Final Project

data 440

Abby Gaddi

Creating the Corpus

In [1]:
```
import glob
import pandas as pd

path = '/Users/abby/Desktop/DS-SSH/'
data_folder = path + 'Converted sessions/'
clean_data = path + 'csv_sessions/'

files = glob.glob(data_folder+'*Ses*')
files.sort()
```

In [6]:
```
for file in files:
    # files within each session
    file_list = glob.glob(file+'/*.txt')
    file_list.sort()

    # extract speech txt and also create columns based on filename
    data = []
    for speech in file_list:
        with open(speech) as f:
            info = speech.split('/')[-1].replace('.txt','').split('_')
            code = info[0]
            session = info[1]
            year = info[2]
            data.append([code, session, year, f.read()])
    df = pd.DataFrame(data, columns = ['code', 'session', 'year', 'statement'])

    # create csv by session
    df.to_csv(clean_data + 'session' + session+'.csv', index = False)
```

In [11]:
```
# create the full dataframe
sessions = glob.glob(clean_data+'*ses*')
sessions.sort()

df = pd.DataFrame()
for sesh in sessions:
    #csv = sesh.split('/')[-1]
    text = pd.read_csv(sesh)
    df = df.append(text)
```

In [12]:
```
df.head()
```

Out[12]:

	code	session	year	statement
0	ALB	25	1970	33: May I first convey to our President the co...
1	ARG	25	1970	177.\t : It is a fortunate coincidence that pr...
2	AUS	25	1970	100.\t It is a pleasure for me to extend to y...
3	AUT	25	1970	155.\t May I begin by expressing to Ambassado...
4	BEL	25	1970	176. No doubt each of us, before coming up to ...

In [9]:
```
# import the economic class csv from World Bank
ec_class = pd.read_csv(path + 'class.csv', index_col = 0)
```

In [10]:
```
ec_class.head()
```

Out[10]:

	Economy	Code	Income group
1	Afghanistan	AFG	Low income
2	Albania	ALB	Upper middle income
3	Algeria	DZA	Lower middle income
4	American Samoa	ASM	Upper middle income
5	Andorra	AND	High income

```
In [17]: # rename column for easy merging
         df.rename(columns = {'code': 'Code'}, inplace = True)
```

```
In [18]: corpus = pd.merge(df, ec_class, on = 'Code', how = 'left')
```

```
In [19]: corpus
```

Out[19]:

	Code	session	year	statement	Economy	Income group
0	ALB	25	1970	33: May I first convey to our President the co...	Albania	Upper middle income
1	ARG	25	1970	177.\t : It is a fortunate coincidence that pr...	Argentina	Upper middle income
2	AUS	25	1970	100.\t It is a pleasure for me to extend to y...	Australia	High income
3	AUT	25	1970	155.\t May I begin by expressing to Ambassado...	Austria	High income
4	BEL	25	1970	176. No doubt each of us, before coming up to ...	Belgium	High income
...	...	...	...	...	...	...
8088	WSM	73	2018	I have had the privilege of addressing the G...	Samoa	Upper middle income
8089	YEM	73	2018	On behalf of the Government and the people of ...	Yemen, Rep.	Low income
8090	ZAF	73	2018	I have the honour to address the General Assem...	South Africa	Upper middle income
8091	ZMB	73	2018	Let me join other world leaders in con...	Zambia	Lower middle income
8092	ZWE	73	2018	It is my honour and pleasure to deliver my mai...	Zimbabwe	Lower middle income

8093 rows × 6 columns

```
In [20]: corpus.isna().sum()
```

```
Out[20]: Code              0
         session           0
         year              0
         statement         0
         Economy         110
         Income group    110
         dtype: int64
```

```
In [21]: corpus[corpus['Income group'].isnull()]['Code'].value_counts()
```

```
Out[21]: YUG     27
         CSK     22
         YDYE    19
         DDR     18
         VAT     16
         EU       8
         Name: Code, dtype: int64
```

Since some nations have changed, they do not appear on the world bank economy group for 2020. In the dataset with country names and codes alongside the UN data, in 200
Serbia. Also, Czechoslovakia (CSK) is now Czech and Slovakia respectively. Both are classified high income. Democratic Yemen (YDYE), German Democratic Republic (DDR), t
Union (ED) were kept in the main topic model but removed when subsetting the economy groups since they had no match.

```
In [23]: for index, row in corpus[corpus['Code'] == 'YUG'].iterrows():
             if corpus.iloc[index, 1] >2002:
                 corpus.loc[index, 'Income group'] = 'Upper middle income'
```

```
In [24]: for index, row in corpus[corpus['Code'] == 'CSK'].iterrows():
              corpus.loc[index, 'Income group'] = 'High income'
```

```
In [25]: corpus.isna().sum()
```

```
Out[25]: Code              0
         session           0
         year              0
         statement         0
         Economy         110
         Income group     88
         dtype: int64
```

```
In [26]: corpus.to_csv(path + 'corpus.csv', index = False)
```

Clean & Summarize

In [11]:
```
import nltk
import re
import pycountry
```

In [27]:
```
corpus = pd.read_csv('corpus.csv')
```

In [28]:
```
# check if loaded correctly
corpus.head()
```

Out[28]:

	Code	session	year	statement	Economy	Income group
0	ALB	25	1970	33: May I first convey to our President the co...	Albania	Upper middle income
1	ARG	25	1970	177.\t : It is a fortunate coincidence that pr...	Argentina	Upper middle income
2	AUS	25	1970	100.\t It is a pleasure for me to extend to y...	Australia	High income
3	AUT	25	1970	155.\t May I begin by expressing to Ambassado...	Austria	High income
4	BEL	25	1970	176. No doubt each of us, before coming up to ...	Belgium	High income

In [13]:
```
def get_summary(text):
    """function that takes the statement from UNGA speech
       then returns a summary with the 50 highest scoring sentences"""

    # remove country names
    for country in pycountry.countries:
        if country.name in text:
            text = text.replace(country.name, '')

    # remove characters
    text = re.sub(r'\[[0-9]*\]', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    ftext = re.sub('[^a-zA-Z]', ' ', text )
    ftext = re.sub(r'\s+', ' ', ftext)

    # Tokenize
    sentence_list = nltk.sent_tokenize(text)

    # Get all stopwords for removal
    stopwords = nltk.corpus.stopwords.words('english')

    # Empty dictionary for storing frequencies
    word_frequencies = {}

    # Loop through words
    # if not in stopwords list, check if it's in the dictionary, update count by 1

    for word in nltk.word_tokenize(ftext):
        if word not in stopwords:
            if word not in word_frequencies.keys():
                word_frequencies[word] = 1
            else:
                word_frequencies[word] += 1
    # Create an empty dictionary to store scores
    sentence_scores = {}

    # loop through each sentence
    # go through each word in the sentence
    # get the frequency in the word's dictionary key
    # add the word frequency to the sentence score

    for sent in sentence_list:
        for word in nltk.word_tokenize(sent.lower()):
            if word in word_frequencies.keys():
                    if sent in sentence_scores.keys():
                        sentence_scores[sent] += word_frequencies[word]
                    else:
                        sentence_scores[sent] = word_frequencies[word]

    nrsentences = 50
    summary_sentences = [x[0] for x in sorted(sentence_scores.items(),key=lambda x: x[1], reverse=True)[:nrsentences]]
    summary = ' '.join(summary_sentences)
    return summary
```

