.1.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from en-core-web-sm==3.1.0) (3.1.0)
Requirement already satisfied: pathy>=0.3.5 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.6.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.8.2)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.0.5)
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.3.2)
Requirement already satisfied: jinja2 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.0.1)
Requirement already satisfied: setuptools in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (49.6.0.post20210108)
Requirement already satisfied: packaging>=20.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (21.0)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.8.2)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.0.5)
Requirement already satisfied: catalogue<2.1.0,>=2.0.4 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.0.4)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.25.1)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.4.1)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.7 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.0.8)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (4.61.2)
Requirement already satisfied: numpy>=1.15.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.21.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.0.5)
Requirement already satisfied: thinc<8.1.0,>=8.0.7 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (8.0.7)
Requirement already satisfied: pyparsing>=2.0.2 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from packaging>=20.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.4.7)
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from pathy>=0.3.5->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (5.1.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.10.0.0)
Requirement already satisfied: idna<3,>=2.5 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.26.6)
Requirement already satisfied: chardet<5,>=3.0.2 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2021.5.30)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from typer<0.4.0,>=0.3.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages (from jinja2->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.0.1)
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
ERROR: Could not find a version that satisfies the requirement nltk.downloader (from versions: none)
ERROR: No matching distribution found for nltk.downloader
```

```
/Users/elissaiosbaxevanis/opt/anaconda3/envs/py39-demo/lib/python3.9/site-packages/gensim/similarities/__init__.py:15: UserWarning: The gensim.similarities.levenshtein submodule is disabled, because the optional Levenshtein 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 [2]:

```
#def fxn():
 #    warnings.warn("deprecated", DeprecationWarning)

#with warnings.catch_warnings():
 #    warnings.simplefilter("ignore")

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

Question 1

In [3]:

```
# NLTK Stop words
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['https','co','amp','from', 'subject', 're', 'edu', 'use', 'not', 'would',
'say', 'could', '_', 'be', 'know', 'good', 'go', 'get', 'do', 'done', 'try', 'many', 'some',
'nice', 'thank', 'think', 'see', 'rather', 'easy', 'easily', 'lot', 'lack', 'make', 'want',
'seem', 'run', 'need', 'even', 'right', 'line', 'even', 'also', 'may', 'take', 'come'])
import nltk
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

nltk.download()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/elissaiosbaxevanis/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

Out[3]:

```
True
```

In [4]:

```
df = pd.read_csv('tweets.csv')
print(df.head())
```

```
                   _id                                               text
0  1409477134209458181  My son and I went on a tour to the Allianz Are...
1  1409477134691835908  @Veronic35709033 @SFODan @SimonCalder @grantsh...
2  1409477135228719105  (Ethereum looks to retake $2K days before Lond...
3  1409477135711064066  No Riley Dean in the squad either. Suggests an...
4  1409477137204191238  Highest ever temperatures in Canada - and US n...
```

In [43]:

```
df['word_count'] = df.apply(lambda x:len(x['text'].strip().split()),axis=1)
```

In [44]:

```
df['word_count'].describe()
```

Out[44]:

```
count    53399.000000
mean        24.068447
std         13.321996
min          4.000000
25%         13.000000
50%         21.000000
75%         34.000000
max        104.000000
Name: word_count, dtype: float64
```

In [9]:

```
def sent_to_words(sentences):
    for sent in sentences:
        sent = re.sub('\S*@\S*\s?', '', sent)  # remove emails
        sent = re.sub('\s+', ' ', sent)  # remove newline chars
        sent = re.sub("\'", "", sent)  # remove single quotes
        sent = re.sub(r"http\S+", "", sent) #remove https
        sent = re.sub(r'&amp','',sent) #remove amp
        sent = gensim.utils.simple_preprocess(str(sent), deacc=True)
        yield(sent)

# Convert to list
data = df.text.values.tolist()
data_words = list(sent_to_words(data))
```

In [10]:

```
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=200) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=200)
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

def process_words(texts, stop_words=stop_words, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """Remove Stopwords, Form Bigrams, Trigrams and Lemmatization"""
    texts = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]
    texts = [bigram_mod[doc] for doc in texts]
    texts = [trigram_mod[bigram_mod[doc]] for doc in texts]
    texts_out = []
#     nlp = spacy.load('en', disable=['parser', 'ner'])
    nlp = spacy.load("en_core_web_sm")
    nlp.max_length = 200000
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    # remove stopwords once more after lemmatization
    texts_out = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts_out]
    return texts_out

data_ready = process_words(data_words)  # processed Text Data!
```

In [11]:

```
#create dictionary
id2word = corpora.Dictionary(data_ready)
#create corpus
corpus = [id2word.doc2bow(text) for text in data_ready]
```

In [12]:

```
from gensim.matutils import kullback_leibler
def kl_div_func(model):
    x = model.get_topics()
    score = 0
    counter = 0
    score_list = []
    for i in x:
        for j in x:
            if all(j!=i):
                counter +=1
                score += kullback_leibler(i,j)
            #else:
             #  score_list.append(0)
    return score/counter
```

In [108…

```
def compute_coherence_values(dictionary, corpus, texts, rng):
    """
    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 respective number of topics
    """
    coherence_values = []
    model_list = []
    perplexity_values = []
    kl_div_values = []
    for num_topics in rng:
        model = gensim.models.ldamulticore.LdaMulticore(corpus=corpus, num_topics=num_topics, id2word=id2word,
                                                random_state=100,
                                                workers=4,
                                                chunksize=1000,
                                                passes=10,
                                                alpha='symmetric',
                                                iterations=50,
                                                per_word_topics=True,)
        model_list.append(model)
        kl_div_values.append(kl_div_func(model))
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='u_mass')
        coherence_values.append(coherencemodel.get_coherence())
        perplexity_values.append(2**(-model.log_perplexity(corpus)))
        print("Num topics:",num_topics)
    return model_list, coherence_values, perplexity_values, kl_div_values
```

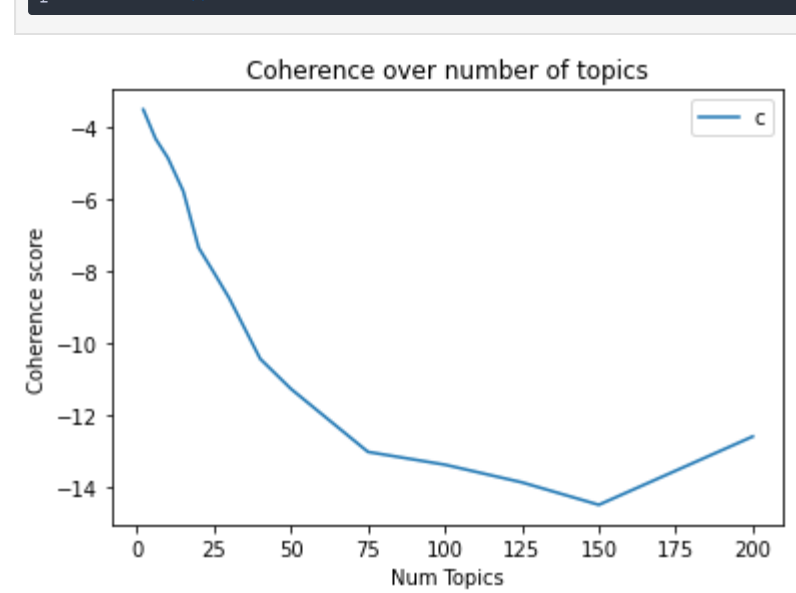
In [14]:

```
x = [2,6,10,15,20,25,30,40,50,75,100,125,150,200]
model_list, coherence_values, perplexity_values, kl_div_values = compute_coherence_values(dictionary=id2word,

corpus=corpus,

texts=data_ready,
                                                                              rng=x)
```

```
Num topics: 2
Num topics: 6
Num topics: 10
Num topics: 15
Num topics: 20
Num topics: 25
Num topics: 30
Num topics: 40
Num topics: 50
Num topics: 75
Num topics: 100
Num topics: 125
Num topics: 150
Num topics: 200
```

