CPE 352 Data Science

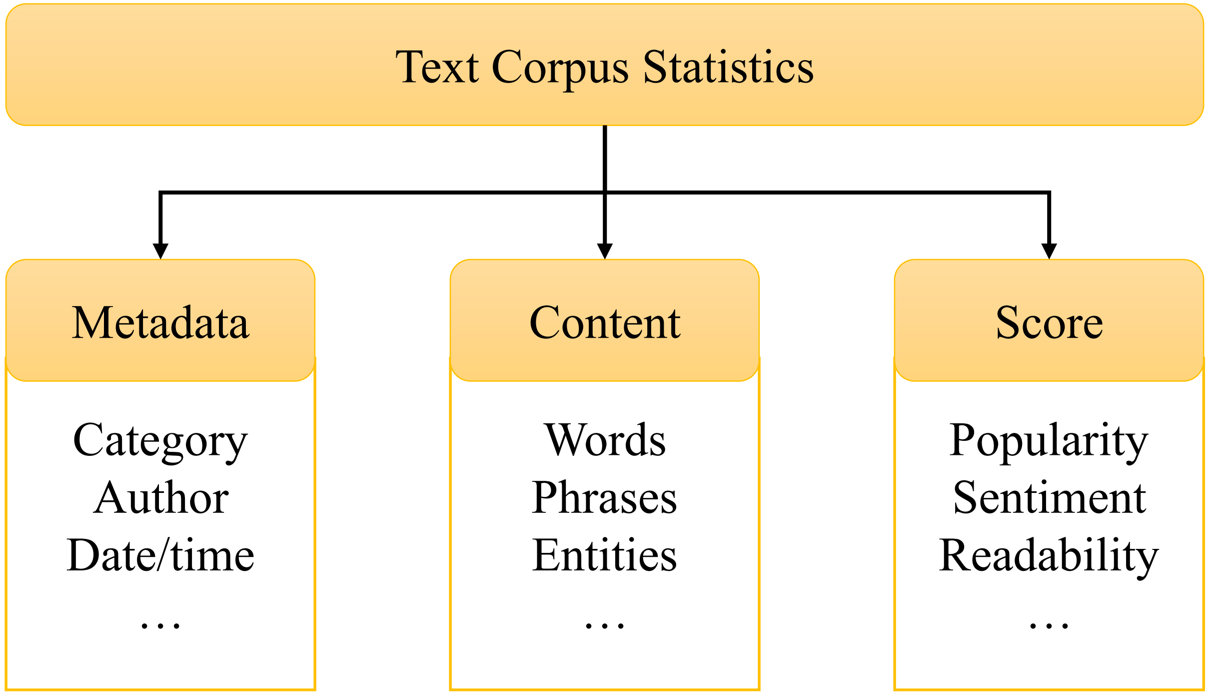
4 – Textual Data

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

Text EDA

- Exploratory data analysis (EDA) is a process of systematically examining data on an aggregated level



Dataset
UN General Debate Dataset

- 7,507 speeches held at the annual sessions of the United Nations General Assembly from 1970 to 2016.
- Created by Mikhaylov, Baturo and Dasandi at Havard
- The purpose is for understanding and measuring state preferences in world politics

Loading dataset

```
import pandas as pd
```

```
df = pd.read_csv('un-general-debates-blueprint.csv.gz')
```

```
df.sample(3)
```

	session	year	country	country_name	speaker	position	text
4578	55	2000	NER	Niger	Ousmane Moutari	UN Representative	wish at\nthe very outset to convey to the cou...
5070	58	2003	FJI	Fiji	Keliopate Tavola	Minister for Foreign Affairs	Mr. President, my Government\nand country war...
1601	37	1982	HTI	Haiti	ESTIME	NaN	\nOn behalf of the Government of Haiti and in ...

index

What are primary keys?

Getting an overview of data

What to look for?

- Point measurements (mean, standard deviation, …)
- Quality measurement (null, missing, incorrect format, ...)
- Distribution (center, skewness, shape, etc.)
- Association (relationships between distributions and factors)

Getting an overview of data

1. Create summary statistics
2. Check for missing values
3. Plot distributions of interesting attributes
4. Compare distributions across categories
5. Visualize developments over time

Pandas commands for getting information about dataframe

df.columns

List of columns names

df.dtypes

Tuples (column name, data type)

df.info()

Dtypes plus memory consumptions

df.describe()

Summary statistics

2. DataFrame summary statistics

```
df['length'] = df['text'].str.len()
```

```
df.columns
```

```
Index(['session', 'year', 'country', 'country_name', 'speaker', 'position',
       'text', 'length'],
      dtype='object')
```

```
df.dtypes
```

```
session          int64
year             int64
country         object
country_name    object
speaker         object
position        object
text            object
length           int64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7507 entries, 0 to 7506
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   session       7507 non-null   int64 
 1   year          7507 non-null   int64 
 2   country       7507 non-null   object
 3   country_name  7507 non-null   object
 4   speaker       7507 non-null   object
 5   position      4502 non-null   object
 6   text          7507 non-null   object
 7   length        7507 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 469.3+ KB
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
session	7507.0	49.610763	12.892155	25.0	39.0	51.0	61.0	70.0
year	7507.0	1994.610763	12.892155	1970.0	1984.0	1996.0	2006.0	2015.0
length	7507.0	17967.281604	7860.038463	2362.0	12077.0	16424.0	22479.5	72041.0

```
df[['country','speaker']].describe(include='O').T
```

	count	unique	top	freq
country	7507	199	NZL	46
speaker	7480	5428	Seyoum Mesfin	12

Checking for missing data

- Data can be missing if they are not properly filled
- This can occur when human operators made mistakes
- Missing data can create problems in analysis
- Missing data can occur in different scenarios
  - Data are missing for all values in rows
  - Data are missing for all values in columns
  - Data are missing for some values without patterns
  - Data are missing for some values with patterns

3. Checking for missing data

Checking

```
df.isna().sum()
```

```
session           0
year              0
country           0
country_name      0
speaker          27
position       3005
text              0
dtype: int64
```

Fixing

```
df['speaker'].fillna('unknown',inplace=True)
```

```
df.isna().sum()
```

```
session           0
year              0
country           0
country_name      0
speaker           0
position       3005
text              0
dtype: int64
```

More problems

```python
df[df['speaker'].str.contains('Bush')]['speaker'].value_counts()
```

```
George W. Bush        4
Mr. George W. Bush    2
Bush                  1
George Bush           1
Mr. George W Bush     1
Name: speaker, dtype: int64
```

George W. Bush

43rd U.S. President

georgewbush.com

George Walker Bush is an American politician and businessman who served as the 43rd president of the United States from 2001 to 2009. A member of the Republican Party, Bush previously served as the 46th governor of Texas from 1995 to 2000. Wikipedia

Born: July 6, 1946 (age 75 years), New Haven, Connecticut, United States

George H. W. Bush

41st U.S. President

George Herbert Walker Bush was an American politician, diplomat, and businessman who served as the 41st president of the United States from 1989 to 1993. Wikipedia

Born: June 12, 1924, Milton, Massachusetts, United States

Died: November 30, 2018, Houston, Texas, United States

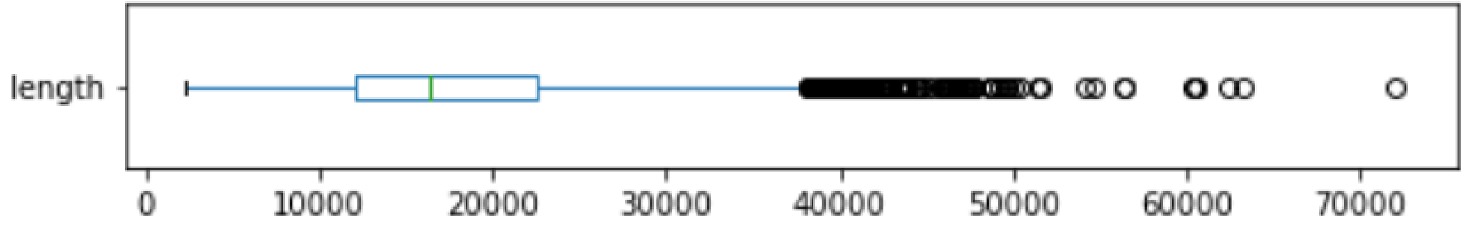
Plot distribution of interesting attributes

- Data distribution can reveal
  - Center location
  - Range
  - Spread
  - Skewness
  - Outliers
- Understanding distributions allow us to determine the nature of data and work with them accordingly

Length distribution

```
df['length'].plot(kind='box', vert=False, figsize=(8, 1))
```

```
<AxesSubplot:>
```

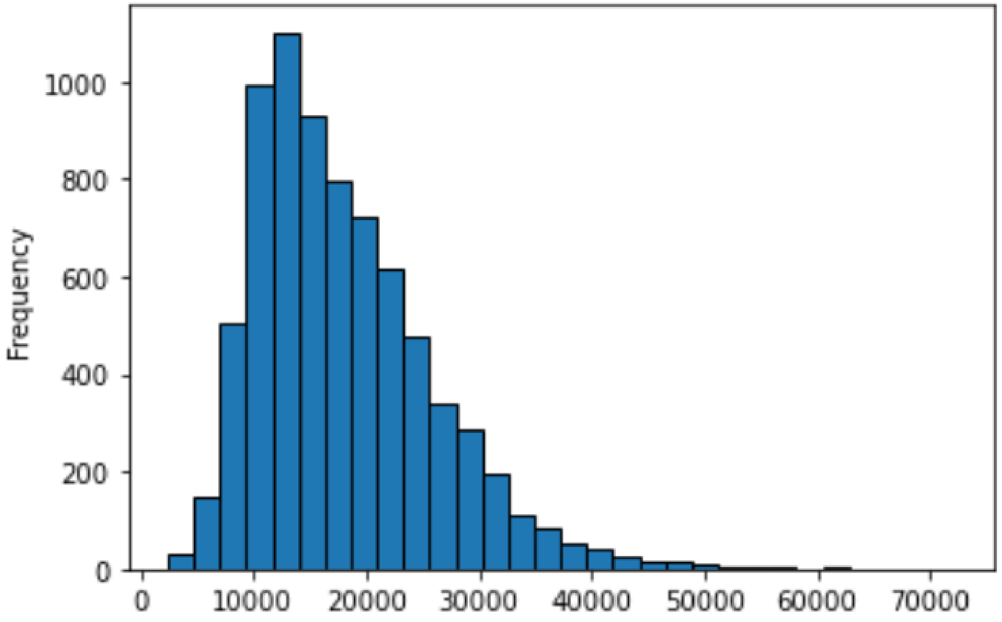


- 50% of the speeches have length between 12,000 to 22,000 characters, with the median at about 16,000 and a long tail with many outliers to the right
- The distribution is left skewed.

Length distribution (2)

```
df['length'].plot(kind='hist', bins=30, figsize=(6, 4), edgecolor='k')
```

<AxesSubplot:ylabel='Frequency'>



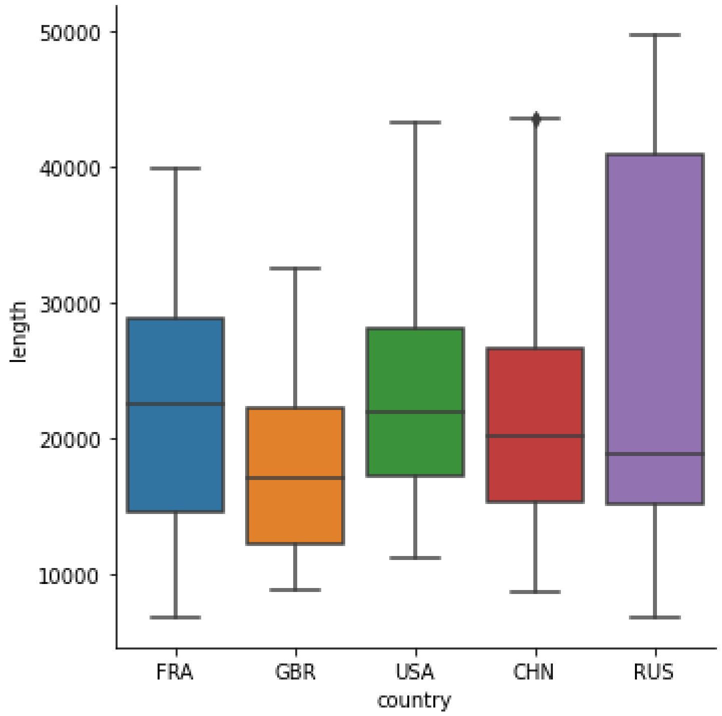
Comparing value distributions across categories

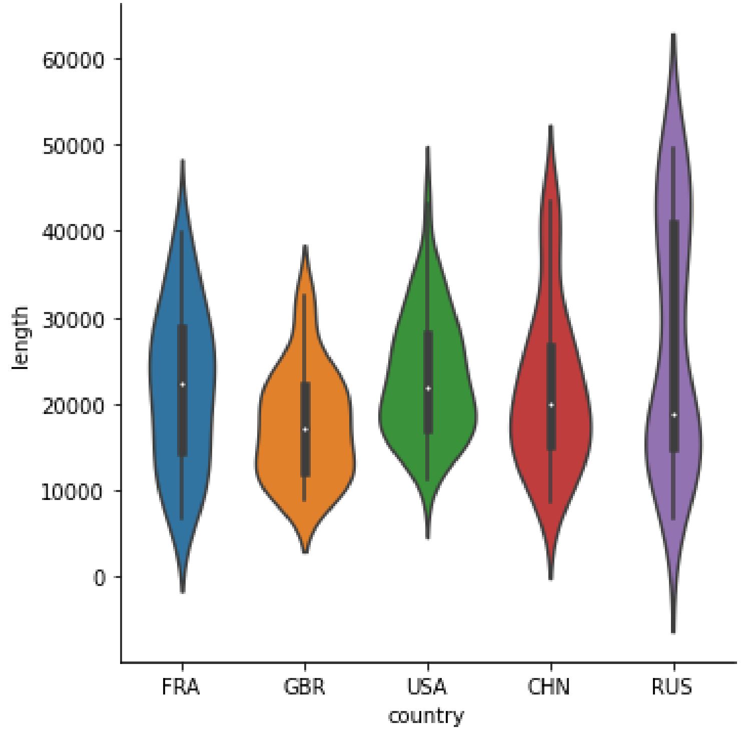
- Specific patterns may arise when observing subset of data
- Let’s look at distribution of speech length and country

5. Distributions across categories

```python
import seaborn as sns

where = df['country'].isin(['USA','FRA','GBR','CHN','RUS'])
sns.catplot(data=df[where], x='country', y='length', kind='box')
sns.catplot(data=df[where], x='country', y='violin', kind='violin')
```





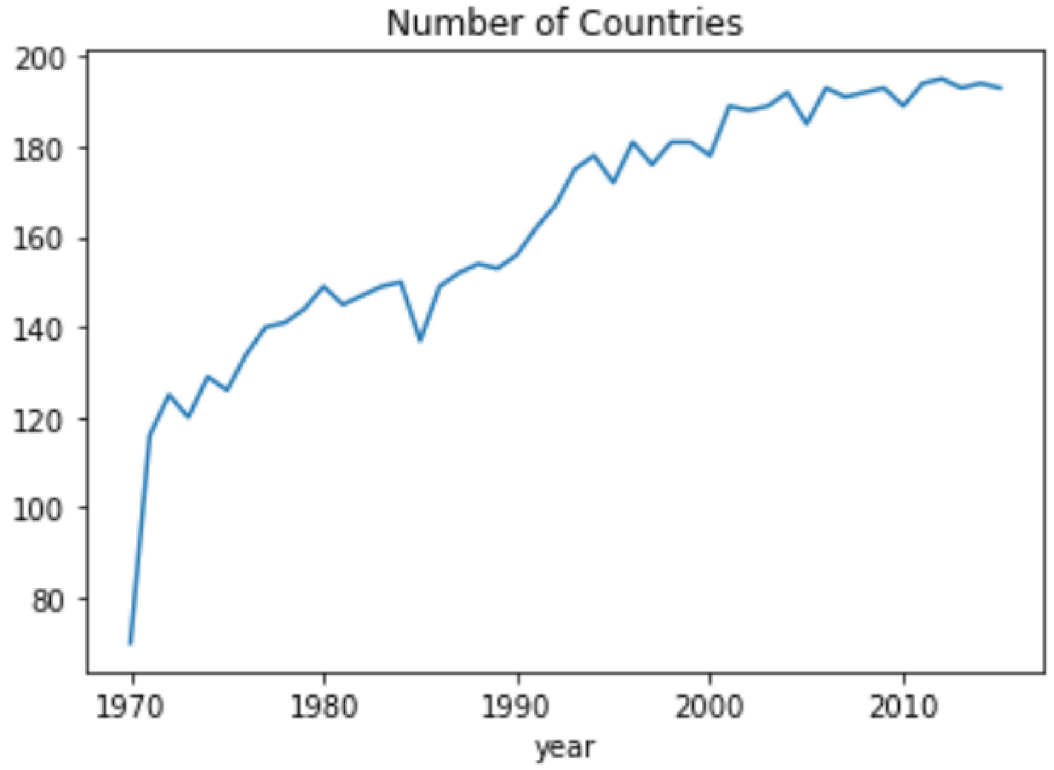
Visualizing developments over time

- If your data contains date or time attributes, it is always interesting to visualize developments over time
- Data may show trends, seasonality, cycle, etc.
- These time series elements are important to understand the changes in different aspects at different times

6. Development over time Number of countries

```
df.groupby('year').size().plot(title='Number of Countries')
```

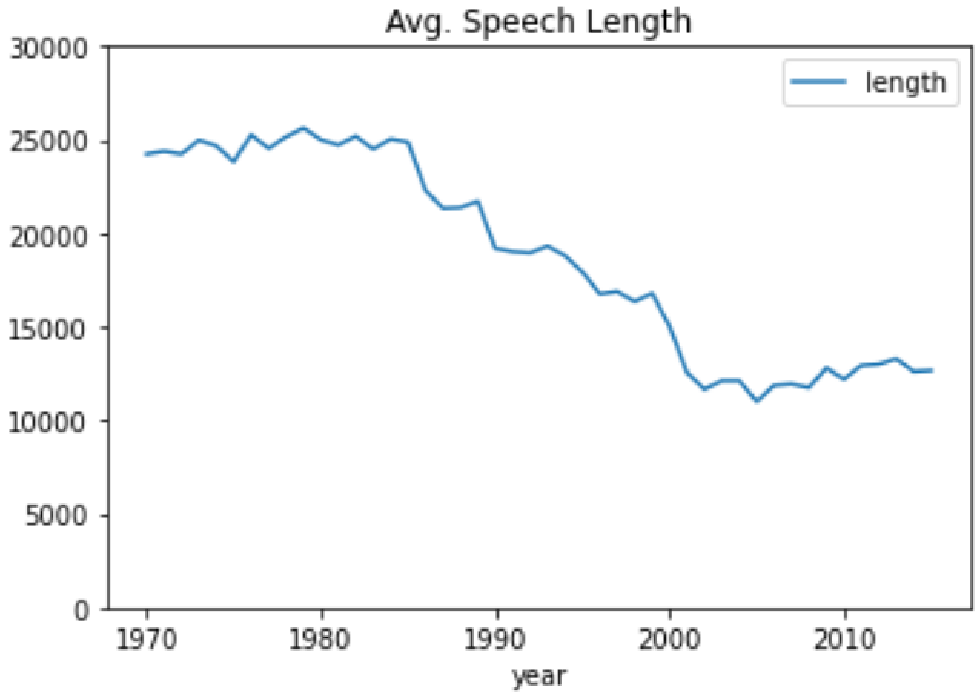
```
<AxesSubplot:title={'center':'Number of Countries'}, xlabel='year'>
```



6. Development over time
Speech length

```
df.groupby('year').agg({'length':'mean'})\
  .plot(title='Avg. Speech Length', ylim=(0,30000))
```

<AxesSubplot:title={'center':'Avg. Speech Length'}, xlabel='year'>



Building a simple text processing pipeline

- Analysis of metadata such as categories, time, authors and other attributes gives some first insights on the corpus
- Text content provide much more interesting insights
- We will develop a basic blueprint to prepare text for a quick analysis

Source Text → Case-Folding → Tokenization → Stop Word Removal → Prepared Tokens

7. Simple text processing Case-Folding

7.1 Case Folding

```
str.lower('Hello World!')
```

```
'hello world!'
```

7. Simple text processing Tokenization

7.2 Tokenization

```
import regex as re

def tokenize(text):
    return re.findall(r'[\w-]*\p{L}[\w-]*', text)
```

```
text = "Let's defeat SARS-Cov-2 together in 2021!"
tokens = tokenize(text)
print("|".join(tokens))
```

```
Let|s|defeat|SARS-Cov-2|together|in
```

7. Simple text processing Stopword Removal

7.3 Stopword Removal

```
import nltk

stopwords = set(nltk.corpus.stopwords.words('english'))
stopwords
```

```
{'a',
 'about',
 'above',
 'after',
 'again',
```

```
def remove_stop(tokens):
    return [t for t in tokens if t.lower() not in stopwords]
```

```
remove_stop(tokens)
```

```
['Let', 'defeat', 'SARS-Cov-2', 'together']
```

7. Simple text processing
Building processing pipeline

7.4 Processing a pipeline

```
pipeline = [str.lower, tokenize, remove_stop]

def prepare(text, pipeline):
    tokens = text
    for transform in pipeline:
        tokens = transform(tokens)
    return tokens
```

```
prepare(text, pipeline)
```

```
['let', 'defeat', 'sars-cov-2', 'together']
```

Pandas higher order functions

Series.map

  Works element by element on a Pandas Series

Series.apply

  Same as map but allows additional parameters

DataFrame.applymap

  Element by element on a Pandas DataFrame

DataFrame.apply

  Words on rows or columns of a DataFrame and supports aggregation

7. Simple text processing Applying pipeline

7.5 Applying pipeline

```
df['tokens'] = df['text'].apply(prepare, pipeline=pipeline)
df.sample(3)
```

	session	year	country	country_name	speaker	position	text	length	tokens
6262	64	2009	MAR	Morocco	Taib Fassi Fihri	Minister for Foreign Affairs	On behalf of the \nKingdom of Morocco, I shoul...	14465	[behalf, kingdom, morocco, like, congratulate,...
4094	52	1997	ZAF	South Africa	Alfred Nzo	Minister for Foreign Affairs	South Africa welcomes your\nelection, Sir, as...	18539	[south, africa, welcomes, election, sir, presi...
3581	50	1995	BEN	Benin	Mr. Monnou	Minister for Foreign Affairs	You have the difficult and noble task, Sir, of...	16198	[difficult, noble, task, sir, guiding, work, g...

7. Simple text processing Number of tokens

BIG DATA EXPERIENCE

7.6 Counting number of tokens (words)

```
df['num_tokens'] = df['tokens'].map(len)
```

```
df.head()
```

	session	year	country	country_name	speaker	position	text	length	tokens	num_tokens
0	25	1970	ALB	Albania	Mr. NAS	NaN	33: May I first convey to our President the co...	51419	[may, first, convey, president, congratulation...	4092
1	25	1970	ARG	Argentina	Mr. DE PABLO PARDO	NaN	177.\t : It is a fortunate coincidence that pr...	29286	[fortunate, coincidence, precisely, time, unit...	2341
2	25	1970	AUS	Australia	Mr. McMAHON	NaN	100.\t It is a pleasure for me to extend to y...	31839	[pleasure, extend, mr, president, warmest, con...	2575
3	25	1970	AUT	Austria	Mr. KIRCHSCHLAEGER	NaN	155.\t May I begin by expressing to Ambassado...	26616	[may, begin, expressing, ambassador, hambro, b...	2166
4	25	1970	BEL	Belgium	Mr. HARMEL	NaN	176. No doubt each of us, before coming up to ...	25911	[doubt, us, coming, rostrum, wonders, usefulne...	2064

KMUTT g-able

Word frequency analysis

- Frequency used words and phrases can give us some basic understanding of topics
- However, word frequency analysis ignores the order and the context
- Bag of words
- Not work well with complex task, but work well for classification

Counting words with a Counter

```python
from collections import Counter

tokens = tokenize("She likes my cats and my cats like my sofa")

counter = Counter(tokens)
print(counter)
```

```
Counter({'my': 3, 'cats': 2, 'She': 1, 'likes': 1, 'and': 1, 'like': 1, 'sofa': 1})
```

Adding more tokens to count

```python
more_tokens = tokenize("She likes dogs and cats")
counter.update(more_tokens)
print(counter)
```

```
Counter({'my': 3, 'cats': 3, 'She': 2, 'likes': 2, 'and': 2, 'like': 1, 'sofa': 1, 'dogs': 1})
```

Parallel count

```
%%time
import numpy as np
tokens = df['tokens'].explode().values
counter = Counter(tokens)
print(counter.most_common(5))
```

```
[('nations', 124508), ('united', 120763), ('international', 117223), ('world', 89421), ('countries', 85734)]
Wall time: 1.8 s
```

```
%%time
counter = Counter()

df['tokens'].map(counter.update)

print(counter.most_common(5))
```

```
[('nations', 124508), ('united', 120763), ('international', 117223), ('world', 89421), ('countries', 85734)]
Wall time: 961 ms
```

Word Counting, DataFrame version

```
def count_words(df, column='tokens', preprocess=None, min_freq = 2):

    # process tokens and update counter
    def update(doc):
        tokens = doc if preprocess is None else preprocess(doc)
        counter.update(tokens)

    # create counter and run through all data
    counter = Counter()
    df[column].map(update)

    # transform counter into a DataFrame
    freq_df = pd.DataFrame.from_dict(counter, orient='index', columns=['freq'])
    freq_df = freq_df.query('freq > @min_freq')
    freq_df.index.name = 'token'

    return freq_df.sort_values('freq', ascending=False)
```

BIG DATA EXPERIENCE

KMUTT g-able

count_words function

```
freq_df = count_words(df)
freq_df.head(6)
```

	freq
token	
nations	124508
united	120763
international	117223
world	89421
countries	85734
peace	72625

Counting words with preprocessing

```
# count words with 10 or more characters
count_words(df, column='text',
            preprocess=lambda text: re.findall(r"\w{10,}", text))
```

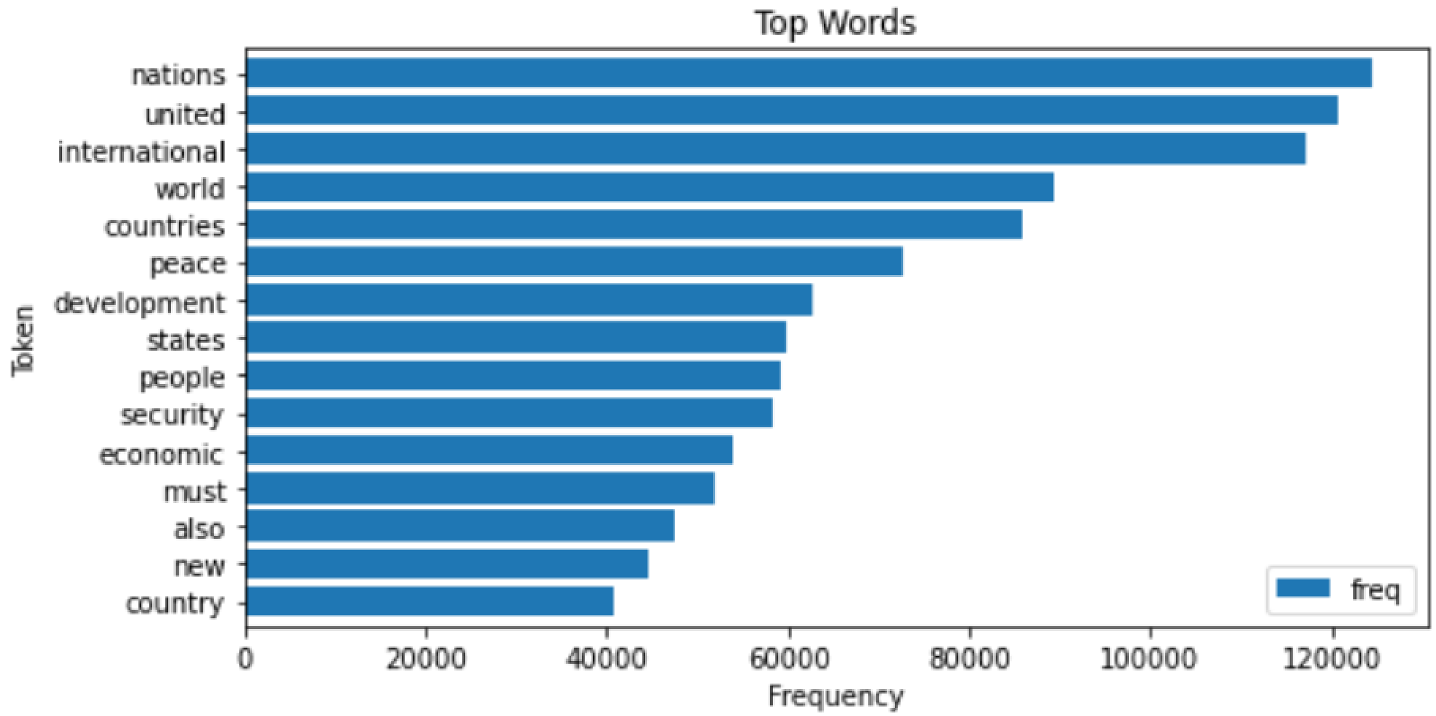
token	freq
international	106974
development	51334
Government	35528
Organization	33763
developing	25177
...	...
preexisting	3

BIG DATA EXPERIENCE
KMUTT g-able

10. Frequency plot

```
ax = freq_df.head(15).plot(kind='barh', width=0.8, figsize=(8,4))
ax.invert_yaxis()
ax.set(xlabel='Frequency', ylabel='Token', title='Top Words')
```

```
[Text(0.5, 0, 'Frequency'), Text(0, 0.5, 'Token'), Text(0.5, 1.0, 'Top Words')]
```



Word Clouds

- Plot of frequency distributions give detailed information about the token frequencies
- But it is difficult to compare across time periods, categories, and others
- Word clouds, in contrast, visualize the frequencies by different font sizes
- Easier to comprehend and to compare
- Not suitable for long words or word with capital letters as they get unproportionally high attraction

Installing wordcloud

```
!pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.1-cp38-cp38-win_amd64.whl (155 kB)
Requirement already satisfied: numpy>=1.6.1 in c:\users\santi\appdata\roaming\python\python38\site-packages (from wordcloud)
(1.19.5)
Requirement already satisfied: matplotlib in c:\users\santi\anaconda3\lib\site-packages (from wordcloud) (3.3.2)
Requirement already satisfied: pillow in c:\users\santi\anaconda3\lib\site-packages (from wordcloud) (8.0.1)
Requirement already satisfied: cycler>=0.10 in c:\users\santi\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\santi\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\santi\anaconda3\lib\site-packages (from mat
plotlib->wordcloud) (2.4.7)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\santi\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(2020.6.20)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\santi\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.
3.0)
Requirement already satisfied: six in c:\users\santi\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->wordcloud) (1.
15.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
```

Getting the text

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = df.query("year==2015 and country=='USA'")['text'].values[0]
text
```

```
'Seventy years after the founding of the United Nations it is worth reflecting on wh
ther, have helped to achieve. Out of the ashes of the Second World War, having witne
age, the United States has worked with many nations in the Assembly to prevent a thi
old adversaries; by supporting the steady emergence of strong democracies accountabl
Power; and by building an international system that imposes a cost on those who choo
at recognizes the dignity and equal worth of all people.\nThat has been the work of
s body has, at its best, pursued. Of course, there have been too many times when, co
ideals. Over the seven decades, terrible conflicts have claimed untold victims. But
y, to make a system of international rules and norms that are better and stronger an
nal order that has underwritten unparalleled advances in human liberty and prosperit
```

BIG DATA EXPERIENCE

Building word clouds

```python
wc = WordCloud(max_words=100, stopwords=stopwords)
wc.generate(text)
plt.figure(dpi=150)
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

Keyword-in-context (KWIC)

- KWIC produces a list of text fragments of equal length showing the left and right context of a keyword
- KWIC analysis is implemented in NLTK and textacy

```
!pip install textacy
```

```
Collecting textacy
  Downloading textacy-0.11.0-py3-none-any.whl (200 kB)
Requirement already satisfied: tqdm>=4.19.6 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (4.50.2)
Requirement already satisfied: joblib>=0.13.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (0.17.0)
Requirement already satisfied: numpy>=1.17.0 in c:\users\santi\appdata\roaming\python\python38\site-packages (from textacy) (1.
19.5)
Requirement already satisfied: spacy>=3.0.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (3.1.1)
Requirement already satisfied: cachetools>=4.0.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (4.2.1)
Requirement already satisfied: scikit-learn>=0.19.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (0.23.2)
Collecting jellyfish>=0.8.0
  Downloading jellyfish-0.8.4-cp38-cp38-win_amd64.whl (27 kB)
Requirement already satisfied: requests>=2.10.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (2.24.0)
Requirement already satisfied: cytoolz>=0.10.1 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (0.11.0)
Requirement already satisfied: networkx>=2.0 in c:\users\santi\anaconda3\lib\site-packages (from textacy) (2.5)
```

KWIC display function

```
from textacy.extract.kwic import keyword_in_context
import random

def kwic(doc_series, keyword, window=35, print_samples=5):

    def add_kwic(text):
        kwic_list.extend(keyword_in_context(text, keyword, ignore_case=True, window_width=window))

    kwic_list = []
    doc_series.map(add_kwic)

    if print_samples is None or print_samples==0:
        return kwic_list
    else:
        k = min(print_samples, len(kwic_list))
        print(f"{k} random samples out of {len(kwic_list)} " + \
              f"contexts for '{keyword}':")
        for sample in random.sample(list(kwic_list), k):
            print(re.sub(r'[\n\t]', ' ', sample[0]) + '  ' + \
                  sample[1]+'  ' +\
                  re.sub(r'[\n\t]',' ', sample[2]))
```

KWIC

```
kwic(df[df['year']==2015]['text'], 'sdgs', print_samples=5)
```

```
5 random samples out of 73 contexts for 'sdgs':
the Sustainable Development Goals (   SDGs   ) will be an effective tool in glob
ovatively to achieve the 17 global    SDGs    and the 169 targets. It has to be
the Sustainable Development Goals (   SDGs   ) contained therein, along with the
 population in accordance with the    SDGs   . The leaders of the Pacific small
eholder for and beneficiary of the    SDGs   . The development of human capital
```

BIG DATA EXPERIENCE

Word combination

- Knowing word frequency is not enough
- For examples, climate changes or political climate are combinations of word
- We are looking for 2 types of word sequences: componds and collocations
  - Compound is a combination of two or more words with a specific meaning
    - Close form: earthquake, hyphenated: self-confidence, open form: climate change
  - Collocation are words that are frequently used together, such as red carpet, united nation, etc.

N-gram

- N-gram of size 1 are single words, also called unigram
- N-gram of size 2 are combinations of 2 words, also called bigram
- Stick with N ≤ 3, because the number of combinations increases exponentially

Bigram of [She ,like, my, cat, and, my, cat, like, my, sofa]

She like | like my | my cat | cat and | and my | my cat | cat like | like my | my sofa

N-gram

13. N-gram

```
def ngrams(tokens, n=2, sep=' '):
    return [sep.join(ngram) for ngram in zip(*[tokens[i:] for i in range(n)])]
```

```
text = "the visible manifestation of the global climate change"
tokens = tokenize(text)
print("|".join(ngrams(tokens,2)))
```

```
the visible|visible manifestation|manifestation of|of the|the global|global climate|climate change
```

BIG DATA EXPERIENCE

N-gram with stopword removal

14. N-gram with stopword

```
def ngrams(tokens, n=2, sep=' ', stopwords=set()):
    return [sep.join(ngram) for ngram in zip(*[tokens[i:] for i in range(n)])
            if len([t for t in ngram if t in stopwords])==0]
```

```
text = "the visible manifestation of the global climate change"
tokens = tokenize(text)
print("|".join(ngrams(tokens,2, stopwords=stopwords)))
```

```
visible manifestation|global climate|climate change
```

Comparing frequencies across time intervals and categories

- Like Google trends
- This kind of trend analysis computes frequencies by da and visualizes them with a line chart
- We want to track the development of certain keywords over the course of the years in our UN Debates dataset

Creating frequency timelines

```
def count_keywords(tokens, keywords):
    tokens = [t for t in tokens if t in keywords]
    counter = Counter(tokens)
    return [counter.get(k, 0) for k in keywords]
```

```
keywords = ['nuclear','terrorism','climate','freedom']
tokens = ['nuclear','climate','climate','freedom','climate','freedom']

print(count_keywords(tokens, keywords))
```

```
[1, 0, 3, 2]
```

Creating frequency timelines

```
def count_keywords_by(df, by, keywords, column='tokens'):
    freq_matrix = df[column].apply(count_keywords, keywords=keywords)
    freq_df = pd.DataFrame.from_records(freq_matrix, columns=keywords)
    freq_df[by] = df[by]

    return freq_df.groupby(by).sum().sort_values(by)
```

```
freq_df = count_keywords_by(df, by='year', keywords=keywords)
```

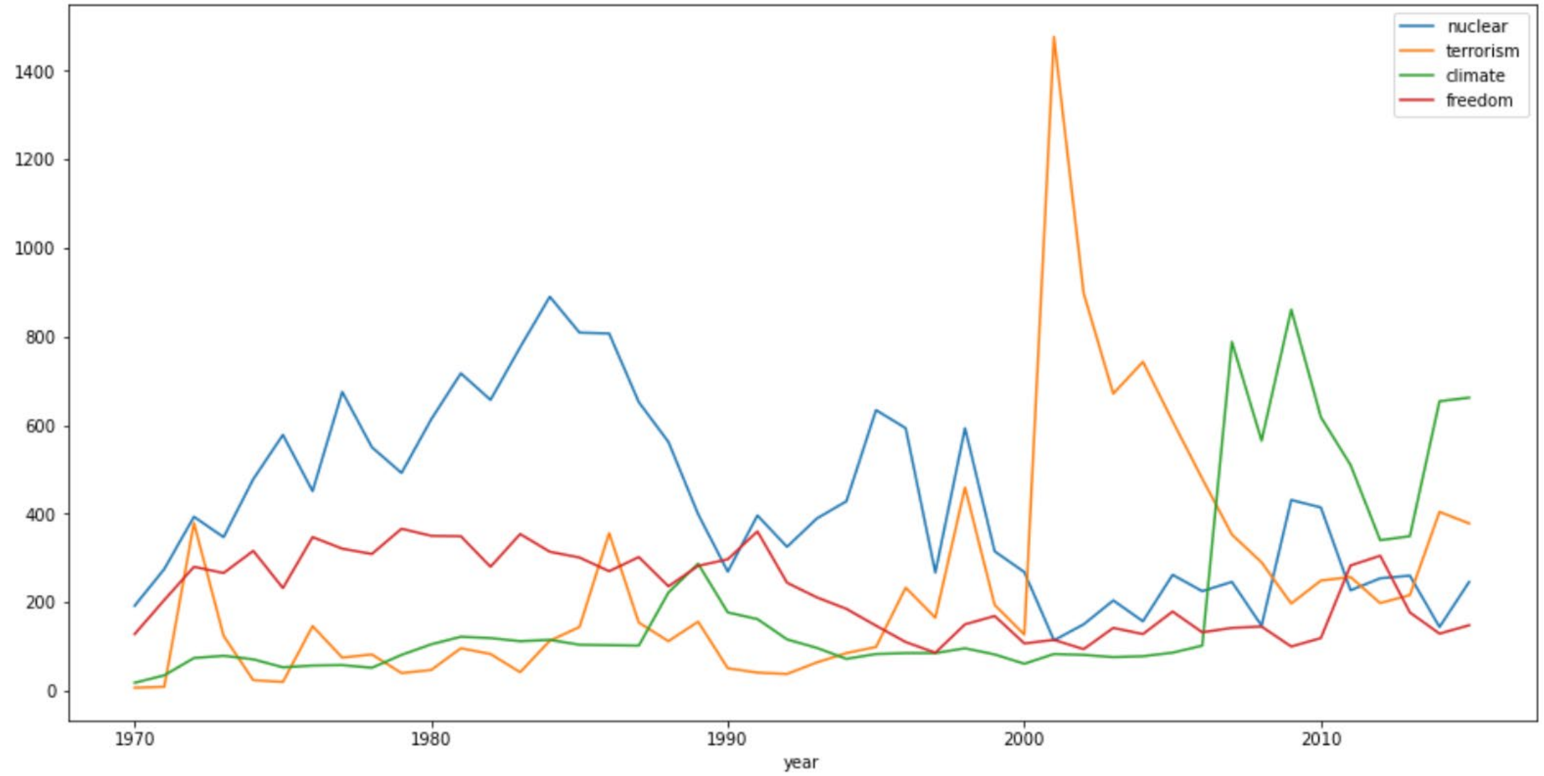
```
freq_df
```

year	nuclear	terrorism	climate	freedom
1970	192	7	18	128
1971	275	9	35	205
1972	393	379	74	280
1973	347	124	79	266
1974	478	24	71	316

Plot the trend graph

```
freq_df.plot(kind='line', figsize=(16,8))
```

<AxesSubplot:xlabel='year'>



Lab

1. Find the top 10 word bigram from UN General Debates of years 1970 – 1990 and compare with those of years 1990 – the latest (remove stopwords first)
2. Create a bigram word cloud of the UN General Debates dataset of years 1970 – 1990 and 1990 to the latest (remove stopwords first)
3. Create a trend graph showing the bigram and word trend of “climate change”, “global warming”, “wars” and 3 others of your choices
4. Submit your work to LEB2 before the next class

End of Lecture 2

Question?