In [14]:
```
summaries = []
for row, statement in corpus['statement'].iteritems():
    summaries.append(get_summary(statement))
```

```
In [16]: corpus['summary'] = summaries
         corpus.head()
```

Out[16]:

	Code	session	year	statement	Economy	Income group	summary
0	ALB	25	1970	33: May I first convey to our President the co...	Albania	Upper middle income	This is shown by the struggle of the heroic pe...
1	ARG	25	1970	177.\t : It is a fortunate coincidence that pr...	Argentina	Upper middle income	If we consider the consequences of the qualita...
2	AUS	25	1970	100.\t It is a pleasure for me to extend to y...	Australia	High income	It was an unhappy and disturbed world in which...
3	AUT	25	1970	155.\t May I begin by expressing to Ambassado...	Austria	High income	The twenty-fifth anniversary of the United Nat...
4	BEL	25	1970	176. No doubt each of us, before coming up to ...	Belgium	High income	We therefore attach the greatest importance to...

```
In [17]: # updated corpus
         corpus.to_csv(path + 'corpus.csv', index = False)
```

Splitting corpus into economy groups

```
In [18]: corpus['Income group'].value_counts()

Out[18]: High income            2519
         Upper middle income    2160
         Lower middle income    2086
         Low income             1240
         Name: Income group, dtype: int64
```

```
In [19]: # create subsets
         high = corpus[corpus['Income group'] == 'High income']
         upper = corpus[corpus['Income group'] == 'Upper middle income']
         lower = corpus[corpus['Income group'] == 'Lower middle income']
         low = corpus[corpus['Income group'] == 'Low income']
```

```
In [20]: # save to csv to separate folders
         high.to_csv(path + 'high/' + 'high.csv', index = False)
         upper.to_csv(path + 'upper/' + 'upper.csv', index = False)
         lower.to_csv(path + 'lower/' + 'lower.csv', index = False)
         low.to_csv(path + 'low/' + 'low.csv', index = False)
```

Topic Models

Each model follows the same code; however, I did not run the code that creates the LDA model itself since I did those in a different jupyter notebook. The visualizations were cr
made models.

```
In [22]: # import libraries
         # Import topic modeling modules
         from gensim import models
         from gensim.corpora import Dictionary, MmCorpus
         from gensim.test.utils import datapath

         # Import visualization modules
         import pyLDAvis.gensim as gensimvis
         import pyLDAvis

         import wordcloud

         %matplotlib inline
         import matplotlib
         import matplotlib.pyplot as plt

         # Deprecation Warnings kept showing up in every single cell
         # used ignore warning to stop most, not all
         import warnings;
         warnings.filterwarnings('ignore');
```

```
In [23]: def filter_corpus(filename, textcol=1, filter_string='', makelower=True):
             """Import corpus (from csv in id, text format), filtering texts as requested."""
             import csv
             csv.field_size_limit(1000000000)
             docs, textids = [], []
             with open(filename, 'r') as infile:
                 for row in csv.reader(infile):
                     text = row[textcol]
                     if len(filter_string) == 0 or filter_string in text:
                         docs.append(text.lower() if makelower else text)
                         textids.append(row[0])
             return docs, textids
```

In [24]:
```
def prep_corpus(docs, additional_stopwords=set(),
                no_below=5, no_above=0.5):
    """Prepare corpus: generate gensim-style dictionary & corpus formats.

    Also strip stopwords and remove very (un)common words.
    """
    print('Building dictionary...')
    doctokens = [[x for x in doc.split() if len(x) > 1 or x.lower() == 'i']
                 for doc in docs]
    corpusdict = Dictionary(doctokens)

    stopwords = set(nltk.corpus.stopwords.words('english')).union(additional_stopwords)
    stopword_ids = map(corpusdict.token2id.get, stopwords)

    corpusdict.filter_tokens(stopword_ids)
    # corpusdict.compactify()
    corpusdict.filter_extremes(no_below=no_below, no_above=no_above, keep_n=None)
    corpusdict.compactify()

    print('Building corpus...')
    corpus = [corpusdict.doc2bow(doc) for doc in doctokens]

    return corpusdict, corpus
```

1. General Topic Model

In [25]:
```
corpusroot = path
corpusfile = corpusroot + 'corpus.csv'
```

In [26]:
```
# create key filter string
# using 'the' in order to capture ALL summarised statements
filter_string = 'the'

# Filenames for gensim-format corpus
corpus_mm = corpusroot + filter_string + '.mm'
corpus_dict = corpusroot + filter_string + '.dict'
```

In [27]:
```
# Load the corpus, keeping only those texts containing the filter string.
# Display the number of texts retained, to make sure it is a reasonable number
# (for the RTD analysis, if it is less than 1000, pick something else).

docs, textids = filter_corpus(corpusfile, textcol = 6, filter_string=filter_string)
len(docs)
```

Out[27]:
```
8093
```

In [28]:
```
dictionary, corpus = prep_corpus(docs)
```

```
Building dictionary...
Building corpus...
```

In [29]:
```
%%time

nrtopics = 5  # experiment with this number to see what produces good topics

lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
                               num_topics=nrtopics, passes=10)
lda.save(corpusroot + '_' + str(nrtopics) + '_lda.model')
```

```
CPU times: user 3min 29s, sys: 392 ms, total: 3min 30s
Wall time: 1min 46s
```

In [30]:
```
# Extract top words for each topic

nrwords_wordcloud = 50
topinfo = lda.show_topics(num_topics=nrtopics, num_words=nrwords_wordcloud, formatted=False)
topic_wordsweights = [topdata[1] for topdata in topinfo]

nrwords_plaintext = 12
topic_keywords = [' '.join([wordinfo[0] for wordinfo in topdata[1][:nrwords_plaintext]])
                  for topdata in topinfo]
```

Displaying top words as Strings

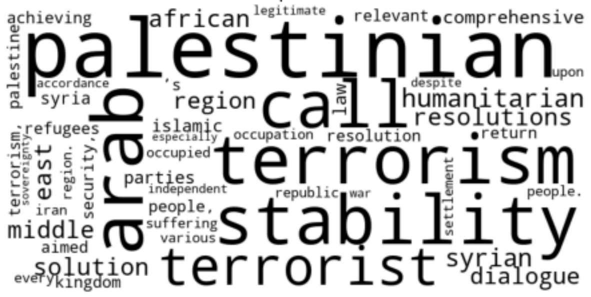
```
In [31]: # display topic keywords, separated by blank lines
         for topic_words in topic_keywords:
             print(topic_words)
             print()
```

```
palestinian arab terrorism stability call terrorist humanitarian syrian solution african resolutions east

republic democratic european law implementation conflict reform resolution stability humanitarian nuclear agreement

sustainable climate challenges change agenda small commitment address goals african island reform

nuclear relations co-operation republic foreign solution delegation present policy measures military independence

every per better today president democracy that, see together poverty let much
```