In [15]:

```
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.title("Coherence over number of topics")
plt.show()
plt.plot(x, perplexity_values)
plt.xlabel("Num Topics")
plt.ylabel("Perplexity score")
plt.legend(("perplexity_values"), loc='best')
plt.title("Perplexity over number of topics")
plt.show()
```



In [16]:

```
results = pd.DataFrame (data = [x,
                                  coherence_values,
                                 perplexity_values,
                                 kl_div_values]).T
results.columns = ['Num Topics', 'Coherence', 'Perplexity', 'KL Divergence']

results['Num Topics'] = results['Num Topics'].apply(lambda x: int(x))
results
```

Out[16]:

	Num Topics	Coherence	Perplexity	KL Divergence
0	2	-3.508417	3.632350e+02	2.022938
1	6	-4.322848	4.191805e+02	3.082971
2	10	-4.856069	5.962021e+02	4.096701
3	15	-5.780834	9.829778e+02	4.757345
4	20	-7.355326	1.361981e+03	5.344533
5	25	-8.052729	1.875987e+03	5.693070
6	30	-8.770779	2.628290e+03	6.020543
7	40	-10.447348	5.172345e+03	6.667627
8	50	-11.283510	1.048559e+04	6.853809
9	75	-13.035845	9.876318e+04	7.361565
10	100	-13.391958	1.607257e+06	7.668391
11	125	-13.883238	1.726601e+07	8.027174
12	150	-14.508615	1.183000e+09	8.569816
13	200	-12.607126	7.220886e+16	8.880422

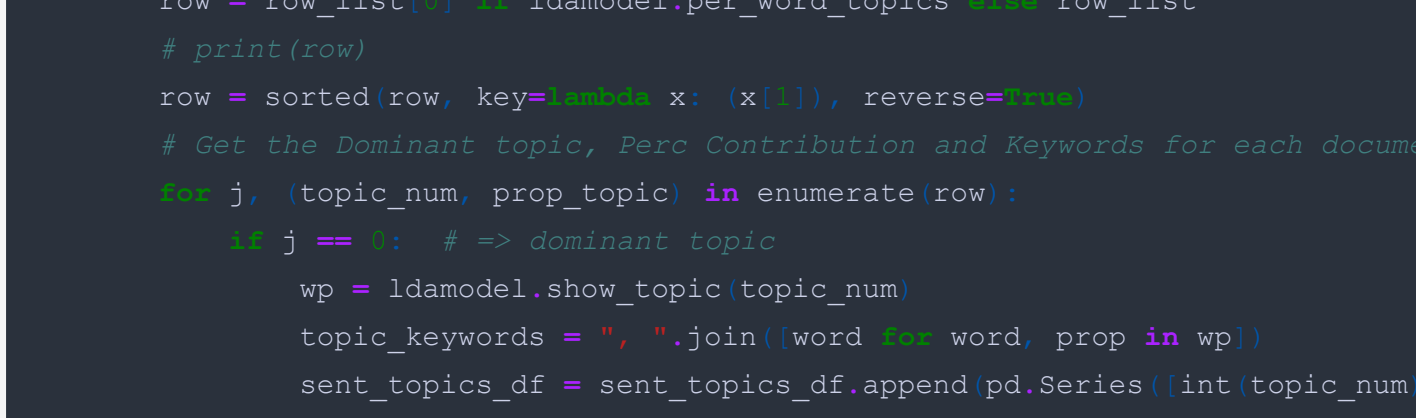
In [207…

```
print(results.iloc[3])
optimal_model = model_list[3]
model_topics = optimal_model.show_topics()
pprint(optimal_model.print_topics())
```

```
Num Topics        15.000000
Coherence         -5.780834
Perplexity       982.977781
KL Divergence      4.757345
Name: 3, dtype: float64
[(0,
  '0.019*"people" + 0.016*"support" + 0.016*"happen" + 0.014*"thing" + '
  '0.014*"tell" + 0.011*"call" + 0.010*"believe" + 0.008*"face" + '
  '0.007*"absolutely" + 0.007*"freedom"'),
 (1,
  '0.024*"listen" + 0.024*"people" + 0.020*"die" + 0.015*"live" + '
  '0.012*"almost" + 0.010*"eat" + 0.009*"touch" + 0.009*"kid" + '
  '0.009*"nowplaye" + 0.008*"shirt"'),
 (2,
  '0.028*"watch" + 0.026*"really" + 0.018*"hope" + 0.017*"love" + 0.017*"back" '
  '+ 0.014*"fuck" + 0.013*"well" + 0.012*"feel" + 0.012*"day" + '
  '0.011*"tournament"'),
 (3,
  '0.044*"time" + 0.043*"year" + 0.029*"week" + 0.025*"last" + 0.019*"day" + '
  '0.015*"old" + 0.014*"next" + 0.014*"end" + 0.012*"month" + 0.011*"start"'),
 (4,
  '0.013*"family" + 0.012*"book" + 0.012*"child" + 0.012*"friend" + '
  '0.009*"support" + 0.009*"great" + 0.009*"free" + 0.009*"amazing" + '
  '0.008*"mate" + 0.008*"love"'),
 (5,
  '0.041*"game" + 0.038*"play" + 0.033*"tomorrow" + 0.028*"euro" + 0.027*"win" '
  '+ 0.027*"team" + 0.018*"football" + 0.017*"goal" + 0.014*"match" + '
  '0.013*"player"'),
 (6,
  '0.026*"case" + 0.024*"covid" + 0.023*"vaccine" + 0.019*"number" + '
  '0.017*"loveisland" + 0.016*"death" + 0.014*"test" + 0.013*"high" + '
  '0.011*"probably" + 0.010*"state"'),
 (7,
  '0.036*"new" + 0.027*"sign" + 0.026*"labour" + 0.024*"name" + 0.021*"add" + '
  '0.014*"role" + 0.012*"country" + 0.011*"party" + 0.011*"lie" + '
  '0.008*"brexit"'),
 (8,
  '0.017*"pay" + 0.015*"care" + 0.012*"public" + 0.009*"health" + '
  '0.008*"support" + 0.008*"instead" + 0.008*"job" + 0.007*"help" + '
  '0.007*"worker" + 0.007*"service"'),
 (9,
  '0.028*"race" + 0.027*"home" + 0.023*"racing" + 0.021*"car" + 0.013*"road" + '
  '0.011*"posted_photo" + 0.010*"send" + 0.010*"drive" + 0.010*"help" + '
  '0.010*"weekend"'),
 (10,
  '0.034*"conservative" + 0.016*"people" + 0.011*"problem" + 0.011*"police" + '
  '0.011*"fire" + 0.010*"black" + 0.009*"woman" + 0.008*"liberal" + '
  '0.008*"post" + 0.008*"comment"'),
 (11,
  '0.018*"news" + 0.016*"new" + 0.010*"check" + 0.010*"read" + 0.010*"order" + '
  '0.009*"find" + 0.009*"link" + 0.009*"pm" + 0.008*"late" + '
  '0.008*"available"'),
 (12,
  '0.026*"look" + 0.021*"city" + 0.014*"manchester" + 0.013*"money" + '
  '0.009*"man" + 0.008*"save" + 0.008*"time" + 0.008*"twitter" + '
  '0.007*"travel" + 0.007*"well"'),
 (13,
  '0.031*"labour" + 0.025*"tory" + 0.019*"vote" + 0.017*"people" + '
  '0.011*"government" + 0.010*"beat" + 0.010*"leave" + 0.010*"well" + '
  '0.009*"score" + 0.009*"lose"'),
 (14,
  '0.024*"work" + 0.013*"great" + 0.013*"school" + 0.012*"big" + 0.010*"class" '
  '+ 0.009*"opportunity" + 0.009*"learn" + 0.009*"tweet" + 0.009*"look" + '
  '0.008*"definitely"')]
```

In [208…

```
from matplotlib import pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import matplotlib.colors as mcolors

cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]  # more colors: 'mcolors.XKCD_COLORS'

cloud = WordCloud(stopwords=stop_words,
                  background_color='white',
                  width=2500,
                  height=1800,
                  max_words=10,
                  colormap='tab10',
                  color_func=lambda *args, **kwargs: cols[i],
                  prefer_horizontal=1.0)

topics = optimal_model.show_topics(formatted=False)

fig, axes = plt.subplots(2, 5, figsize=(10,5), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    fig.add_subplot(ax)
    topic_words = dict(topics[i][1])
    cloud.generate_from_frequencies(topic_words, max_font_size=300)
    plt.gca().imshow(cloud)
    plt.gca().set_title('Topic ' + str(i), fontdict=dict(size=16))
    plt.gca().axis('off')


plt.subplots_adjust(wspace=0, hspace=0)
plt.axis('off')
plt.margins(x=0, y=0)
plt.tight_layout()
plt.show()
```



In [209…

```
def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        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), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # 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=corpus, texts=data_ready)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']
df_dominant_topic.head()
```

Out[209…

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	12.0	0.5192	look, city, manchester, money, man, save, time...	[arena, admire, seat, arena, suddenly, point, ...
1	1	0.0	0.5319	people, support, happen, thing, tell, call, be...	[tell, tell, change, position]
2	2	12.0	0.6603	look, city, manchester, money, man, save, time...	[look, retake, day, big, breakout_ahead, crypt...
3	3	2.0	0.4878	watch, really, hope, love, back, fuck, well, f...	[suggest, half, back, partnership, thewli, sta...
4	4	6.0	0.4040	case, covid, vaccine, number, loveisland, deat...	[high, ever, temperature, likely, follow, well...

In [211…

```
for i,r in df_dominant_topic.head(5).iterrows():
    print('Document:',r['Document_No'], '\nDominant topic:',r['Dominant_Topic'])
    print('Dominant Topic Contribution:',round(r['Topic_Perc_Contrib'],3))
    print('Keywords:',r['Keywords'], '\nText:',r['Text'])
    print("******************************")
```

```
Document: 0
Dominant topic: 12.0
Dominant Topic Contribution: 0.519
Keywords: look, city, manchester, money, man, save, time, twitter, travel, well
Text: ['arena', 'admire', 'seat', 'arena', 'suddenly', 'point', 'pitch', 'dad', 'man', 'camping', 'son', 'penaldo', 'live', 'penalty', 'box', 'perform', 'small', 'game']
******************************
Document: 1
Dominant topic: 0.0
Dominant Topic Contribution: 0.532
Keywords: people, support, happen, thing, tell, call, believe, face, absolutely, freedom
Text: ['tell', 'tell', 'change', 'position']
******************************
Document: 2
Dominant topic: 12.0
Dominant Topic Contribution: 0.66
Keywords: look, city, manchester, money, man, save, time, twitter, travel, well
Text: ['look', 'retake', 'day', 'big', 'breakout_ahead', 'cryptollic']
******************************
Document: 3
Dominant topic: 2.0
Dominant Topic Contribution: 0.488
Keywords: watch, really, hope, love, back, fuck, well, feel, day, tournament
Text: ['suggest', 'half', 'back', 'partnership', 'thewli', 'stay', 'full', 'back']
******************************
Document: 4
Dominant topic: 6.0
Dominant Topic Contribution: 0.404
Keywords: case, covid, vaccine, number, loveisland, death, test, high, probably, state
Text: ['high', 'ever', 'temperature', 'likely', 'follow', 'well', 'spout', 'platitude', 'global', 'warning', 'last', 'week']
******************************
```

In [212…

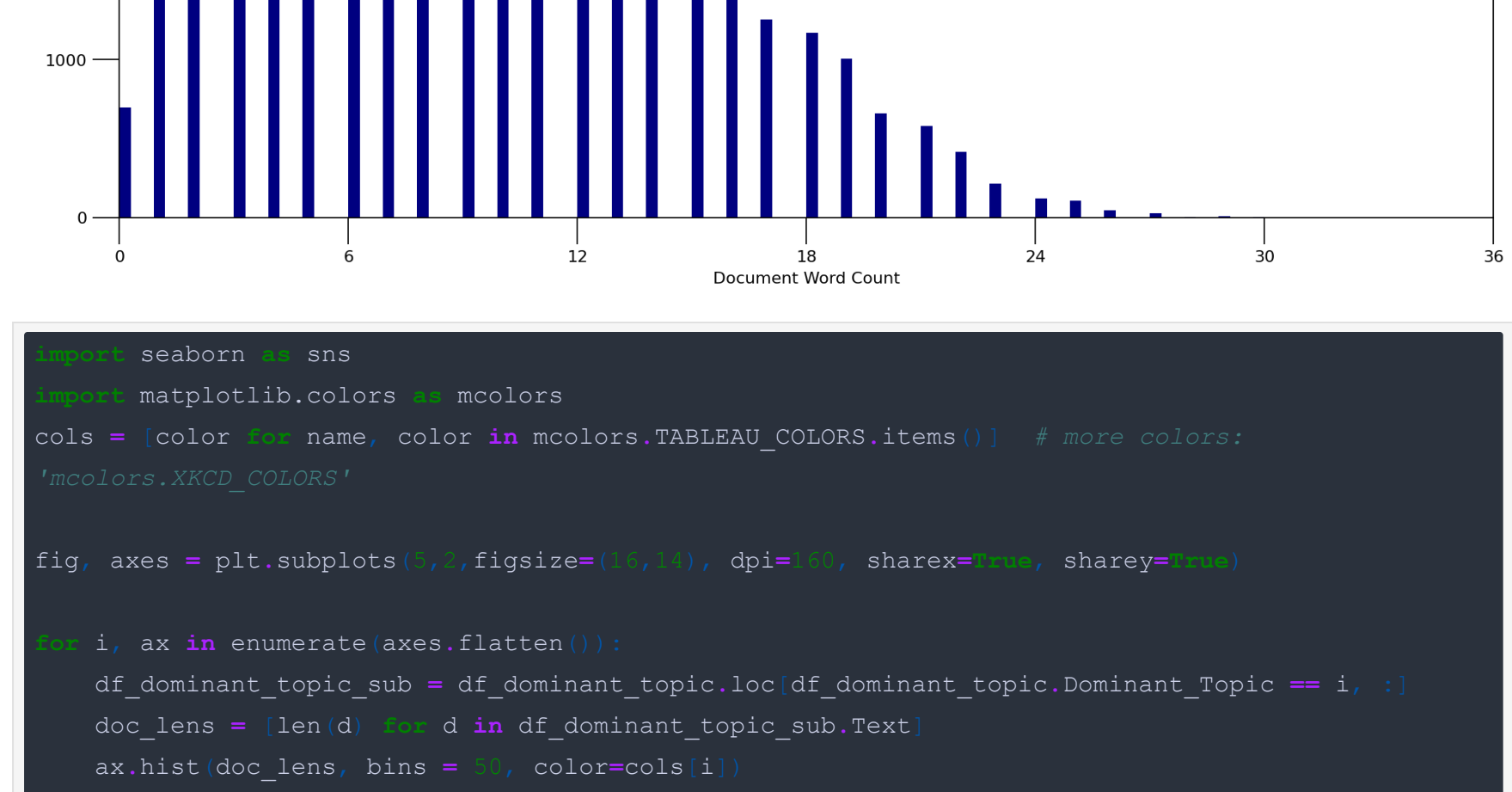
```
# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# 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_contribution], axis=1)

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

# Show
df_dominant_topics.head()
```

Out[212…

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
0.0	12.0	look, city, manchester, money, man, save, time...	3962.0	0.0742
1.0	0.0	people, support, happen, thing, tell, call, be...	2200.0	0.0412
2.0	12.0	look, city, manchester, money, man, save, time...	4513.0	0.0845
3.0	2.0	watch, really, hope, love, back, fuck, well, f...	4983.0	0.0933
4.0	6.0	case, covid, vaccine, number, loveisland, deat...	2838.0	0.0531

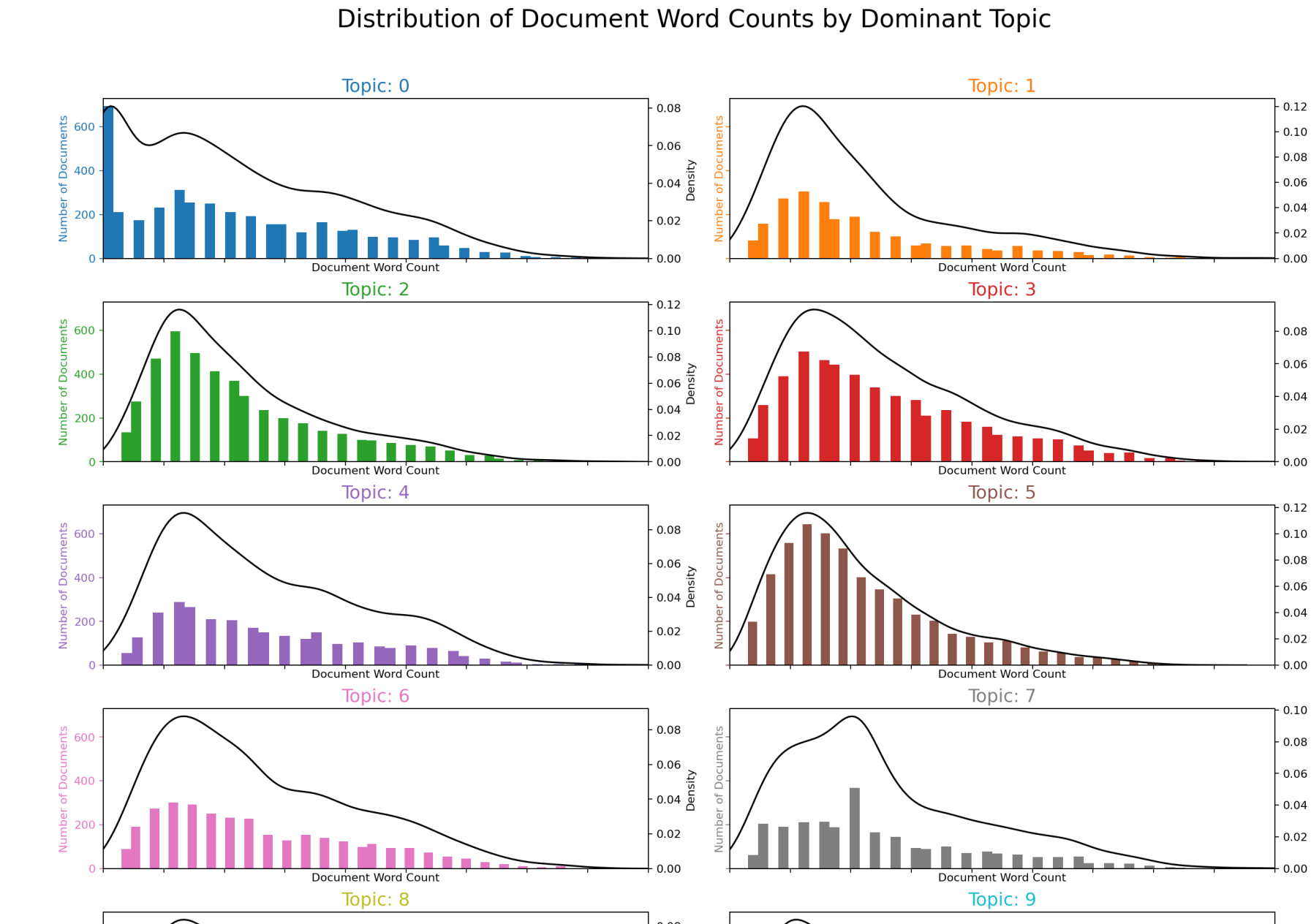
In [22]:

```
doc_lens = [len(d) for d in df_dominant_topic.Text]

# Plot
plt.figure(figsize=(15,10), dpi=160)
plt.hist(doc_lens, bins = 100, color='navy')
plt.text(27.5,  1900, "Mean   : " + str(round(np.mean(doc_lens))))
plt.text(27.5,  1800, "Median : " + str(round(np.median(doc_lens))))
plt.text(27.5,  1700, "Stdev   : " + str(round(np.std(doc_lens))))
plt.text(27.5,  1600, "1%ile    : " + str(round(np.quantile(doc_lens, q=0.01))))
plt.text(27.5,  1500, "99%ile  : " + str(round(np.quantile(doc_lens, q=0.99))))

plt.gca().set(xlim=(0, 36), ylabel='Number of Documents', xlabel='Document Word Count')
plt.tick_params(size=16)
plt.xticks(np.linspace(0,36,7))
plt.title('Distribution of Document Word Counts', fontdict=dict(size=20))
plt.fontdict=dict(size=16)
plt.show()
```



In [23]:

```
import seaborn as sns
import matplotlib.colors as mcolors
cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]  # more colors: 'mcolors.XKCD_COLORS'

fig, axes = plt.subplots(5,2,figsize=(16,14), dpi=160, sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    df_dominant_topic_sub = df_dominant_topic.loc[df_dominant_topic.Dominant_Topic == i, :]
    doc_lens = [len(d) for d in df_dominant_topic_sub.Text]
    ax.hist(doc_lens, bins = 50, color=cols[i])
    ax.tick_params(axis='y', labelcolor=cols[i], color=cols[i])
    sns.kdeplot(doc_lens, color="black", shade=False, ax=ax.twinx())
    ax.set(xlim=(0, 30), xlabel='Document Word Count')
    ax.set_ylabel('Number of Documents', color=cols[i])
    ax.set_title('Topic: '+str(i), fontdict=dict(size=16, color=cols[i]))

fig.tight_layout()
fig.subplots_adjust(top=0.90)
plt.xticks(np.linspace(0,30,10))
fig.suptitle('Distribution of Document Word Counts by Dominant Topic', fontsize=22)
plt.show()
```



Question 2

In [108…
```python
from gensim import corpora, models
from sklearn import preprocessing
from sklearn.cluster import KMeans
from sklearn.cluster import MiniBatchKMeans
from sklearn.feature_extraction.text import TfidfVectorizer

data_ready_joined = [' '.join(x) for x in data_ready]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data_ready_joined)
X_normalised = preprocessing.normalize(X)

kmeans = KMeans(n_clusters=2000, init='k-means++',
                random_state=[illegible], verbose=1).fit(X_normalised)
labels = kmeans.predict(X_normalised)
```

```
Initialization complete
[illegible]
Iteration 15, inertia 40678.701905070884
Iteration 16, inertia 40678.39396970621
Iteration 17, inertia 40678.177103103
Iteration 18, inertia 40677.95368151096
Iteration 19, inertia 40677.86085516916
Iteration 20, inertia 40677.78828718969
Iteration 21, inertia 40677.66233361622
Iteration 22, inertia 40677.52216715702
Iteration 23, inertia 40677.38295848105
Iteration 24, inertia 40677.36188318104
Iteration 25, inertia 40677.35003774849
Converged at iteration 25: strict convergence.
Initialization complete
Iteration 0, inertia 46727.360515128006
Iteration 1, inertia 44135.38242067006
Iteration 2, inertia 42343.545940753735
Iteration 3, inertia 41388.04294515709
Iteration 4, inertia 40998.371718055874
Iteration 5, inertia 40830.258178158736
Iteration 6, inertia 40755.17001354489
Iteration 7, inertia 40718.71890066001
Iteration 8, inertia 40699.53025262958
Iteration 9, inertia 40689.147093463565
Iteration 10, inertia 40683.19392410778
Iteration 11, inertia 40680.240226403745
Iteration 12, inertia 40678.08637172771
Iteration 13, inertia 40676.951483768535
Iteration 14, inertia 40676.5506335152
Iteration 15, inertia 40676.15085675347
Iteration 16, inertia 40675.84364788274
Iteration 17, inertia 40675.73509358366
Iteration 18, inertia 40675.67621620432
Iteration 19, inertia 40675.59456838392
Iteration 20, inertia 40675.54401467192
Iteration 21, inertia 40675.442219488265
Iteration 22, inertia 40675.40764773071
Iteration 23, inertia 40675.37979033927
Converged at iteration 23: strict convergence.
Initialization complete
Iteration 0, inertia 46746.173158986065
Iteration 1, inertia 44129.77467631848
Iteration 2, inertia 42341.30886401051
Iteration 3, inertia 41417.61998583119
Iteration 4, inertia 41052.45762753114
Iteration 5, inertia 40909.26034398588
Iteration 6, inertia 40844.75249752082
Iteration 7, inertia 40813.95304405188
Iteration 8, inertia 40796.251854937844
Iteration 9, inertia 40785.79631856759
Iteration 10, inertia 40779.044378921004
Iteration 11, inertia 40775.086806267296
Iteration 12, inertia 40772.17881104218
Iteration 13, inertia 40770.28426686618
Iteration 14, inertia 40769.14680394454
Iteration 15, inertia 40768.51308447028
Iteration 16, inertia 40767.231317953294
Iteration 17, inertia 40766.4449733886
Iteration 18, inertia 40765.71909223623
Iteration 19, inertia 40765.346089071354
Iteration 20, inertia 40765.23607542194
Iteration 21, inertia 40765.17765877033
Iteration 22, inertia 40765.14753009585
Iteration 23, inertia 40765.13961454556
Converged at iteration 23: strict convergence.
Initialization complete
Iteration 0, inertia 46734.44421152008
Iteration 1, inertia 44121.26007766974
Iteration 2, inertia 42339.297998876085
Iteration 3, inertia 41411.24293678748
Iteration 4, inertia 41047.4350900654
Iteration 5, inertia 40896.91258793108
Iteration 6, inertia 40830.89323404175
Iteration 7, inertia 40798.336799979574
Iteration 8, inertia 40781.23899043614
Iteration 9, inertia 40770.70671463425
Iteration 10, inertia 40763.31305667024
Iteration 11, inertia 40759.39895499076
Iteration 12, inertia 40757.084457458266
Iteration 13, inertia 40756.02230169211
Iteration 14, inertia 40755.451859425964
Iteration 15, inertia 40754.9767753381
Iteration 16, inertia 40754.545272472235
Iteration 17, inertia 40754.23333296165
Iteration 18, inertia 40753.81908065614
Iteration 19, inertia 40753.634369024054
Iteration 20, inertia 40753.531984627014
Iteration 21, inertia 40753.46874088383
Iteration 22, inertia 40753.44586413232
Iteration 23, inertia 40753.437523726585
Iteration 24, inertia 40753.430165131715
Iteration 25, inertia 40753.413260617526
Iteration 26, inertia 40753.39261739049
Iteration 27, inertia 40753.35285404708
Iteration 28, inertia 40753.34084649515
Converged at iteration 28: strict convergence.
Initialization complete
Iteration 0, inertia 46690.05028669648
Iteration 1, inertia 44064.188849535196
Iteration 2, inertia 42287.19236755685
Iteration 3, inertia 41353.70763602796
Iteration 4, inertia 40970.215161335946
Iteration 5, inertia 40807.00128765165
Iteration 6, inertia 40730.81849948304
Iteration 7, inertia 40694.74416973884
Iteration 8, inertia 40674.176913005904
Iteration 9, inertia 40660.13657250985
Iteration 10, inertia 40649.149317367555
Iteration 11, inertia 40643.61430869112
Iteration 12, inertia 40639.84612402973
Iteration 13, inertia 40637.84575808893
Iteration 14, inertia 40636.71223999934
Iteration 15, inertia 40636.133615443476
Iteration 16, inertia 40635.71595277232
Iteration 17, inertia 40635.53813004648
Iteration 18, inertia 40635.28968560105
Iteration 19, inertia 40635.148895973594
Iteration 20, inertia 40635.04898689488
Iteration 21, inertia 40634.851587561345
Iteration 22, inertia 40634.802883126074
Iteration 23, inertia 40634.76648805197
Converged at iteration 23: strict convergence.
Initialization complete
Iteration 0, inertia 46657.35630996885
Iteration 1, inertia 44110.98119455133
Iteration 2, inertia 42319.6651657073
Iteration 3, inertia 41392.54465805931
Iteration 4, inertia 41036.086756623314
Iteration 5, inertia 40896.83533275122
Iteration 6, inertia 40831.38466776782
Iteration 7, inertia 40792.20396439257
Iteration 8, inertia 40769.15340126377
Iteration 9, inertia 40755.03274831069
Iteration 10, inertia 40747.76605236902
Iteration 11, inertia 40742.36777157968
Iteration 12, inertia 40738.92365980132
Iteration 13, inertia 40736.39731740539
Iteration 14, inertia 40734.60237226231
Iteration 15, inertia 40731.23207623418
Iteration 16, inertia 40730.35321334469
Iteration 17, inertia 40729.8581455827
Iteration 18, inertia 40729.601617703775
Iteration 19, inertia 40729.538984713006
Iteration 20, inertia 40729.49422100294
Iteration 21, inertia 40729.45444297817
Iteration 22, inertia 40729.44916540866
Converged at iteration 22: strict convergence.
Initialization complete
Iteration 0, inertia 46642.757057926
Iteration 1, inertia 44114.05748108758
Iteration 2, inertia 42366.07274628276
Iteration 3, inertia 41432.32581199269
Iteration 4, inertia 41053.24871222908
Iteration 5, inertia 40900.802331933904
Iteration 6, inertia 40829.51534201843
Iteration 7, inertia 40795.17312159057
Iteration 8, inertia 40777.61219123836
Iteration 9, inertia 40767.9806505158
Iteration 10, inertia 40762.768889602274
Iteration 11, inertia 40759.16801625811
Iteration 12, inertia 40756.320400659584
Iteration 13, inertia 40753.97016776258
Iteration 14, inertia 40752.72468860281
Iteration 15, inertia 40751.805363991574
Iteration 16, inertia 40750.88456109975
Iteration 17, inertia 40749.83950209231
Iteration 18, inertia 40748.945626579676
Iteration 19, inertia 40748.47119191348
Iteration 20, inertia 40748.19767265437
Iteration 21, inertia 40748.04058207154
Iteration 22, inertia 40747.89067716657
Iteration 23, inertia 40747.84126348619
Iteration 24, inertia 40747.7996358904
Iteration 25, inertia 40747.782984137084
Iteration 26, inertia 40747.71514490848
Iteration 27, inertia 40747.64167086519
Converged at iteration 27: strict convergence.
Initialization complete
Iteration 0, inertia 46640.50895085083
Iteration 1, inertia 44043.20073208792
Iteration 2, inertia 42324.183912550696
Iteration 3, inertia 41432.4502757048
Iteration 4, inertia 41082.959810698325
Iteration 5, inertia 40939.40476502182
Iteration 6, inertia 40870.43900476872
Iteration 7, inertia 40833.76361552657
Iteration 8, inertia 40814.32898804301
Iteration 9, inertia 40803.338219410914
Iteration 10, inertia 40796.68627268163
Iteration 11, inertia 40792.191813014884
Iteration 12, inertia 40789.878933213324
Iteration 13, inertia 40788.24102822233
Iteration 14, inertia 40787.39483293856
Iteration 15, inertia 40786.84078639672
Iteration 16, inertia 40786.44589102815
Iteration 17, inertia 40786.1728913348
Iteration 18, inertia 40786.03340048974
Iteration 19, inertia 40785.9052861757
Iteration 20, inertia 40785.76330191109
Iteration 21, inertia 40785.70036117174
Converged at iteration 21: strict convergence.
Initialization complete
Iteration 0, inertia 46687.419886689
Iteration 1, inertia 44077.42859836541
Iteration 2, inertia 42318.32384827878
Iteration 3, inertia 41392.14569040401
Iteration 4, inertia 41013.80541210987
Iteration 5, inertia 40868.27292306214
Iteration 6, inertia 40804.00841851317
Iteration 7, inertia 40775.06598691289
Iteration 8, inertia 40758.48455395214
Iteration 9, inertia 40749.39996629425
Iteration 10, inertia 40742.96550404284
Iteration 11, inertia 40738.80015198877
Iteration 12, inertia 40736.41244082353
Iteration 13, inertia 40734.93190890679
Iteration 14, inertia 40733.18496202586
Iteration 15, inertia 40732.14137923378
Iteration 16, inertia 40731.67229842993
Iteration 17, inertia 40731.28649960898
Iteration 18, inertia 40730.845247665995
Iteration 19, inertia 40730.69360295192
Iteration 20, inertia 40730.54622300482
Iteration 21, inertia 40730.337059063204
Iteration 22, inertia 40730.078851956714
Iteration 23, inertia 40729.88091078307
Iteration 24, inertia 40729.70121380341
Iteration 25, inertia 40729.62100861331
Converged at iteration 25: strict convergence.
Initialization complete
Iteration 0, inertia 46650.81495033527
Iteration 1, inertia 44056.08496721898
Iteration 2, inertia 42313.63425874588
Iteration 3, inertia 41379.706319112105
Iteration 4, inertia 40988.09566912087
Iteration 5, inertia 40834.36277207186
Iteration 6, inertia 40769.07423613576
Iteration 7, inertia 40735.50688079718
Iteration 8, inertia 40718.04705295063
Iteration 9, inertia 40709.399822296975
Iteration 10, inertia 40703.93939410133
Iteration 11, inertia 40701.43057739934
Iteration 12, inertia 40699.363577617645
Iteration 13, inertia 40698.247864805926
Iteration 14, inertia 40697.48354090704
Iteration 15, inertia 40696.97442888281
Iteration 16, inertia 40696.54929013527
Iteration 17, inertia 40696.26826132248
Iteration 18, inertia 40696.074930998264
Iteration 19, inertia 40695.91433144401
Iteration 20, inertia 40695.76351426999
Iteration 21, inertia 40695.550871158805
Iteration 22, inertia 40695.43800347732
Iteration 23, inertia 40695.340032548425
Iteration 24, inertia 40695.26092091764
Iteration 25, inertia 40695.16525200498
Iteration 26, inertia 40695.042894664155
Iteration 27, inertia 40694.91874010412
Iteration 28, inertia 40694.73212215039
Iteration 29, inertia 40694.52965329397
Iteration 30, inertia 40694.327481680106
Iteration 31, inertia 40694.17122169081
Iteration 32, inertia 40694.02303420591
Iteration 33, inertia 40693.81488535831
Iteration 34, inertia 40693.70614866655
Iteration 35, inertia 40693.64449603126
Iteration 36, inertia 40693.61058504709
Iteration 37, inertia 40693.583535195285
Iteration 38, inertia 40693.5534573157
Converged at iteration 38: strict convergence.
```

In [231…
```python
df_dominant_topic['Cluster'] = labels
df_new = df_dominant_topic[['Cluster','Text']]
df_new = df_new.groupby(['Cluster'])['Text'].agg(sum)
#df_new = pd.DataFrame({'Cluster':ser.index, 'Text':ser.values})
df_new
```

Out[231…
```
Cluster
0       [preface, conservative, racist, preface, volum...
1       [deny, read, tweet, clear, wtc, labour, rob, t...
2       [wide, vote, vote, remain, idiot, comprehend, ...
3       [humble, tribute, narsimha, rao, former, spotl...
4       [depart, depart, cape, fear, green, list, depa...
                              ...
1995    [free, channel, taxi, drive, coach, hotel, inc...
1996    [talking, point, roja, survive, dramatic, talk...
1997    [little, reminder, sticker, beauty, list, etsy...
1998    [conservative, vow, never, watch, hammer, thro...
1999    [abi, willing, send, wonderful, child, poster,...
Name: Text, Length: 2000, dtype: object
```

In [230…
```python
df_new['word_count'] = df_new.apply(lambda x:len(x['Text'].strip().split()),axis=1)
df_new['word_count'].describe()
```

Out[230…
```
count      2000.000000
mean        616.916000
std        3530.153949
min           5.000000
25%          43.000000
50%          99.000000
75%         506.250000
max      151355.000000
Name: word_count, dtype: float64
```

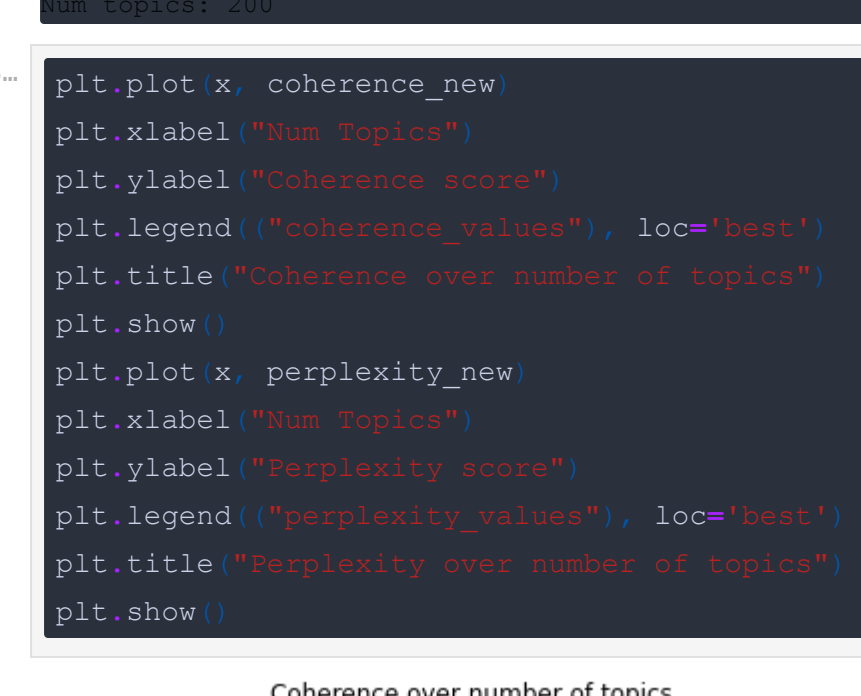
In [232…
```python
bigram = gensim.models.Phrases(data_words_new, min_count=5, threshold=200) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words_new], threshold=200)
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

def process_words(texts, stop_words=stop_words, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """Remove Stopwords, Form Bigrams, Trigrams and Lemmatization"""
    texts = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]
    texts = [bigram_mod[doc] for doc in texts]
    texts = [trigram_mod[bigram_mod[doc]] for doc in texts]
    texts_out = []
#   nlp = spacy.load('en', disable=['parser', 'ner'])
    nlp = spacy.load('en_core_web_sm')
    nlp.max_length = [illegible]
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    # remove stopwords once more after lemmatization
    texts_out = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts_out]
    return texts_out

data_ready_new = process_words(list(df_new))
```

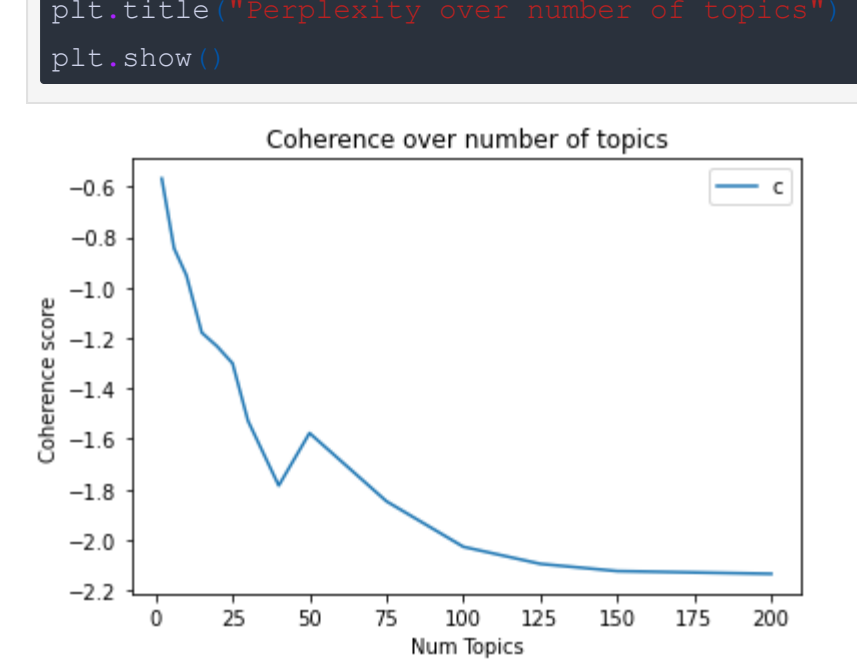
In [233…
```python
#create dictionary
id2word_new = corpora.Dictionary(data_ready_new)
#create corpus
corpus_new = [id2word_new.doc2bow(text) for text in data_ready_new]
```

In [234…
```python
def compute_coherence_values(dictionary, corpus, texts, rng):
    """
    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 respective number of topics
    """
    coherence_values = []
    model_list = []
    perplexity_values = []
    kl_div_values = []
    for num_topics in rng:
        model = gensim.models.ldamulticore.LdaMulticore(corpus=corpus, num_topics=num_topics, id2word=id2word,
                                                random_state=100,
                                                workers=4,
                                                chunksize=1000,
                                                passes=10,
                                                alpha='symmetric',
                                                iterations=50,
                                                per_word_topics=True,)
        model_list.append(model)
        kl_div_values.append(kl_div_func(model))
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='u_mass')
        coherence_values.append(coherencemodel.get_coherence())
        perplexity_values.append(2**(-model.log_perplexity(corpus)))
        print("Num topics:",num_topics)
    return model_list, coherence_values, perplexity_values, kl_div_values

x = [2,6,10,15,20,25,30,40,50,75,100,125,150,200]
model_listnew, coherence_new, perplexity_new, kl_div_new = compute_coherence_values(dictionary=id2word_new,

corpus=corpus_new,

texts=data_ready_new,
                                                                                    rng=x)
```

```
Num topics: 2
Num topics: 6
Num topics: 10
Num topics: 15
Num topics: 20
Num topics: 25
Num topics: 30
Num topics: 40
Num topics: 50
Num topics: 75
Num topics: 100
Num topics: 125
Num topics: 150
Num topics: 200
```

In [235…
```python
plt.plot(x, coherence_new)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.title("Coherence over number of topics")
plt.show()
plt.plot(x, perplexity_new)
plt.xlabel("Num Topics")
plt.ylabel("Perplexity score")
plt.legend(("perplexity_values"), loc='best')
plt.title("Perplexity over number of topics")
plt.show()
```





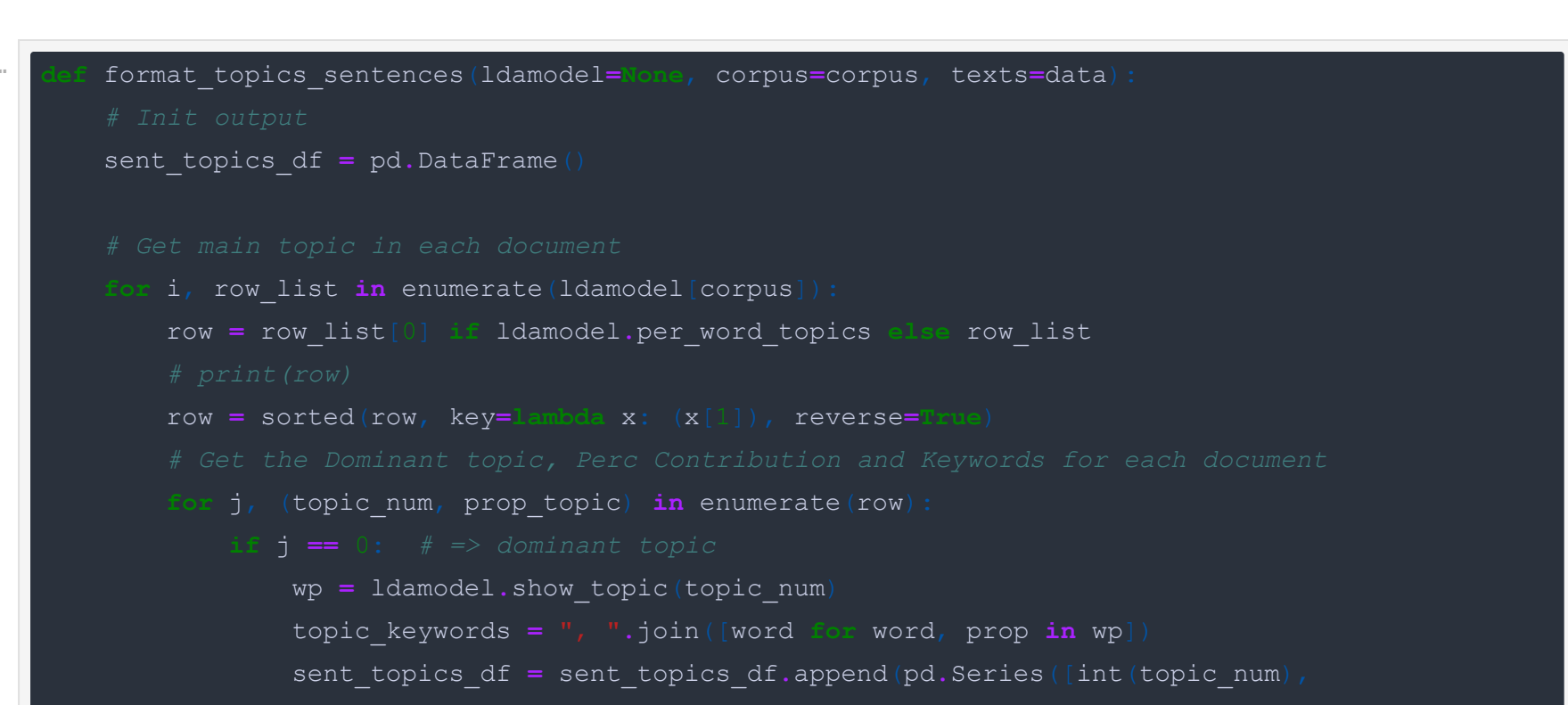
In [237…
```python
results_new = pd.DataFrame (data = [x,
                                    coherence_new,
                                   perplexity_new,
                                   kl_div_new]).T
results_new.columns = ['Num Topics', 'Coherence', 'Perplexity', 'KL Divergence']

results_new['Num Topics'] = results_new['Num Topics'].apply(lambda x: int(x))
results_new
```

Out[237…

	Num Topics	Coherence	Perplexity	KL Divergence
0	2	-0.567685	335.450281	0.300335
1	6	-0.844670	337.028777	0.647678
2	10	-0.951322	339.783860	0.900427
3	15	-1.178870	346.635754	1.158070
4	20	-1.233415	394.646467	1.345367
5	25	-1.300419	427.517353	1.510271
6	30	-1.528056	464.385954	1.663062
7	40	-1.784999	541.024809	1.933453
8	50	-1.576982	650.360994	2.161757
9	75	-1.847373	961.997117	2.612865
10	100	-2.027892	1507.850378	2.917681
11	125	-2.095475	2347.645630	3.229738
12	150	-2.124150	3156.035539	3.451730
13	200	-2.134961	7415.086333	3.736823

In [238…
```python
optimal_model_new = model_listnew[8]
```

In [239…
```python
from matplotlib import pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import matplotlib.colors as mcolors

cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]  # more colors: 'mcolors.XKCD_COLORS'

cloud = WordCloud(stopwords=stop_words,
                  background_color='white',
                  width=2500,
                  height=1800,
                  max_words=10,
                  colormap='tab10',
                  color_func=lambda *args, **kwargs: cols[i],
                  prefer_horizontal=1.0)

topics = optimal_model_new.show_topics(formatted=False)

fig, axes = plt.subplots(2, 5, figsize=(10,10), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    fig.add_subplot(ax)
    topic_words = dict(topics[i][1])
    cloud.generate_from_frequencies(topic_words, max_font_size=300)
    plt.gca().imshow(cloud)
    plt.gca().set_title('Topic ' + str(i), fontdict=dict(size=16))
    plt.gca().axis('off')


plt.subplots_adjust(wspace=0, hspace=0)
plt.axis('off')
plt.margins(x=0, y=0)
plt.tight_layout()
plt.show()
```



In [248…
```python
def format_topic_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        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),
round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # 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_topic_sentences(ldamodel=optimal_model_new, corpus=corpus_new,
texts=data_ready_new)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords',
'Text']
df_dominant_topic.head()
```

Out[248…

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	30.0	0.4683	admire, level, diagnose, thesis, sewing, tradi...	[preface, conservative, racist, preface, volum...
1	1	38.0	0.5839	exclude, hon, clog, lankan, salary, medusa, ti...	[deny, read, clear, taxpayer, big, business, k...
2	2	34.0	0.5520	stay, categorisation, cpfc, instance, sputnikv...	[wide, vote, vote, remain, comprehend, simple,...
3	3	0.0	0.4551	naturally, thur, legalwalk, fest, comparativel...	[humble, tribute, narsimha, rao, former, spotl...
4	4	29.0	0.8911	woollen, clouds_wind, dungeness, regret, mood,...	[depart, depart, cape, fear, green, list, depa...

In [265…
```python
# 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_contribution], axis=1)

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

# Show
df_dominant_topics.head()
```

Out[265…

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
0.0	30.0	admire, level, diagnose, thesis, sewing, tradi...	79.0	0.0395
1.0	38.0	exclude, hon, clog, lankan, salary, medusa, ti...	32.0	0.0160
2.0	34.0	stay, categorisation, cpfc, instance, sputnikv...	22.0	0.0110
3.0	0.0	naturally, thur, legalwalk, fest, comparativel...	27.0	0.0135
4.0	29.0	woollen, clouds_wind, dungeness, regret, mood,...	32.0	0.0160

Question 3

In [264…
```python
q3_res = pd.DataFrame({'Model' : ['LDA', 'Groupped LDA'],
                       'Num Topics': [results.iloc[4]['Num Topics'],
                                      results_new.iloc[8]['Num Topics']],
                       'Coherence': [results.iloc[4]['Coherence'], results_new.iloc[8]['Coherence']],
                       'Perplexity': [results.iloc[4]['Perplexity'], results_new.iloc[8]['Perplexity']],
                       'KL-Divergence': [results.iloc[4]['KL Divergence'], results_new.iloc[8]['KL Divergence']]})
q3_res
```

Out[264…

	Model	Num Topics	Coherence	Perplexity	KL-Divergence
0	LDA	20.0	-7.355326	1361.981055	5.344533
1	Groupped LDA	50.0	-1.576982	650.360994	2.161757

In [ ]: