State of the Union (1947-2022): Sentiment Analysis and Topic Modeling

By: Shirsho Dasgupta (2022)

General Notes:

Some visualizations may not be compatible with viewing directly on Github. View it by copying the URL here or run the code after downloading this repo.

Notes on Sentiment Analysis:

The code reads the text of every State of the Union speech (delivered in-person on the Hill) from 1947 to 2022 and performs sentiment analysis on them using the NRC Word-Emotion Association Lexicon (EmoLex).

The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing.

The code matches the words in the speeches to that of the dictionary then adds up the factor of the emotion.

Since EmoLex is, at the end of the day, crowdsourced and finite, there might be some words which are outside its scope or some nuances which are not accounted for (the code performs word-to-word comparison). The analysis is always at best an approximation.

Notes on Topic Modeling:

The code deploys machine-learning modules on the text of every State of the Union speech (delivered in-person on the Hill) from 1947 to 2022. The algorithm calculates the frequency of each word (controlling for how common they are etc.) and generates topics that were touched on.

The code uses the Scikitlearn module to compute Non-Negative Matrix Factorization (NMF)/Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) methods. It also deploys Gensim to compute the same.

Sources:

State of the Union Archived Speeches at University of California, Santa Barbara.

NRC Emotion Lexicon

Importing libraries

In [1]:
```
import numpy as np
import re
import pandas as pd
%matplotlib inline
```

Creating dataframe of speeches

Importing SOTU speeches

In [2]:
```
import glob

filenames = glob.glob("sotu/*")
```

In [3]:
```
speeches = [open(filename).read() for filename in filenames]
len(speeches)
```

Out[3]: 75

Storing SOTU speeches in one dataframe

In [4]:
```
### stores entire text of speech under one column and the associated filename under another
speeches_df = pd.DataFrame({"text": speeches, "filename": filenames})
speeches_df.head(3)
```

Out[4]:

	text	filename
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt

In [5]:
```
### creates two new columns to store year and name of president
speeches_df["year"] = " "
speeches_df["pres"] = " "
speeches_df["id"] = " "

### loop runs through dataframe
for i in range(0, len(speeches_df)):

    ### extracts (from filename) and stores year of speech
    speeches_df["year"][i] = int(speeches_df["filename"][i][-8:-4])
    x = speeches_df["year"][i]

    ### condition checks what year the speech was delivered and stores the name of the president accordingly
    if (x >= 1947) & (x <= 1952):
        speeches_df["pres"][i] = "truman"
    elif (x >= 1953) & (x <= 1960):
        speeches_df["pres"][i] = "eisenhower"
    elif (x >= 1961) & (x <= 1963):
        speeches_df["pres"][i] = "kennedy"
    elif (x >= 1964) & (x <= 1969):
        speeches_df["pres"][i] = "johnson"
    elif (x >= 1970) & (x <= 1974):
        speeches_df["pres"][i] = "nixon"
    elif (x >= 1975) & (x <= 1977):
        speeches_df["pres"][i] = "ford"
    elif (x >= 1978) & (x <= 1980):
        speeches_df["pres"][i] = "carter"
    elif (x >= 1981) & (x <= 1988):
        speeches_df["pres"][i] = "reagan"
    elif (x >= 1989) & (x <= 1992):
        speeches_df["pres"][i] = "bush sr."
    elif (x >= 1993) & (x <= 2000):
        speeches_df["pres"][i] = "clinton"
```

```
    elif (x >= 2001) & (x <= 2008):
        speeches_df["pres"][i] = "bush jr."
    elif (x >= 2009) & (x <= 2016):
        speeches_df["pres"][i] = "obama"
    elif (x >= 2017) & (x <= 2020):
        speeches_df["pres"][i] = "trump"
    elif (x > 2020):
        speeches_df["pres"][i] = "biden"

    speeches_df["id"][i] = speeches_df["pres"][i] + "-" + str(speeches_df["year"][i])
```

In [6]:
```
### displays final master dataframe
speeches_df
```

Out[6]:

	text	filename	year	pres	id
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021
...	...	...	...	...	...
70	Mr. Speaker, Mr. Vice President, Members of Co...	sotu/01-30-2018.txt	2018	trump	trump-2018
71	Mr. Speaker, Vice President Cheney, Members of...	sotu/01-28-2003.txt	2003	bush jr.	bush jr.-2003
72	Thank you very much. Mr. Speaker, Vice Preside...	sotu/01-29-2002.txt	2002	bush jr.	bush jr.-2002
73	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-08-1951.txt	1951	truman	truman-1951
74	Mr. Vice President, my old colleague from Mass...	sotu/01-11-1962.txt	1962	kennedy	kennedy-1962

75 rows × 5 columns

Creating separate dataframes for each president

In [7]:
```
### filters by name of president and stores in separate dataframes
speeches_df_obama = speeches_df[speeches_df["pres"] == "obama"]
speeches_df_trump = speeches_df[speeches_df["pres"] == "trump"]
speeches_df_biden = speeches_df[speeches_df["pres"] == "biden"]
```

Sentiment analysis

Importing NRC Emotion Lexicon

Download compressed file from http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

Move the relevant text-file to the directory

In [8]:
```
filepath = "NRC-Emotion-Lexicon-Wordlevel-v0.92.txt"
emolex_df = pd.read_csv(filepath,  names = ["word", "emotion", "association"], sep = "\t", keep_default_na = False)
emolex_df = emolex_df.pivot(index = "word", columns = "emotion", values = "association").reset_index()
emolex_df.head()
```

Out[8]:

emotion	word	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust
0	aback	0	0	0	0	0	0	0	0	0	0
1	abacus	0	0	0	0	0	0	0	0	0	1
2	abandon	0	0	0	1	0	1	0	1	0	0
3	abandoned	1	0	0	1	0	1	0	1	0	0
4	abandonment	1	0	0	1	0	1	0	1	1	0

Calculating share of words using TfidfVectorizer

In [9]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer(vocabulary = emolex_df.word, use_idf = False, norm = "l1")
matrix = vec.fit_transform(speeches_df.text)
vocab = vec.get_feature_names()
wordcount_df = pd.DataFrame(matrix.toarray(), columns = vocab)
wordcount_df.head()
```

```
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is dep
recated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[9]:

	aback	abacus	abandon	abandoned	abandonment	abate	abatement	abba	abbot	abbreviate	...	zephyr	zeppelin	zest	zip	zodiac	zone	zoo	zoological	zoology	zoom
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00105	0.0	0.0	0.0	0.0

5 rows × 14154 columns

Applying EmoLex

The words which appear per category in the NRC Emotion Lexicon dictionary are identified and their shares are added up. The sum gives the total share of the document that relates to a particular emotion or sentiment. For instance say a document has two "angry" words each making up 0.01 of the text, then the anger-factor as it were of the text is 0.02 (= 0.01 + 0.01).

In [10]:
```
### identifies all negative words
neg_words = emolex_df[emolex_df.negative == 1]["word"]
### adds up shares of negative words
speeches_df["negative"] = wordcount_df[neg_words].sum(axis = 1)

### the above process is repeated per sentiment

pos_words = emolex_df[emolex_df.positive == 1]["word"]
speeches_df["positive"] = wordcount_df[pos_words].sum(axis = 1)

angry_words = emolex_df[emolex_df.anger == 1]["word"]
speeches_df["anger"] = wordcount_df[angry_words].sum(axis = 1)

anticip_words = emolex_df[emolex_df.anticipation == 1]["word"]
speeches_df["anticipation"] = wordcount_df[anticip_words].sum(axis = 1)
```

```
disgust_words = emolex_df[emolex_df.disgust == 1]["word"]
speeches_df["disgust"] = wordcount_df[disgust_words].sum(axis = 1)

fear_words = emolex_df[emolex_df.fear == 1]["word"]
speeches_df["fear"] = wordcount_df[fear_words].sum(axis = 1)

joy_words = emolex_df[emolex_df.joy == 1]["word"]
speeches_df["joy"] = wordcount_df[joy_words].sum(axis = 1)

sad_words = emolex_df[emolex_df.sadness == 1]["word"]
speeches_df["sadness"] = wordcount_df[sad_words].sum(axis = 1)

surprise_words = emolex_df[emolex_df.surprise == 1]["word"]
speeches_df["surprise"] = wordcount_df[surprise_words].sum(axis = 1)

trust_words = emolex_df[emolex_df.trust == 1]["word"]
speeches_df["trust"] = wordcount_df[trust_words].sum(axis = 1)
```

In [11]:
```
### displays final dataframe
speeches_df.head()
```

Out[11]:

	text	filename	year	pres	id	negative	positive	anger	anticipation	disgust	fear	joy	sadness	surprise	trust
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955	0.109145	0.328276	0.045512	0.128108	0.028234	0.093974	0.096924	0.040455	0.030763	0.226717
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954	0.114540	0.304916	0.044979	0.116632	0.022490	0.091004	0.080544	0.041318	0.031381	0.211297
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972	0.112086	0.292398	0.049708	0.115010	0.039961	0.082846	0.092593	0.036062	0.034113	0.195906
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993	0.126192	0.243971	0.042625	0.123948	0.021873	0.058329	0.076276	0.053842	0.035895	0.162086
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021	0.132283	0.266142	0.058268	0.114961	0.034646	0.087664	0.080315	0.061942	0.039370	0.167454

Exporting the dataset

In [12]:
```
speeches_df.to_csv("sotu_analysis.csv", index = False)
```

Re-importing created dataset

In [13]:
```
speeches = pd.read_csv("sotu_analysis.csv")
```

In [14]:
```
speeches.shape
```

Out[14]:
```
(75, 15)
```

Plotting sentiments

Importing Altair libraries

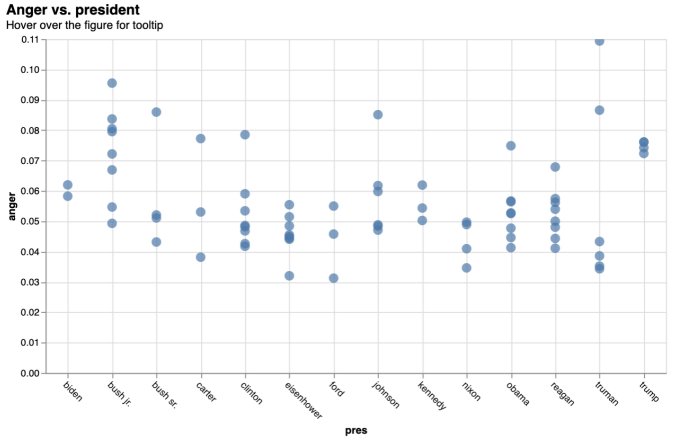
In [15]:
```
import altair as alt

alt.renderers.enable("default")
```

Out[15]:
```
RendererRegistry.enable('default')
```

Plotting anger-factor by president

In [16]:
```
alt.Chart(speeches).mark_circle(size = 100).encode(
    x = "pres",
    y = "anger",
    tooltip = ["pres", "year", "negative"]
).configure_axis(
    grid = True
).properties(
    width = 650,
    height = 350,title = {
      "text": "Anger vs. president",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).configure_axisX(
    labelAngle = 45
).interactive()
```

Out[16]:

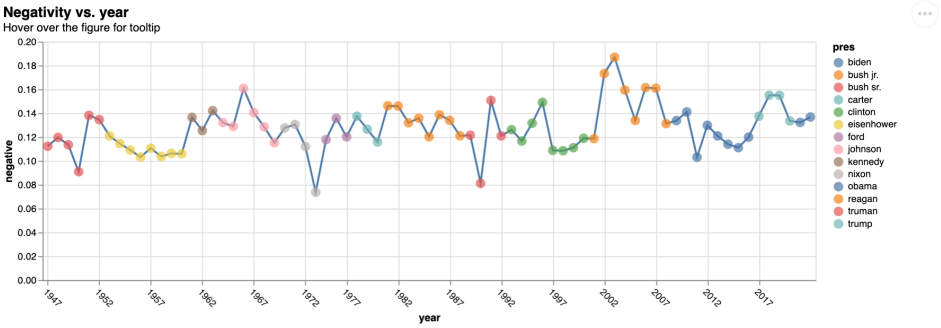


Plotting negativity-factor by year

In [17]:
```
### creates chart with point-markings
points = alt.Chart(speeches).mark_circle(size = 100).encode(
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    alt.Y("negative"),
    alt.Color("pres"),
    tooltip = ["pres", "year", "negative"] ### adds tooltip on mouse-hover
)
```

```
### creates line chart
line = alt.Chart(speeches).mark_line().encode(
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    alt.Y("negative"),
)

### layers both charts and adds some other properties
(line + points).configure_axis(
    grid = True
).configure_axisX(
    labelAngle = 45
).properties(
    width = 810,
    height = 250,
    title = {
      "text": "Negativity vs. year",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).interactive()
```

Out[17]:



Topic modeling

Setting dataframe

In [18]:

```
sotu = speeches[["text", "filename", "year", "pres", "id"]].copy()
sotu.head()
```

Out[18]:

	text	filename	year	pres	id
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021

Non-Negative Matrix Factorization (NMF)/Latent Semantic Indexing (LSI) topic modeling

Vectorizing all words

In [19]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english")

class StemmedTfidVectorizer(TfidfVectorizer):
    def build_analyzer(self):
        analyzer = super(StemmedTfidVectorizer, self).build_analyzer()
        return lambda doc:(stemmer.stem(word) for word in analyzer(doc))

tfidf_vectorizer = StemmedTfidVectorizer(stop_words = "english", min_df = 5, max_df = 0.40)
x = tfidf_vectorizer.fit_transform(sotu.text.astype(str))
```

Deploying NMF module

In [20]:

```
%%time

### imports modules
from sklearn.decomposition import NMF

### selects number of topics
model = NMF(n_components = 10)
model.fit(x)

### selects number of words per model
n_words = 7
### stores each word
feature_names = tfidf_vectorizer.get_feature_names()

### creates new arrays to store list of topics and topic numbers
topic_list_nmf = []
topic_list_index_nmf = []

### applies NMF modeling
for topic_idx, topic in enumerate(model.components_):
    top_n = [feature_names[i]
             for i in topic.argsort()
             [-n_words:]][::-1]
    top_features = ' '.join(top_n)
    topic_list_nmf.append(f"{'_'.join(top_n[:3])}")
    topic_list_index_nmf.append("Topic " + str(topic_idx + 1))

    ### displays topics
    print("Topic " + str(topic_idx + 1) + ": " + top_features)
```

```
Topic 1: kid class manufactur oil doesn innov gas
Topic 2: agricultur expenditur adequ atom adjust 1946 communist
Topic 3: iraq iraqi terrorist al qaida saddam regim
Topic 4: ryan isi pillar gang hero incred veteran
```

```
Topic 5: nicaragua revolut excel veto gramm rudman governor
Topic 6: oil 1974 1980 barrel 1976 capabl branch
Topic 7: vietnam south expenditur communist abund desir wish
Topic 8: 21st ought gun got read class grade
Topic 9: vaccin pandem gun folk 19 childcar cancer
Topic 10: communist session atlant disarma nam viet lack
CPU times: user 480 ms, sys: 20.8 ms, total: 500 ms
Wall time: 147 ms
```

```
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/sklearn/decomposition/_nmf.py:289: FutureWarning: The 'init' value, when 'init=None' and n_components is less than n_samples and n_features, will be changed from 'nndsvd' to 'nndsvda' in 1.1 (renaming of 0.26).
  warnings.warn(
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Converting computed factors into dataset

In [21]:
```
### converts counts into numbers
amounts = model.transform(x) * 100

### sets up dataframe with corresponding topic-numbers and amount numbers
topics_nmf = pd.DataFrame(amounts, columns = topic_list_index_nmf)

### sets up dataframe with corresponding topics and subjects
topics_list_nmf = pd.DataFrame(amounts, columns = topic_list_nmf)

topics_nmf
```

Out[21]:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	0.613844	54.268987	0.000000	0.000000	0.200546	0.000000	1.904521	0.173698	0.000000	1.615704
1	0.330403	51.035306	0.000000	0.000000	0.000000	3.638860	2.363694	0.000000	0.000000	0.189744
2	0.000000	0.000000	0.000000	1.137020	6.012578	12.888952	7.798940	0.000000	0.000000	25.867264
3	8.488645	1.810723	0.000000	0.000000	0.000000	7.050203	2.624654	24.156173	0.000000	2.904309
4	0.000000	0.000000	0.000000	0.000000	0.000000	1.639435	0.063283	0.000000	90.373168	0.000000
...	...	...	...	...	...	...	...	...	...	...
70	0.000000	0.000000	0.000000	77.829350	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
71	0.000000	1.374149	52.437751	0.000000	0.000000	0.000000	0.000000	2.176445	0.000000	0.000000
72	0.000000	0.000000	41.596349	9.516587	1.327411	0.274926	0.000000	0.000000	6.703602	0.000000
73	0.000000	8.065589	0.000000	1.125444	0.000000	0.000000	0.000000	0.000000	5.305285	39.098590
74	0.000000	4.553163	0.000000	0.000000	1.240334	2.385833	0.000000	0.000000	2.058022	63.850685

75 rows × 10 columns

Storing topic list as a dataframe

In [22]:
```
topic_indices = list(topics_nmf)
topic_subjects = list(topics_list_nmf)
topic_desc = pd.DataFrame(topic_indices, topic_subjects).reset_index()
topic_desc_nmf = topic_desc.rename(columns = {"index": "topic_desc", 0: "topic"})
topic_desc_nmf
```

Out[22]:

	topic_desc	topic
0	kid_class_manufactur	Topic 1
1	agricultur_expenditur_adequ	Topic 2
2	iraq_iraqi_terrorist	Topic 3
3	ryan_isi_pillar	Topic 4
4	nicaragua_revolut_excel	Topic 5
5	oil_1974_1980	Topic 6
6	vietnam_south_expenditur	Topic 7
7	21st_ought_gun	Topic 8
8	vaccin_pandem_gun	Topic 9
9	communist_session_atlant	Topic 10

Merging speeches database with topics dataframe

In [23]:
```
nmf_merged = topics_nmf.join(sotu)
nmf_merged.head(3)
```

Out[23]:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	text	filename	year	pres	id
0	0.613844	54.268987	0.0	0.00000	0.200546	0.000000	1.904521	0.173698	0.0	1.615704	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	0.330403	51.035306	0.0	0.00000	0.000000	3.638860	2.363694	0.000000	0.0	0.189744	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	0.000000	0.000000	0.0	1.13702	6.012578	12.888952	7.798940	0.000000	0.0	25.867264	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972

Rearranging dataframe for plotting

In [24]:
```
nmf_1 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 1"]].copy()
nmf_1["topic"] = "Topic 1"
nmf_1 = nmf_1.rename(columns = {"Topic 1": "factor"})

nmf_2 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 2"]].copy()
nmf_2["topic"] = "Topic 2"
nmf_2 = nmf_2.rename(columns = {"Topic 2": "factor"})

nmf_3 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 3"]].copy()
nmf_3["topic"] = "Topic 3"
nmf_3 = nmf_3.rename(columns = {"Topic 3": "factor"})

nmf_4 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 4"]].copy()
nmf_4["topic"] = "Topic 4"
nmf_4 = nmf_4.rename(columns = {"Topic 4": "factor"})

nmf_5 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 5"]].copy()
nmf_5["topic"] = "Topic 5"
nmf_5 = nmf_5.rename(columns = {"Topic 5": "factor"})

nmf_6 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 6"]].copy()
nmf_6["topic"] = "Topic 6"
nmf_6 = nmf_6.rename(columns = {"Topic 6": "factor"})
```

```
nmf_7 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 7"]].copy()
nmf_7["topic"] = "Topic 7"
nmf_7 = nmf_7.rename(columns = {"Topic 7": "factor"})

nmf_8 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 8"]].copy()
nmf_8["topic"] = "Topic 8"
nmf_8 = nmf_8.rename(columns = {"Topic 8": "factor"})

nmf_9 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 9"]].copy()
nmf_9["topic"] = "Topic 9"
nmf_9 = nmf_9.rename(columns = {"Topic 9": "factor"})

nmf_10 = nmf_merged[["text", "filename", "year", "pres", "id", "Topic 10"]].copy()
nmf_10["topic"] = "Topic 10"
nmf_10 = nmf_10.rename(columns = {"Topic 10": "factor"})

nmf_master = pd.concat([nmf_1, nmf_2, nmf_3, nmf_4, nmf_5, nmf_6, nmf_7, nmf_8, nmf_9, nmf_10],ignore_index = True)
nmf_master = pd.merge(nmf_master, topic_desc_nmf, on = "topic", how = "left")

nmf_master
```

Out[24]:

	text	filename	year	pres	id	factor	topic	topic_desc
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955	0.613844	Topic 1	kid_class_manufactur
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954	0.330403	Topic 1	kid_class_manufactur
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972	0.000000	Topic 1	kid_class_manufactur
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993	8.488645	Topic 1	kid_class_manufactur
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021	0.000000	Topic 1	kid_class_manufactur
...	...	...	...	...	...	...	...	...
745	Mr. Speaker, Mr. Vice President, Members of Co...	sotu/01-30-2018.txt	2018	trump	trump-2018	0.000000	Topic 10	communist_session_atlant
746	Mr. Speaker, Vice President Cheney, Members of...	sotu/01-28-2003.txt	2003	bush jr.	bush jr.-2003	0.000000	Topic 10	communist_session_atlant
747	Thank you very much. Mr. Speaker, Vice Preside...	sotu/01-29-2002.txt	2002	bush jr.	bush jr.-2002	0.000000	Topic 10	communist_session_atlant
748	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-08-1951.txt	1951	truman	truman-1951	39.098590	Topic 10	communist_session_atlant
749	Mr. Vice President, my old colleague from Mass...	sotu/01-11-1962.txt	1962	kennedy	kennedy-1962	63.850685	Topic 10	communist_session_atlant

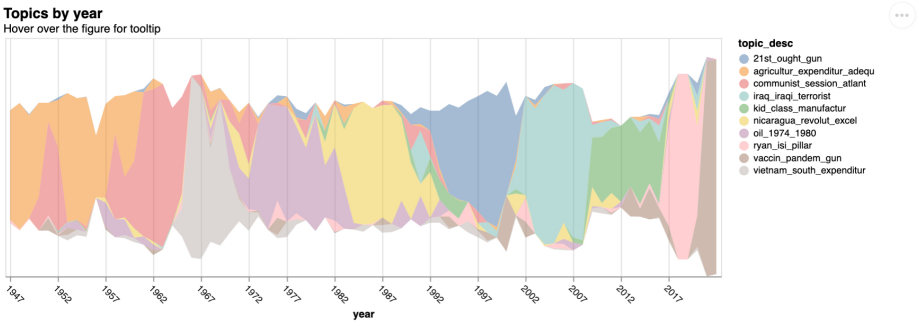
750 rows × 8 columns

Plotting steamgraph

In [25]:

```
alt.Chart(nmf_master).mark_area(opacity = 0.5).encode(
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    alt.Y("sum(factor):Q", stack = "center", axis = None),
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    tooltip = ["pres", "year", "topic_desc"]
).configure_axis(
    grid = True
).configure_axisX(
    labelAngle = 45
).properties(
    width = 750,
    height = 250,
    title = {
      "text": "Topics by year",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).interactive()
```

Out[25]:



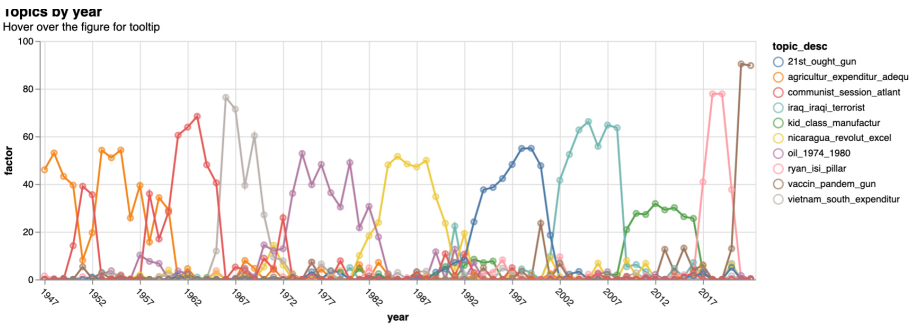
Plotting line chart

In [26]:

```
lines = alt.Chart(nmf_master).mark_line().encode(
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    y = "factor",
)

points = alt.Chart(nmf_master).mark_point().encode(
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    y = "factor",
    tooltip = ["pres", "year", "topic_desc"]
)

chart = lines + points

chart.configure_axis(
    grid = True
).configure_axisX(
    labelAngle = 45
).properties(
    width = 750,
    height = 250,
    title = {
      "text": "Topics by year",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).interactive()
```

Out[26]:

Out[26]:



Note: This same figure can also be drawn using selected topics.

Latent Dirichlet Allocation (LDA) topic modeling

Displaying dataframe

In [27]:
```
sotu
```

Out[27]:

	text	filename	year	pres	id
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021
...	...	...	...	...	...
70	Mr. Speaker, Mr. Vice President, Members of Co...	sotu/01-30-2018.txt	2018	trump	trump-2018
71	Mr. Speaker, Vice President Cheney, Members of...	sotu/01-28-2003.txt	2003	bush jr.	bush jr.-2003
72	Thank you very much. Mr. Speaker, Vice Preside...	sotu/01-29-2002.txt	2002	bush jr.	bush jr.-2002
73	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-08-1951.txt	1951	truman	truman-1951
74	Mr. Vice President, my old colleague from Mass...	sotu/01-11-1962.txt	1962	kennedy	kennedy-1962

75 rows × 5 columns

Vectorizing all words

In [28]:
```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english")

analyzer = CountVectorizer().build_analyzer()

class StemmedCountVectorizer(CountVectorizer):
    def build_analyzer(self):
        analyzer = super(CountVectorizer, self).build_analyzer()
        return lambda doc: (stemmer.stem(word) for word in analyzer(doc))

vectorizer = StemmedCountVectorizer(stop_words = "english", min_df = 5, max_df = 0.4)
x = vectorizer.fit_transform(sotu.text)
```

Note: LDA modeling has in-built TF-IDF unlike NMF, so only a CountVectorizer is required.

Deploying LDA module

In [29]:
```
%%time

### imports modules
from sklearn.decomposition import LatentDirichletAllocation

### assigns number of topics
n_topics = 10
model = LatentDirichletAllocation(n_components = n_topics)
model.fit(x)

### assigns number of words per topic
n_words = 7
feature_names = vectorizer.get_feature_names()

### applies LDA modeling
topic_list_lda = []
topic_list_index_lda = []
for topic_idx, topic in enumerate(model.components_):
    top_n = [feature_names[i]
             for i in topic.argsort()
             [-n_words:]][::-1]
    top_features = " ".join(top_n)
    topic_list_lda.append(f"{'_'.join(top_n[:3])}")
    topic_list_index_lda.append("Topic " + str(topic_idx + 1))

    print(f"Topic {topic_idx + 1}: {top_features}")
```

```
Topic 1: ought class kid governor revolut got tough
Topic 2: terrorist iraq iraqi afghanistan al qaida regim
Topic 3: vietnam oil communist latin expenditur south needi
Topic 4: hussein saddam inspector intellig chemic biolog agent
Topic 5: vietnam south voic door knowledg desir planet
Topic 6: session surplus read invit john senior mayor
Topic 7: communist agricultur expenditur adequ republ branch affair
Topic 8: veteran disput bargain 1946 collect atom jurisdict
Topic 9: vietnam januari abund battl travel 1969 seventi
Topic 10: 21st veteran class terrorist manufactur gun kid
CPU times: user 3.96 s, sys: 45.8 ms, total: 4.01 s
Wall time: 1.01 s
```

```
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is dep
recated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Converting computed factors into dataset

In [30]:
```
### converts counts into numbers
amounts = model.transform(x) * 100

### sets up dataframe with corresponding topic-numbers and amount numbers
topics_lda = pd.DataFrame(amounts, columns = topic_list_index_lda)

### sets up dataframe with corresponding topics and subjects
topics_list_lda = pd.DataFrame(amounts, columns = topic_list_lda)

topics_lda
```

Out[30]:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	0.012183	0.012182	0.012183	0.012182	0.012182	0.012182	99.890359	0.012183	0.012183	0.012183
1	0.014840	0.014839	0.014840	0.014839	0.014839	0.014839	99.866447	0.014839	0.014839	0.014839
2	23.976947	0.030310	14.822932	0.030308	0.030310	0.030309	49.192545	0.030308	11.825722	0.030311
3	63.740839	0.019497	19.967020	0.019496	0.019498	0.019497	9.034511	0.019498	0.019497	7.140647
4	0.011754	0.011753	0.011754	0.011752	0.011753	0.011752	0.011754	0.011752	0.011753	99.894222
...	...	...	...	...	...	...	...	...	...	...
70	0.013282	0.013284	0.013282	0.013281	0.013282	0.013282	0.013282	0.013281	0.013282	99.880461
71	0.013407	46.632949	0.013407	53.259794	0.013407	0.013407	0.013408	0.013406	0.013407	0.013407
72	0.019965	99.820326	0.019963	0.019964	0.019963	0.019963	0.019964	0.019963	0.019963	0.019967
73	0.028498	0.028497	0.028497	0.028494	1.266543	0.028493	93.560961	0.028494	0.028493	4.973030
74	0.013390	0.013389	55.967734	0.013388	0.013390	0.013389	43.925150	0.013389	0.013390	0.013390

75 rows × 10 columns

Storing topics list as a dataframe

In [31]:
```
topic_indices = list(topics_lda)
topic_subjects = list(topics_list_lda)
topic_desc = pd.DataFrame(topic_indices, topic_subjects).reset_index()
topic_desc_lda = topic_desc.rename(columns = {"index": "topic_desc", 0: "topic"})
topic_desc_lda
```

Out[31]:

	topic_desc	topic
0	ought_class_kid	Topic 1
1	terrorist_iraq_iraqi	Topic 2
2	vietnam_oil_communist	Topic 3
3	hussein_saddam_inspector	Topic 4
4	vietnam_south_voic	Topic 5
5	session_surplus_read	Topic 6
6	communist_agricultur_expenditur	Topic 7
7	veteran_disput_bargain	Topic 8
8	vietnam_januari_abund	Topic 9
9	21st_veteran_class	Topic 10

Merging speeches database with topics dataframe

In [32]:
```
lda_merged = topics_lda.join(sotu)
lda_merged.head(3)
```

Out[32]:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	text	filename	year	pres	id
0	0.012183	0.012182	0.012183	0.012182	0.012182	0.012182	99.890359	0.012183	0.012183	0.012183	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	0.014840	0.014839	0.014840	0.014839	0.014839	0.014839	99.866447	0.014839	0.014839	0.014839	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	23.976947	0.030310	14.822932	0.030308	0.030310	0.030309	49.192545	0.030308	11.825722	0.030311	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972

Rearranging dataframe for plotting

In [33]:
```
lda_1 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 1"]].copy()
lda_1["topic"] = "Topic 1"
lda_1 = lda_1.rename(columns = {"Topic 1": "factor"})

lda_2 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 2"]].copy()
lda_2["topic"] = "Topic 2"
lda_2 = lda_2.rename(columns = {"Topic 2": "factor"})

lda_3 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 3"]].copy()
lda_3["topic"] = "Topic 3"
lda_3 = lda_3.rename(columns = {"Topic 3": "factor"})

lda_4 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 4"]].copy()
lda_4["topic"] = "Topic 4"
lda_4 = lda_4.rename(columns = {"Topic 4": "factor"})

lda_5 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 5"]].copy()
lda_5["topic"] = "Topic 5"
lda_5 = lda_5.rename(columns = {"Topic 5": "factor"})

lda_6 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 6"]].copy()
lda_6["topic"] = "Topic 6"
lda_6 = lda_6.rename(columns = {"Topic 6": "factor"})

lda_7 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 7"]].copy()
lda_7["topic"] = "Topic 7"
lda_7 = lda_7.rename(columns = {"Topic 7": "factor"})

lda_8 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 8"]].copy()
lda_8["topic"] = "Topic 8"
lda_8 = lda_8.rename(columns = {"Topic 8": "factor"})

lda_9 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 9"]].copy()
lda_9["topic"] = "Topic 9"
lda_9 = lda_9.rename(columns = {"Topic 9": "factor"})

lda_10 = lda_merged[["text", "filename", "year", "pres", "id", "Topic 10"]].copy()
lda_10["topic"] = "Topic 10"
lda_10 = lda_10.rename(columns = {"Topic 10": "factor"})

```

```
lda_master = pd.concat([lda_1, lda_2, lda_3, lda_4, lda_5, lda_6, lda_7, lda_8, lda_9, lda_10], ignore_index = True)
lda_master = pd.merge(lda_master, topic_desc_lda, on = "topic", how = "left")

lda_master
```

Out[33]:

	text	filename	year	pres	id	factor	topic	topic_desc
0	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955	0.012183	Topic 1	ought_class_kid
1	Mr. President, Mr. Speaker, Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954	0.014840	Topic 1	ought_class_kid
2	Mr. Speaker, Mr. President, my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972	23.976947	Topic 1	ought_class_kid
3	Mr. President, Mr. Speaker, Members of the Hou...	sotu/02-17-1993.txt	1993	clinton	clinton-1993	63.740839	Topic 1	ought_class_kid
4	Madam Speaker, Madam Vice President—no Preside...	sotu/04-28-2021.txt	2021	biden	biden-2021	0.011754	Topic 1	ought_class_kid
...	...	...	...	...	...	...	...	...
745	Mr. Speaker, Mr. Vice President, Members of Co...	sotu/01-30-2018.txt	2018	trump	trump-2018	99.880461	Topic 10	21st_veteran_class
746	Mr. Speaker, Vice President Cheney, Members of...	sotu/01-28-2003.txt	2003	bush jr.	bush jr.-2003	0.013407	Topic 10	21st_veteran_class
747	Thank you very much. Mr. Speaker, Vice Preside...	sotu/01-29-2002.txt	2002	bush jr.	bush jr.-2002	0.019967	Topic 10	21st_veteran_class
748	Mr. President, Mr. Speaker, Members of the Con...	sotu/01-08-1951.txt	1951	truman	truman-1951	4.973030	Topic 10	21st_veteran_class
749	Mr. Vice President, my old colleague from Mass...	sotu/01-11-1962.txt	1962	kennedy	kennedy-1962	0.013390	Topic 10	21st_veteran_class

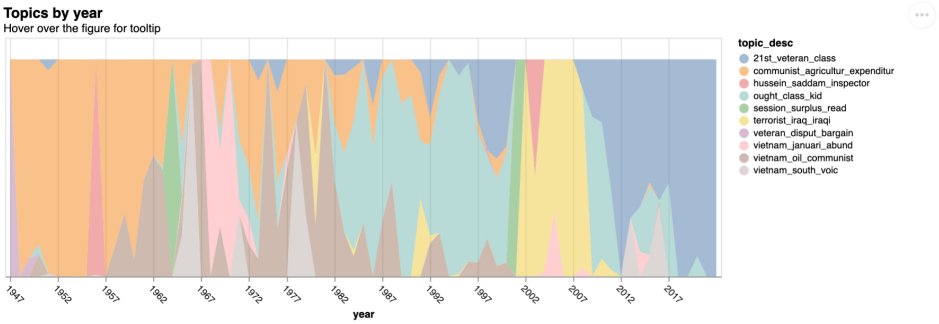
750 rows × 8 columns

Plotting steamgraph

In [34]:

```
alt.Chart(lda_master).mark_area(opacity = 0.5).encode(
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    alt.Y("sum(factor):Q", stack = "center", axis = None),
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    tooltip = ["pres", "year", "topic_desc"]
).configure_axis(
    grid = True
).configure_axisX(
    labelAngle = 45
).properties(
    width = 750,
    height = 250,
    title = {
      "text": "Topics by year",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).interactive()
```

Out[34]:

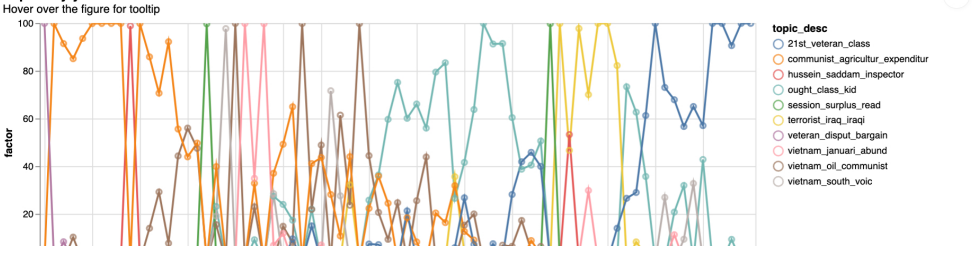


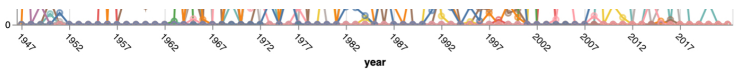
Plotting line chart

In [35]:

```
lines = alt.Chart(lda_master).mark_line().encode(
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    y = "factor",
)

points = alt.Chart(lda_master).mark_point().encode(
    alt.Color("topic_desc", scale = alt.Scale(scheme = "tableau10")),
    alt.X("year:O", axis = alt.Axis(values = list(range(1947, 2022, 5)))),
    y = "factor",
    tooltip = ["pres", "year", "topic_desc"]
)

chart = lines + points

chart.configure_axis(
    grid = True
).configure_axisX(
    labelAngle = 45
).properties(
    width = 750,
    height = 250,
    title = {
      "text": "Topics by year",
      "subtitle": "Hover over the figure for tooltip",
    }
).configure_title(
    fontSize = 15,
    anchor = "start",
).interactive()
```

Out[35]:





Comparing NMF and LDA models:

The topics generated by both models are nearly the identical but the ones from NMF seem to make a bit more sense.

Note: When applying topic modeling, sometimes which method is chosen boils down to a subjective decision. GridSearchCV can be used to deploy an algorithm to suggest the best parameters but that too is not always foolproof. The primary question should always be "What makes sense?"

Gensim modeling

Setting dataframe

In [36]:
```
sotu.text = sotu.text.str.replace("[^A-Za-z ]", " ")

sotu.head(3)
```

```
/var/folders/8_/vf0g3np57bdb6_x59hshmtcr0000gn/T/ipykernel_49503/1114454686.py:1: FutureWarning: The default value of regex will change from True to False in a future version.
  sotu.text = sotu.text.str.replace("[^A-Za-z ]", " ")
```

Out[36]:

	text	filename	year	pres	id
0	Mr President Mr Speaker Members of the Con...	sotu/01-06-1955.txt	1955	eisenhower	eisenhower-1955
1	Mr President Mr Speaker Members of the Eig...	sotu/01-07-1954.txt	1954	eisenhower	eisenhower-1954
2	Mr Speaker Mr President my colleagues in t...	sotu/01-20-1972.txt	1972	nixon	nixon-1972

Deploying LSI/NMF modeling with Gensim

In [37]:
```
### imports modules
from gensim.utils import simple_preprocess

### stores text of speeches and pre-processes
texts = sotu.text.apply(simple_preprocess)

### imports modules
from gensim import corpora

dictionary = corpora.Dictionary(texts)
dictionary.filter_extremes(no_below = 5, no_above = 0.4)
corpus = [dictionary.doc2bow(text) for text in texts]

### imports modules
from gensim import models

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

### assigns number of topics
n_topics = 10

### builds a model
nmf_model = models.LsiModel(corpus_tfidf,
                            id2word = dictionary,
                            num_topics = n_topics)

### displays topic results
nmf_model.print_topics()
```

Out[37]:
```
[(0,
  '0.085*"iraq" + 0.067*"vietnam" + 0.065*"oil" + 0.064*"terrorists" + 0.058*"kids" + 0.057*"recommend" + 0.055*"companies" + 0.054*"ought" + 0.053*"communist"'),
 (1,
  '0.135*"iraq" + 0.104*"terrorists" + -0.096*"expenditures" + -0.093*"recommend" + 0.087*"al" + -0.082*"peoples" + -0.082*"communist" + 0.081*"iraqi" + 0.079*"qaida"'),
 (2,
  '0.325*"iraq" + 0.262*"iraqi" + 0.178*"terrorists" + 0.176*"saddam" + 0.169*"iraqis" + 0.168*"terror" + 0.154*"hussein" + 0.134*"al" + 0.129*"ought"'),
 (3,
  '-0.247*"ryan" + -0.209*"isis" + -0.162*"gang" + -0.119*"heroes" + -0.119*"totally" + -0.118*"immigration" + -0.115*"incredible" + 0.104*"ouverans" + -0.090*"david"'),
 (4,
  '-0.243*"vietnam" + -0.111*"nicaragua" + 0.103*"folks" + 0.082*"oil" + 0.082*"manufacturing" + -0.078*"drugs" + -0.075*"revolution" + 0.071*71*"veterans" + 0.070*"collective"'),
 (5,
  '-0.202*"ought" + 0.170*"oil" + 0.119*"vietnam" + -0.101*"millennium" + -0.097*"saddam" + -0.096*"bosnia" + -0.082*"tobacco" + -0.081*"gradein" + -0.073*"americorps"'),
```

In [38]:
```
### assigns number of words per topic
n_words = 10

topic_words = pd.DataFrame({})

### computes dataframe with results
for i, topic in enumerate(nmf_model.get_topics()):
    top_feature_ids = topic.argsort()[-n_words:][::-1]
    feature_values = topic[top_feature_ids]
    words = [dictionary[id] for id in top_feature_ids]
    topic_df = pd.DataFrame({"value": feature_values, "word": words, "topic": (i + 1)})
    topic_words = pd.concat([topic_words, topic_df], ignore_index = True)

topic_words.head()
```

Out[38]:

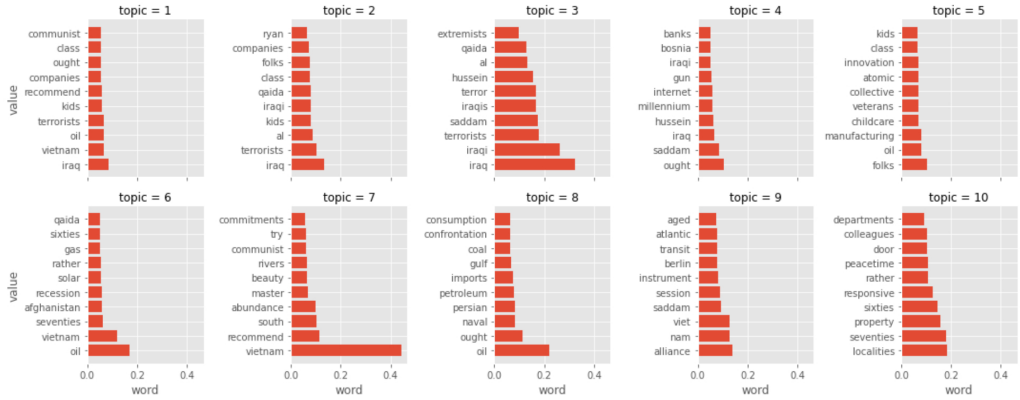
	value	word	topic
0	0.085336	iraq	1
1	0.066523	vietnam	1
2	0.064880	oil	1
3	0.063994	terrorists	1
4	0.057784	kids	1

Visualizing Gensim-LSI/NMF model using Seaborn

In [39]:
```
import matplotlib.pyplot as plt
plt.style.use("ggplot")
import seaborn as sns
```

```
g = sns.FacetGrid(topic_words, col = "topic", col_wrap = 5, sharey = False)
g.map(plt.barh, "word", "value")
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x7fa305cefa90>



Deploying LDA model with Gensim

In [40]:
```
### imports modules
from gensim.utils import simple_preprocess

### stores text of speeches and pre-processes
texts = sotu.text.apply(simple_preprocess)

### imports modules
from gensim import corpora

dictionary = corpora.Dictionary(texts)
dictionary.filter_extremes(no_below = 5, no_above = 0.4, keep_n = 2000)
corpus = [dictionary.doc2bow(text) for text in texts]

### assigns number of topics
from gensim import models

### builds a model
n_topics = 10
lda_model = models.LdaModel(corpus = corpus, num_topics = n_topics)

### displays topic results
lda_model.print_topics()
```

Out[40]:
```
[(0,
  '0.004*"1497" + 0.003*"1290" + 0.002*"1084" + 0.002*"1404" + 0.002*"623" + 0.002*"1514" + 0.002*"1334" + 0.002*"1021" + 0.002*"1531" + 0.002
 (1,
  '0.003*"1514" + 0.003*"1497" + 0.003*"980" + 0.003*"425" + 0.002*"1290" + 0.002*"623" + 0.002*"1099" + 0.002*"189" + 0.002*"1404" + 0.002*"8
 (2,
  '0.004*"1497" + 0.003*"1021" + 0.003*"1514" + 0.003*"1404" + 0.003*"623" + 0.003*"559" + 0.002*"262" + 0.002*"1229" + 0.002*"1334" + 0.002*"
 (3,
  '0.003*"425" + 0.003*"1404" + 0.003*"1084" + 0.002*"1497" + 0.002*"559" + 0.002*"84" + 0.002*"1290" + 0.002*"79" + 0.002*"980" + 0.002*"189"
 (4,
  '0.005*"1497" + 0.004*"1404" + 0.003*"1514" + 0.002*"1229" + 0.002*"1721" + 0.002*"456" + 0.002*"425" + 0.002*"1290" + 0.002*"1364" + 0.002*
 (5,
  '0.003*"623" + 0.003*"1497" + 0.002*"1229" + 0.002*"1514" + 0.002*"1404" + 0.002*"1721" + 0.002*"1084" + 0.002*"1334" + 0.002*"1230" + 0.002
 (6,
  '0.003*"1290" + 0.003*"79" + 0.002*"1021" + 0.002*"425" + 0.002*"623" + 0.002*"1045" + 0.002*"1229" + 0.002*"980" + 0.002*"1084" + 0.002*"15
 (7,
  '0.003*"1514" + 0.002*"1404" + 0.002*"189" + 0.002*"1497" + 0.002*"1721" + 0.002*"623" + 0.002*"7" + 0.002*"1531" + 0.002*"425" + 0.002*"102
 (8,
  '0.003*"1497" + 0.003*"1404" + 0.002*"623" + 0.002*"425" + 0.002*"1290" + 0.002*"1021" + 0.002*"189" + 0.002*"1084" + 0.002*"1045" + 0.002*"
```

In [41]:
```
### assigns number of words per topic
n_words = 10

topic_words = pd.DataFrame({})

### computes dataframe with results
for i, topic in enumerate(lda_model.get_topics()):
    top_feature_ids = topic.argsort()[-n_words:][::-1]
    feature_values = topic[top_feature_ids]
    words = [dictionary[id] for id in top_feature_ids]
    topic_df = pd.DataFrame({"value": feature_values, "word": words, "topic": (i + 1)})
    topic_words = pd.concat([topic_words, topic_df], ignore_index = True)

topic_words.head()
```

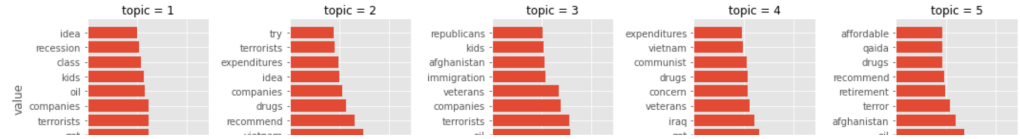
Out[41]:

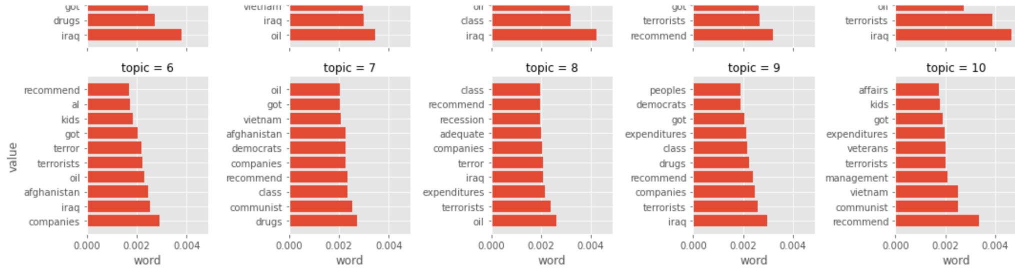
	value	word	topic
0	0.003800	iraq	1
1	0.002715	drugs	1
2	0.002465	got	1
3	0.002459	terrorists	1
4	0.002437	companies	1

Visualizing Gensim-LDA model using Seaborn

In [42]:
```
import matplotlib.pyplot as plt
plt.style.use("ggplot")
import seaborn as sns

g = sns.FacetGrid(topic_words, col = "topic", col_wrap = 5, sharey = False)
g.map(plt.barh, "word", "value")
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x7fa30b275e50>



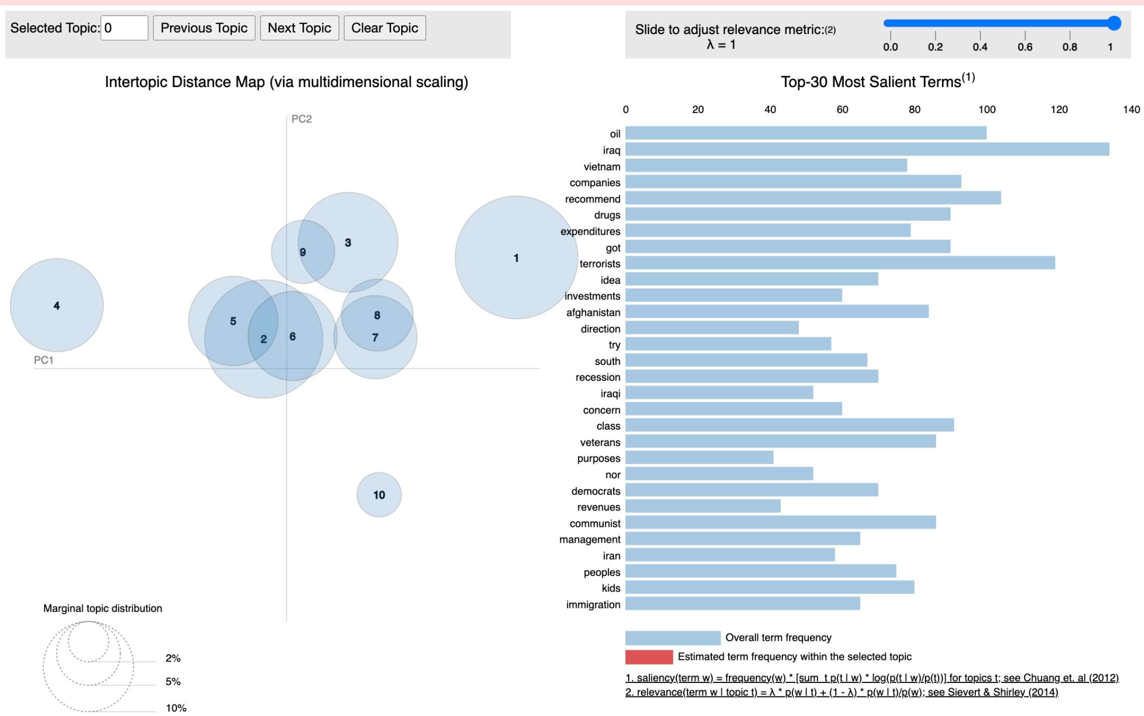


Visualizing Gensim-LDA model using PyLDAvis

In [43]:
```
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model, corpus, dictionary)
vis
```

```
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/pyLDAvis/_prepare.py:246: FutureWarning: In a future version of pandas all argume
nts of DataFrame.drop except for the argument 'labels' will be keyword-only.
  default_term_info = default_term_info.sort_values(
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/shirshodasgupta/Documents/Python/anaconda3/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in fav
our of importlib; see the module's documentation for alternative uses
  from imp import reload
```

Out[43]:



In [ ]:
```

```