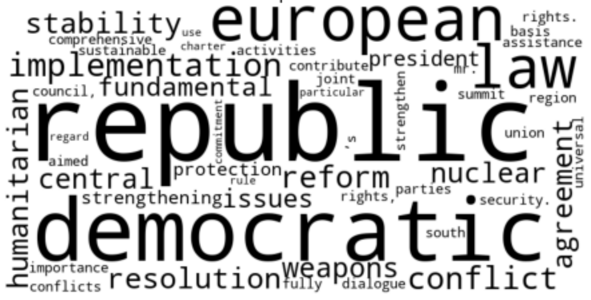
Displaying top words as word clouds

In [32]:
```
for topic_nr, topic in enumerate(topic_wordsweights):

    # Convert top n words and associated probabilities to a dictionary,
    topwords = {word: weight for word, weight in topic}

    # Set up the plot
    plt.figure(figsize=(8,4))
    plt.imshow(wordcloud.WordCloud(width=800, height=400,
                                   background_color='white',
                                   color_func=lambda *args, **kwargs: 'black').fit_words(topwords))
    plt.axis("off")
    plt.title('Topic #{}'.format(topic_nr))
```

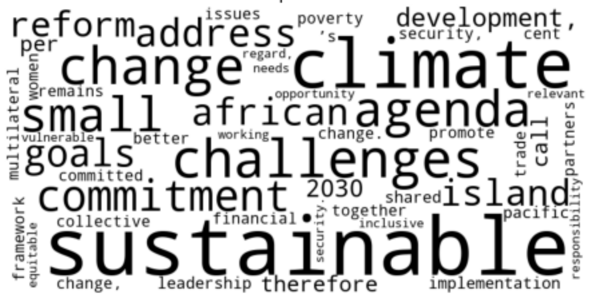
Topic #0



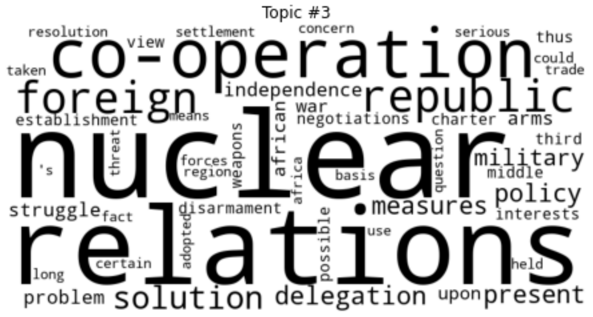
Topic #1



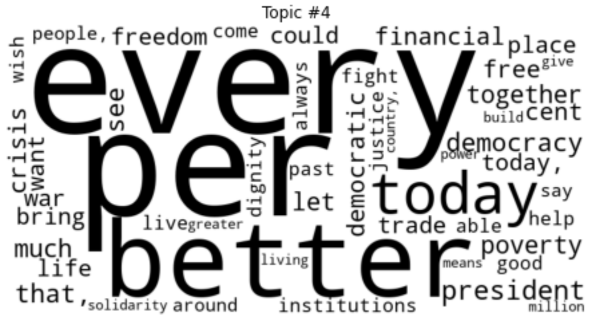
Topic #2



Topic #3



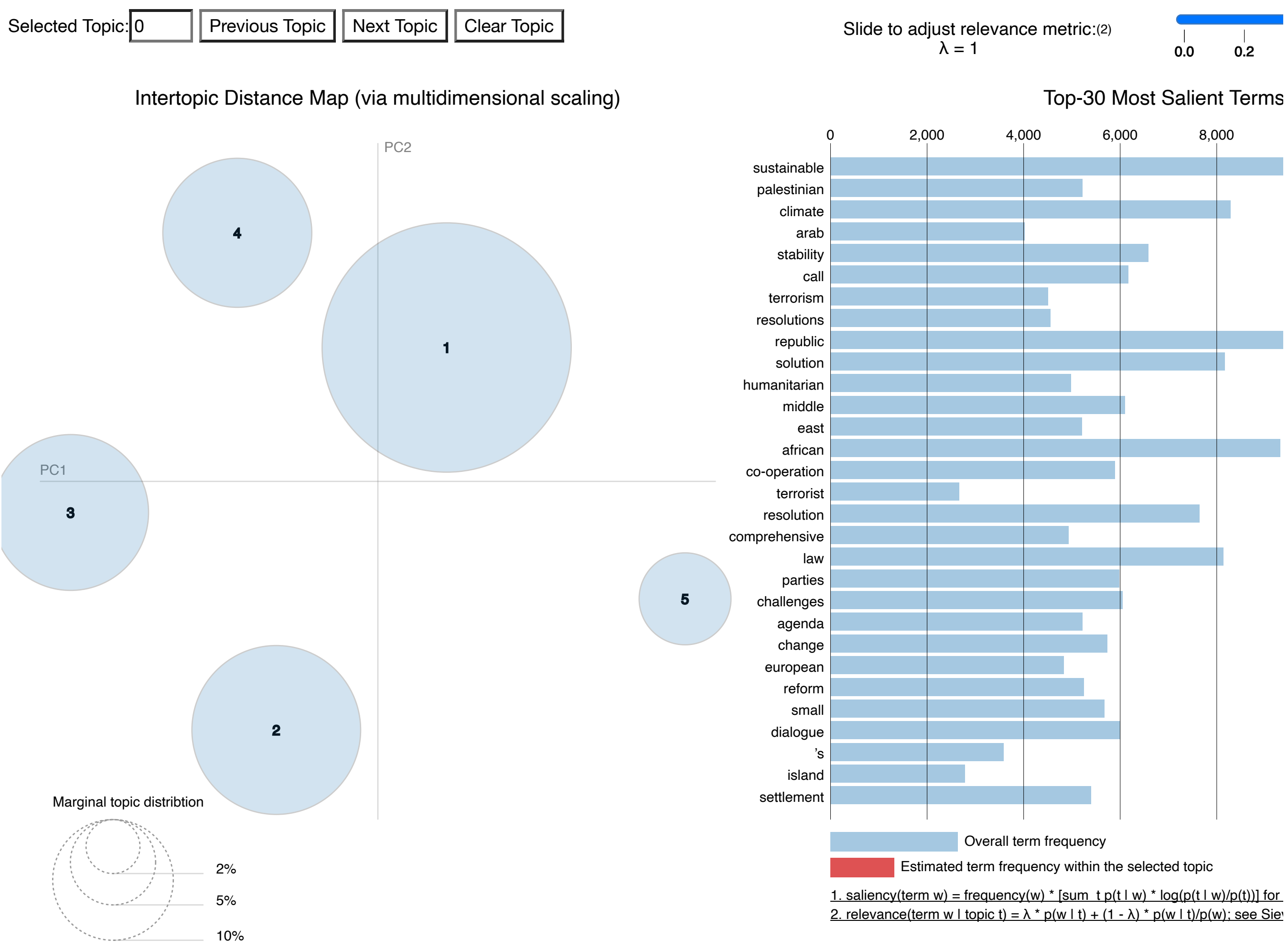
Topic #4



Dynamic Visualization

In [33]:
```
nrtopics = 5  # make sure this number matches the version you want to load!
lda = models.ldamodel.LdaModel.load(corpusroot + '_' + str(nrtopics) + '_lda.model')

lda_vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')
pyLDAvis.display(lda_vis)
```

Out[33]:



2. High income

In [34]:
```
corpusroot = path + 'high/'
corpusfile = corpusroot + 'high.csv'
```

In [35]:
```
# create key filter string
filter_string = 'development'

# Filenames for gensim-format corpus
corpus_mm = corpusroot + filter_string + '.mm'
corpus_dict = corpusroot + filter_string + '.dict'
```

In [36]:
```
# Load the corpus, keeping only those texts containing the filter string.
# Display the number of texts retained, to make sure it is a reasonable number
# (for the RTD analysis, if it is less than 1000, pick something else).

docs, textids = filter_corpus(corpusfile, textcol = 6, filter_string=filter_string)
len(docs)
```

Out[36]:
```
2305
```

In [37]:
```
dictionary, corpus = prep_corpus(docs)
```

```
Building dictionary...
Building corpus...
```

In [40]:
```
#%%time

#nrtopics = 3  # experiment with this number to see what produces good topics

#lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
#                                num_topics=nrtopics, passes=10)
#lda.save(corpusroot + '_' + str(nrtopics) + '_lda.model')
```

In [44]:
```
# Extract top words for each topic

nrwords_wordcloud = 50
topinfo = lda.show_topics(num_topics=nrtopics, num_words=nrwords_wordcloud, formatted=False)
topic_wordsweights = [topdata[1] for topdata in topinfo]

nrwords_plaintext = 12
topic_keywords = [' '.join([wordinfo[0] for wordinfo in topdata[1][:nrwords_plaintext]])
                  for topdata in topinfo]
```

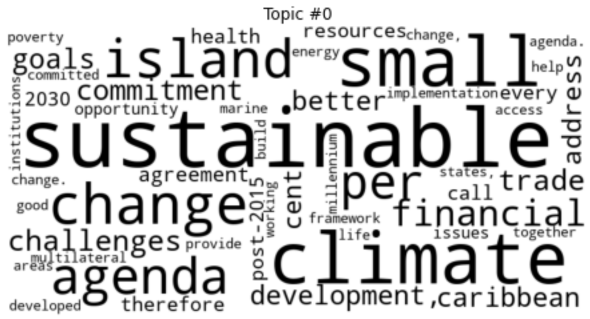
Displaying top words as Strings

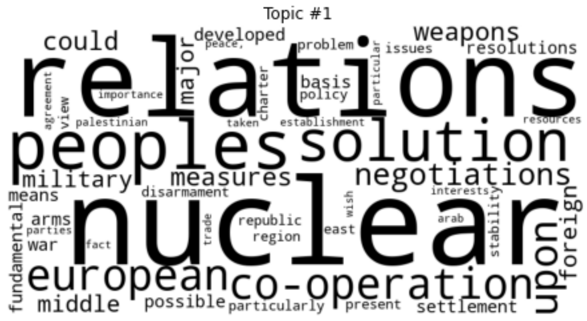
In [45]:
```
# display topic keywords, separated by blank lines
for topic_words in topic_keywords:
    print(topic_words)
    print()
```

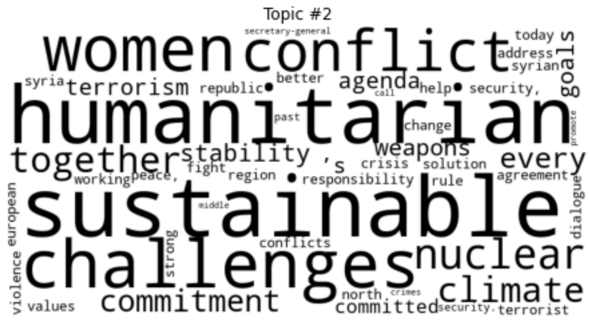
```
sustainable climate small change island agenda per financial challenges trade commitment goals

nuclear relations peoples solution co-operation european upon negotiations weapons could measures military

sustainable humanitarian challenges women conflict nuclear together climate commitment every stability goals
```

Displaying top words as word clouds

```
In [46]: for topic_nr, topic in enumerate(topic_wordsweights):

             # Convert top n words and associated probabilities to a dictionary,
             topwords = {word: weight for word, weight in topic}

             # Set up the plot
             plt.figure(figsize=(8,4))
             plt.imshow(wordcloud.WordCloud(width=800, height=400,
                                            background_color='white',
                                            color_func=lambda *args, **kwargs: 'black').fit_words(topwords))
             plt.axis("off")
             plt.title('Topic #{}'.format(topic_nr))
```



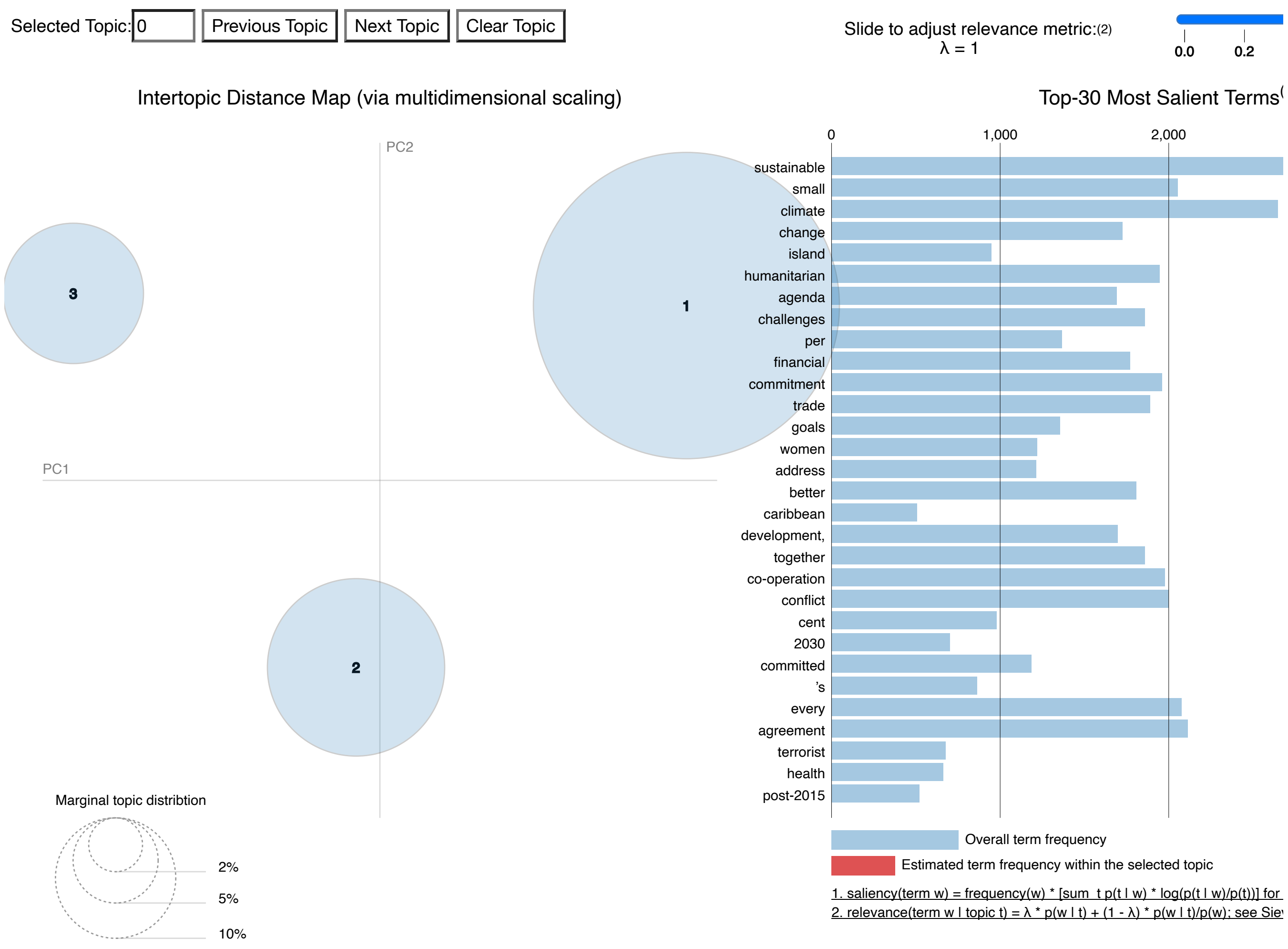




Dynamic Visualization

In [43]:
```
nrtopics = 3  # make sure this number matches the version you want to load!
lda = models.ldamodel.LdaModel.load(corpusroot + '_' + str(nrtopics) + '_lda.model')

lda_vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')
pyLDAvis.display(lda_vis)
```

Out[43]:



3. Upper middle income

In [47]:
```
corpusroot = path + 'upper/'
corpusfile = corpusroot + 'upper.csv'
```

In [48]:
```
# create key filter string
filter_string = 'development'

# Filenames for gensim-format corpus
corpus_mm = corpusroot + filter_string + '.mm'
corpus_dict = corpusroot + filter_string + '.dict'
```

In [49]:
```
# Load the corpus, keeping only those texts containing the filter string.
# Display the number of texts retained, to make sure it is a reasonable number
# (for the RTD analysis, if it is less than 1000, pick something else).

docs, textids = filter_corpus(corpusfile, textcol = 6, filter_string=filter_string)
len(docs)
```

Out[49]:
```
2035
```

In [50]:
```
dictionary, corpus = prep_corpus(docs)
```

```
Building dictionary...
Building corpus...
```

In [ ]:
```
#%%time

#nrtopics = 3  # experiment with this number to see what produces good topics

#lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
#                               num_topics=nrtopics, passes=10)
#lda.save(corpusroot + '_' + str(nrtopics) + '_lda.model')
```

In [53]:
```
# Extract top words for each topic

nrwords_wordcloud = 50
topinfo = lda.show_topics(num_topics=nrtopics, num_words=nrwords_wordcloud, formatted=False)
topic_wordsweights = [topdata[1] for topdata in topinfo]

nrwords_plaintext = 12
topic_keywords = [' '.join([wordinfo[0] for wordinfo in topdata[1][:nrwords_plaintext]])
                  for topdata in topinfo]
```

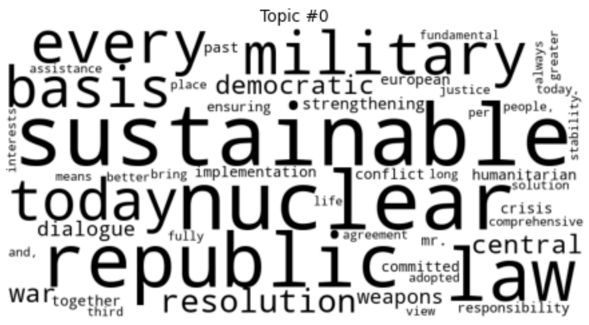
Displaying top words as Strings

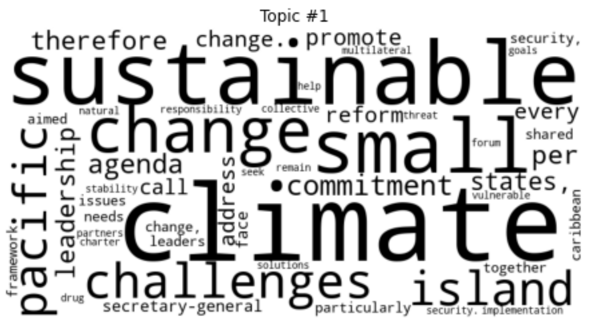
In [54]:
```
# display topic keywords, separated by blank lines
for topic_words in topic_keywords:
    print(topic_words)
    print()
```

```
sustainable nuclear republic law military every today basis resolution democratic war central

climate sustainable small change challenges island pacific commitment agenda per therefore leadership
```

Displaying top words as word clouds

In [55]:
```
for topic_nr, topic in enumerate(topic_wordsweights):

    # Convert top n words and associated probabilities to a dictionary,
    topwords = {word: weight for word, weight in topic}

    # Set up the plot
    plt.figure(figsize=(8,4))
    plt.imshow(wordcloud.WordCloud(width=800, height=400,
                                   background_color='white',
                                   color_func=lambda *args, **kwargs: 'black').fit_words(topwords))
    plt.axis("off")
    plt.title('Topic #{}'.format(topic_nr))
```

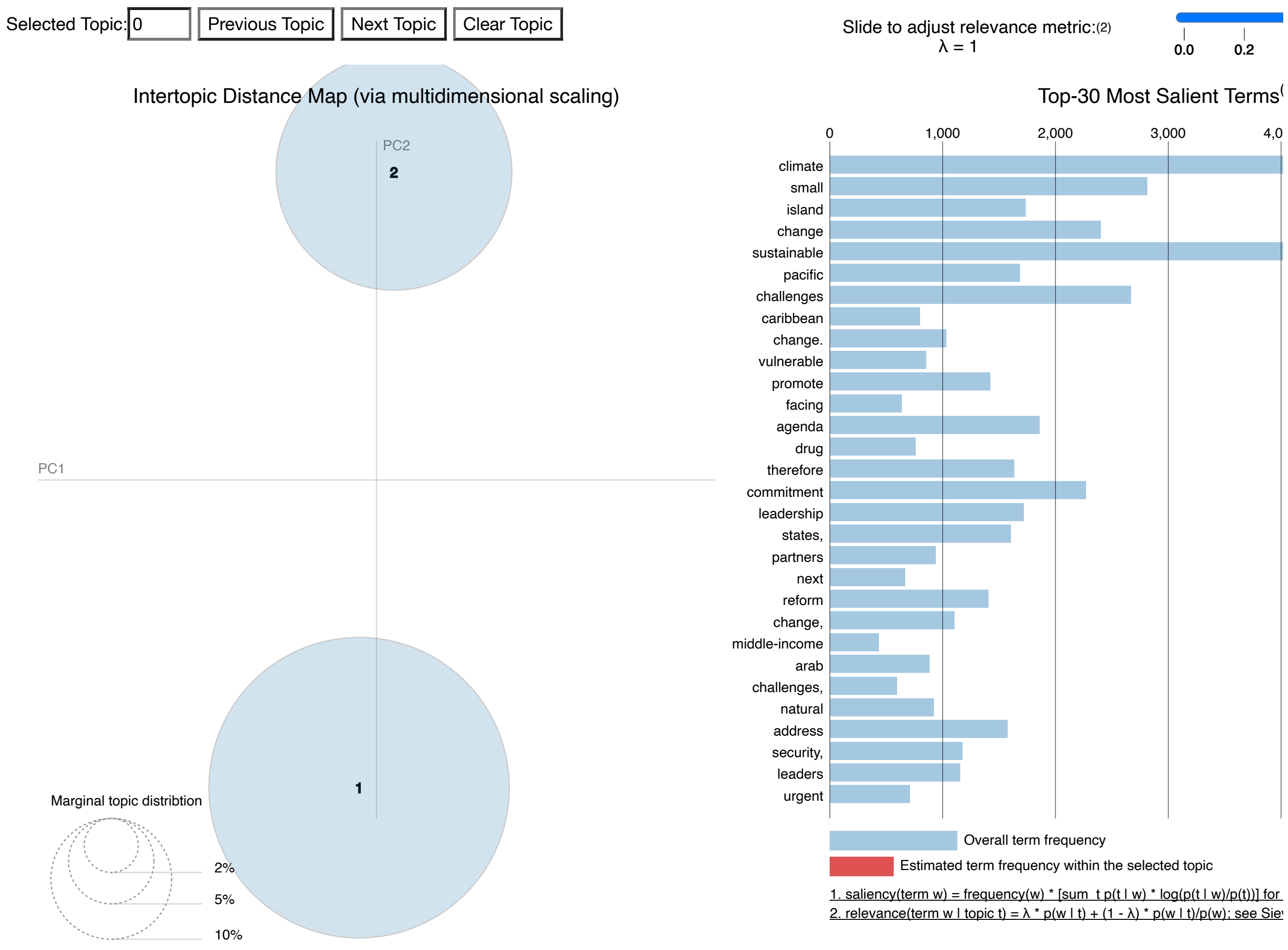




Dynamic Visualization

In [52]:
```
nrtopics = 2  # make sure this number matches the version you want to load!
lda = models.ldamodel.LdaModel.load(corpusroot + '_' + str(nrtopics) + '_lda.model')

lda_vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')
pyLDAvis.display(lda_vis)
```

Out[52]:



4. Lower middle income

In [56]:
```
corpusroot = path + 'lower/'
corpusfile = corpusroot + 'lower.csv'
```

In [57]:
```
# create key filter string
filter_string = 'development'

# Filenames for gensim-format corpus
corpus_mm = corpusroot + filter_string + '.mm'
corpus_dict = corpusroot + filter_string + '.dict'
```

In [58]:
```
# Load the corpus, keeping only those texts containing the filter string.
# Display the number of texts retained, to make sure it is a reasonable number
# (for the RTD analysis, if it is less than 1000, pick something else).

docs, textids = filter_corpus(corpusfile, textcol = 6, filter_string=filter_string)
len(docs)
```

Out[58]:
```
1979
```

In [59]:
```
dictionary, corpus = prep_corpus(docs)
```

```
Building dictionary...
Building corpus...
```

```
In [ ]: #%%time

        #nrtopics = 3  # experiment with this number to see what produces good topics

        #lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
        #                                num_topics=nrtopics, passes=10)
        #lda.save(corpusroot + '_' + str(nrtopics) + '_lda.model')
```

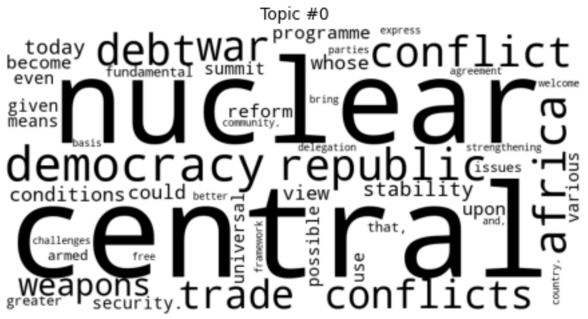
```
In [61]: # Extract top words for each topic

         nrwords_wordcloud = 50
         topinfo = lda.show_topics(num_topics=nrtopics, num_words=nrwords_wordcloud, formatted=False)
         topic_wordsweights = [topdata[1] for topdata in topinfo]

         nrwords_plaintext = 12
         topic_keywords = [' '.join([wordinfo[0] for wordinfo in topdata[1][:nrwords_plaintext]])
                           for topdata in topinfo]
```

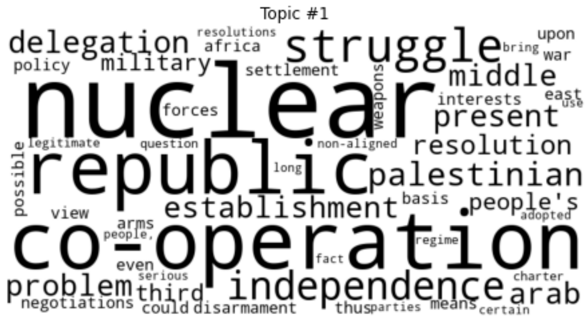
Displaying top words as Strings

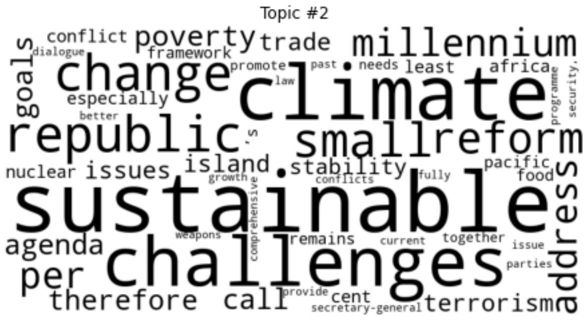
```
In [62]: # display topic keywords, separated by blank lines
         for topic_words in topic_keywords:
             print(topic_words)
             print()

         central nuclear democracy africa republic debt conflict war trade conflicts weapons stability

         nuclear co-operation republic struggle independence palestinian delegation present middle problem establishment arab

         sustainable climate challenges republic change small reform millennium address per poverty agenda
```

Displaying top words as word clouds

```
In [63]: for topic_nr, topic in enumerate(topic_wordsweights):

             # Convert top n words and associated probabilities to a dictionary,
             topwords = {word: weight for word, weight in topic}

             # Set up the plot
             plt.figure(figsize=(8,4))
             plt.imshow(wordcloud.WordCloud(width=800, height=400,
                                            background_color='white',
                                            color_func=lambda *args, **kwargs: 'black').fit_words(topwords))
             plt.axis("off")
             plt.title('Topic #{}'.format(topic_nr))
```



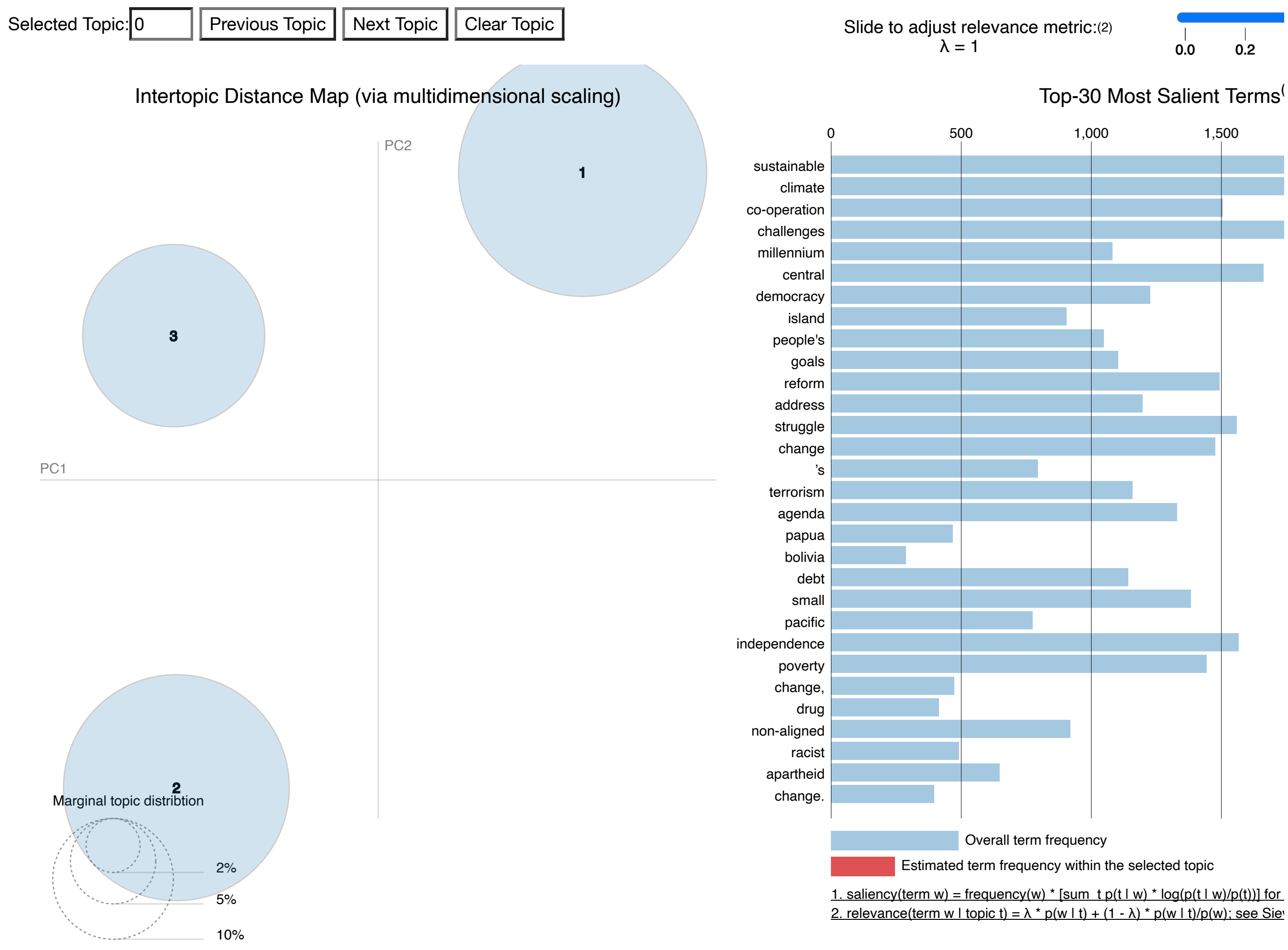




Dynamic Visualization

In [60]:
```
nrtopics = 3  # make sure this number matches the version you want to load!
lda = models.ldamodel.LdaModel.load(corpusroot + '_' + str(nrtopics) + '_lda.model')

lda_vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')
pyLDAvis.display(lda_vis)
```

Out[60]:



5. Low income

In [64]:
```
corpusroot = path + 'low/'
corpusfile = corpusroot + 'low.csv'
```

In [65]:
```
# create key filter string
filter_string = 'development'

# Filenames for gensim-format corpus
corpus_mm = corpusroot + filter_string + '.mm'
corpus_dict = corpusroot + filter_string + '.dict'
```

In [66]:
```
# Load the corpus, keeping only those texts containing the filter string.
# Display the number of texts retained, to make sure it is a reasonable number
# (for the RTD analysis, if it is less than 1000, pick something else).

docs, textids = filter_corpus(corpusfile, textcol = 6, filter_string=filter_string)
len(docs)
```

Out[66]:
```
1177
```

In [67]:
```
dictionary, corpus = prep_corpus(docs)
```

```
Building dictionary...
Building corpus...
```

In [ ]:
```
#%%time

#nrtopics = 3  # experiment with this number to see what produces good topics

#lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
#                               num_topics=nrtopics, passes=10)
#lda.save(corpusroot + '_' + str(nrtopics) + '_lda.model')
```

In [69]:
```
# Extract top words for each topic

nrwords_wordcloud = 50
topinfo = lda.show_topics(num_topics=nrtopics, num_words=nrwords_wordcloud, formatted=False)
topic_wordsweights = [topdata[1] for topdata in topinfo]

nrwords_plaintext = 12
topic_keywords = [' '.join([wordinfo[0] for wordinfo in topdata[1][:nrwords_plaintext]])
                  for topdata in topinfo]
```

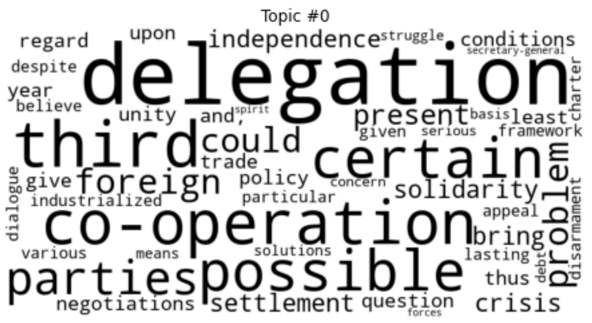
Displaying top words as Strings

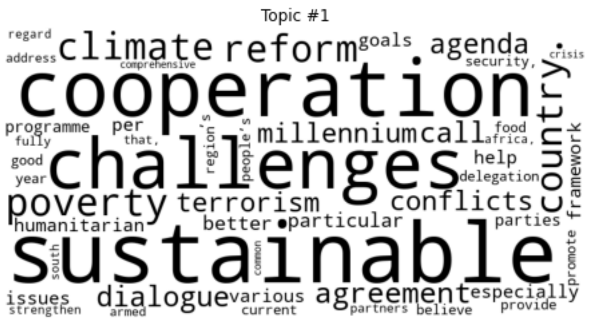
In [70]:
```
# display topic keywords, separated by blank lines
for topic_words in topic_keywords:
    print(topic_words)
    print()
```

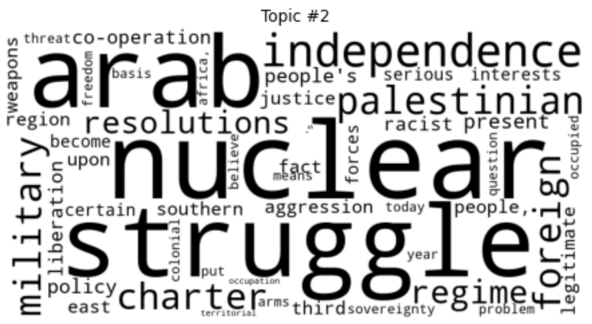
```
delegation third co-operation possible certain parties foreign problem could present solidarity bring

sustainable cooperation challenges poverty country. climate reform agreement dialogue call agenda millennium

nuclear struggle arab independence palestinian military foreign charter regime resolutions present policy
```

Displaying top words as word clouds

```
In [71]: for topic_nr, topic in enumerate(topic_wordsweights):

             # Convert top n words and associated probabilities to a dictionary,
             topwords = {word: weight for word, weight in topic}

             # Set up the plot
             plt.figure(figsize=(8,4))
             plt.imshow(wordcloud.WordCloud(width=800, height=400,
                                            background_color='white',
                                            color_func=lambda *args, **kwargs: 'black').fit_words(topwords))
             plt.axis("off")
             plt.title('Topic #{}'.format(topic_nr))
```



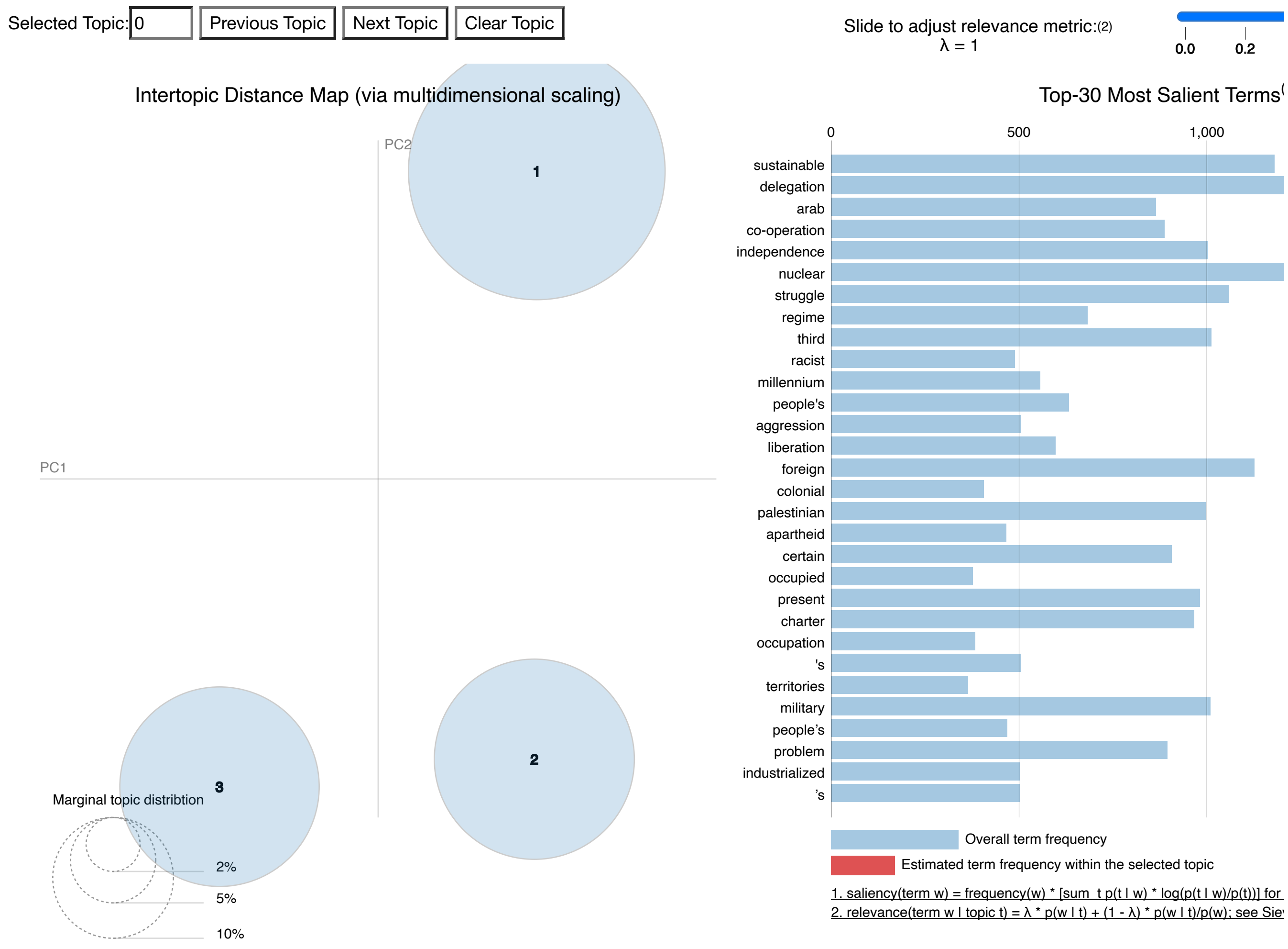




Dynamic Visualization

In [68]:
```
nrtopics = 3  # make sure this number matches the version you want to load!
lda = models.ldamodel.LdaModel.load(corpusroot + '_' + str(nrtopics) + '_lda.model')

lda_vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')
pyLDAvis.display(lda_vis)
```

Out[68]:



In [ ]: