Coding_Presentation

January 12, 2024

1 Week 2 Coding Presentation (Sam Cong)

1.1 Load modules

[1]:
```
import requests
import pandas as pd
import matplotlib.pyplot as plt
import wordcloud
import numpy as np
import seaborn as sns
import sklearn.manifold
import spacy
import nltk

%matplotlib inline
```

1.2 Load helper functions for this homework

[2]:
```
import helper_functions
```

1.3 Exercise 1

Construct cells immediately below this that input and interrogate a corpus relating to your anticipated final project. This could include one of the Davies corpora or some other you have constructed. Turn your text into an nltk `Text` object, and explore all of the features examined above, and others that relate to better understanding your corpus in relation to your research question.

[3]:
```
nlp = spacy.load("en_core_web_sm")
```

For this week's exercise, I used Semantic Scholar API (https://github.com/allenai/s2-folks) to extract the abstract of a small sample of psychology academic papers.

Note: Serving merely roughtly as a skeleton code for the (tentatative) final project, the following code did not impose sufficient filtering conditions (e.g., article type, publication year) to scrape the articles.

[4]:
```
# The S2 API Key used to fetch articles
api_key = 'KOyXiGdu4t38zFJFFyuKF5jQGBGdxCSfDICseepg'
```

```python
headers = {"x-api-key": api_key}
```

[5]:
```python
# Define the paper search endpoint URL
url = 'https://api.semanticscholar.org/graph/v1/paper/search'

# Define the search parameters
query_params = {
    'query': 'relevant search term',
    'fieldsOfStudy': 'Psychology',  # Search term
    'fields': 'title,authors,abstract',  # Fetching title and abstract
    'limit': 100  # Number of results to return
}

response = requests.get(url, headers=headers, params=query_params)

abstract_corpus_df = {"Title": [],
                      "Author": [],
                      "Abstract": []}

# Check if the request was successful
if response.status_code == 200:
    papers = response.json().get('data', [])
    for paper in papers:
        title = paper.get('title')
        author = paper.get('authors')
        abstract = paper.get('abstract')

        # Skip articles with missing info
        if not title or not author or not abstract:
            continue

        author_names = ', '.join([a['name'] for a in author])

        abstract_corpus_df["Title"].append(title)
        abstract_corpus_df["Author"].append(author_names)
        abstract_corpus_df["Abstract"].append(abstract)
else:
    print(f"Request failed with status code {response.status_code}")

abstract_corpus_df = pd.DataFrame(abstract_corpus_df)

abstract_corpus_df.head()
```

[5]:
```
                                               Title  \
0  The Control of Single-color and Multiple-color…
1  Non-pharmacologic and pharmacologic treatments…
2  Confirmation bias in information search, inter…
```

```
3  Meta-analysis on the long-term effectiveness o…
4  On the search for a selective and retroactive …

                                              Author  \
0                   A. Grubert, N. Carlisle, M. Eimer
1  K. Atchison, J. Watt, Delaney Ewert, A. Toohey…
2                          Dáša Vedejová, V. Čavojová
3  A. Hilbert, D. Petroff, S. Herpertz, R. Pietro…
4                               F. Kalbe, L. Schwabe

                                            Abstract
0  The question whether target selection in visua…
1  BACKGROUND\nolder adults living in long-term c…
2  Abstract Confirmation bias is often used as an…
3  OBJECTIVE\nLong-term effectiveness is a critic…
4  Storing motivationally salient experiences pre…
```

Update the clean_raw_text function about the unicode part:

[6]:
```python
abstract_corpus_df['Cleaned_Abstract'] = abstract_corpus_df['Abstract'].
    ↪apply(lambda x: helper_functions.clean_raw_text_updated([x])[0])
abstract_corpus_df['Cleaned_Abstract'].head()
```

[6]:
```
0    The question whether target selection in visua…
1    BACKGROUND\nolder adults living in long-term c…
2    Abstract Confirmation bias is often used as an…
3    OBJECTIVE\nLong-term effectiveness is a critic…
4    Storing motivationally salient experiences pre…
Name: Cleaned_Abstract, dtype: object
```

[7]:
```python
abstract_corpus_df["Abstract_Token"] = abstract_corpus_df['Cleaned_Abstract'].
    ↪apply(lambda x: helper_functions.word_tokenize(x))
abstract_corpus_df["Abstract_Token"].head()
```

[7]:
```
0    [The, question, whether, target, selection, in…
1    [BACKGROUND, older, adults, living, in, long, …
2    [Abstract, Confirmation, bias, is, often, used…
3    [OBJECTIVE, Long, term, effectiveness, is, a, …
4    [Storing, motivationally, salient, experiences…
Name: Abstract_Token, dtype: object
```

[8]:
```python
# abstract_corpus_token as a whole
abstract_corpus_token = abstract_corpus_df["Abstract_Token"].sum()
abstract_corpus_token[:50]
```

[8]:
```
['The',
 'question',
 'whether',
```

```
'target',
'selection',
'in',
'visual',
'search',
'can',
'be',
'effectively',
'controlled',
'by',
'simultaneous',
'attentional',
'templates',
'for',
'multiple',
'features',
'is',
'still',
'under',
'dispute',
'We',
'investigated',
'whether',
'multiple',
'color',
'attentional',
'guidance',
'is',
'possible',
'when',
'target',
'colors',
'remain',
'constant',
'and',
'can',
'thus',
'be',
'represented',
'in',
'long',
'term',
'memory',
'but',
'not',
'when',
'they']
```

[9]:
```python
# Construct the nltk Text object
abstract_corpus_Text = nltk.Text(abstract_corpus_token)
```

After constructing the nltk Text object, I randomly selected the keyword memory, which is big topic in psychology, and mainly used it for exploratory analysis with the following methods: - text.count(): counting the number of times this word appears in the text; - text.concordance(): finding all occurrences of the target word in the text and displaying them accompanied by their immediate context; - text.similar(): finding other words which appear in the same contexts as the specified word; listing most similar words first (similarity for distributional similarity); - text.common_contexts(): finding contexts where the specified words appear; listing most frequent common contexts first; - text.dispersion_plot: producing a plot showing the distribution of the words through the text; - text.collocations(): printing collocations derived from the text, ignoring stopwords.

Reference: https://www.nltk.org/api/nltk.text.Text.html

[10]:
```python
abstract_corpus_Text.count("memory")
```

[10]:
```
74
```

[11]:
```python
abstract_corpus_Text.concordance('memory', lines=10)
```

```
Displaying 10 of 74 matches:
an thus be represented in long term memory but not when they change frequently
ntly and have to be held in working memory Participants searched for one two o
in amplitude as a function of color memory load in variable color blocks which
 target colors were held in working memory In constant color blocks the CDA wa
 were primarily stored in long term memory N2pc components to targets were mea
and can be represented in long term memory and when they change across trials
re have to be maintained in working memory BACKGROUND older adults living in l
earches evidence interpretation and memory recall are the three main component
s did not show confirmation bias in memory recall as there was no difference i
riences preferentially in long term memory is generally adaptive Although such
```

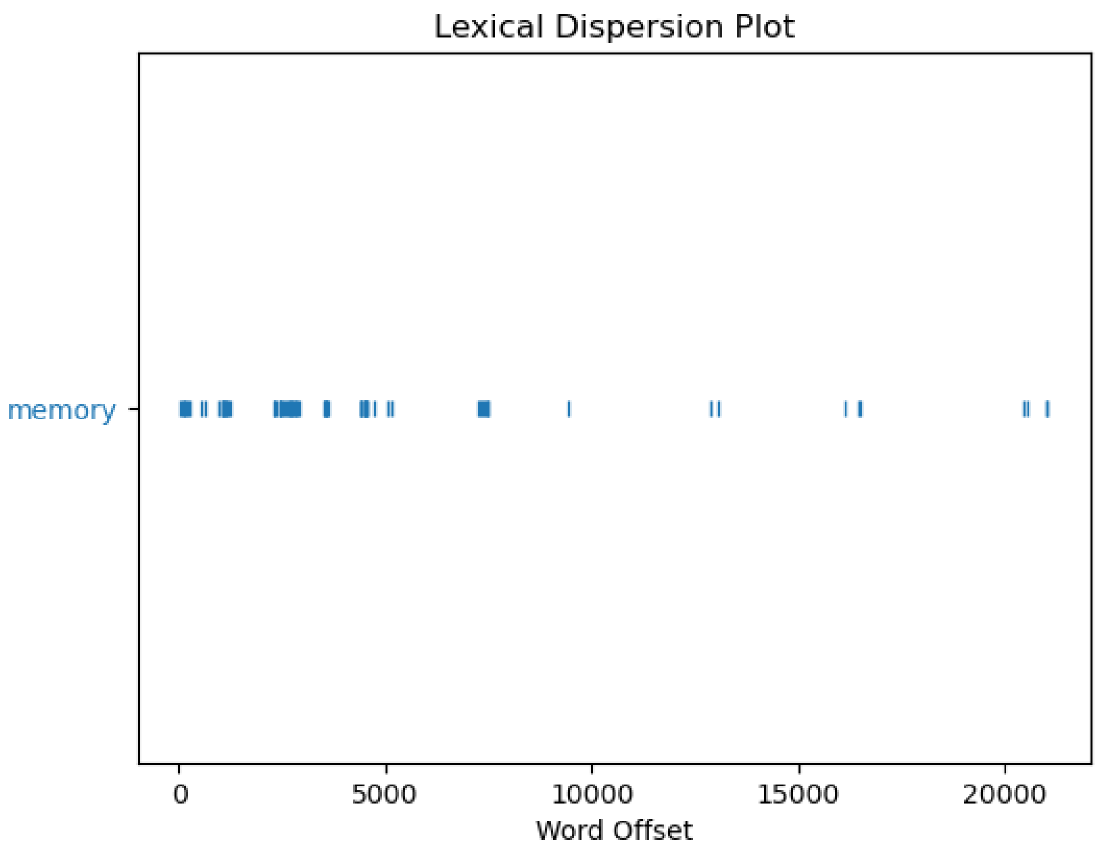
[12]:
```python
abstract_corpus_Text.similar("memory", num=10)
```

```
visual effectiveness cognitive follow illness in is color same care
```

[13]:
```python
abstract_corpus_Text.common_contexts(['memory'], num=10)
```

```
working_capacity term_is term_but working_in retroactive_enhancement
retroactive_effect working_to term_may term_in working_and
```

[14]:
```python
abstract_corpus_Text.dispersion_plot(["memory"])
```



[15]:
```
abstract_corpus_Text.collocations(num=10)
```

```
long term; embryo transfer; working memory; short term; inclusion
criteria; mental health; term memory; rapid cycling; bipolar disorder;
double embryo
```

1.4 Exercise 2

Construct cells immediately below this that filter, stem and lemmatize the tokens in your corpus, and then creates plots (with titles and labels) that map the word frequency distribution, word probability distribution, and at least two conditional probability distributions that help us better understand the social and cultural game underlying the production of your corpus. Create a wordl of words (or normalized words) and add a few vague comments about what mysteries are revealed through it.

First, I will create normalized tokens for the abstract corpus.

[16]:
```
abstract_countsDict = {}
for word in abstract_corpus_token:
    # For filtering words, I chose to use lowercase words
    word_lowercase = word.lower()
```

```
    if word_lowercase in abstract_countsDict:
        abstract_countsDict[word_lowercase] += 1
    else:
        abstract_countsDict[word_lowercase] = 1
abstract_countsDict = sorted(abstract_countsDict.items(), key = lambda x :␣
 ↪x[1], reverse = True)
abstract_countsDict[:20]
```

[16]:
```
[('the', 882),
 ('and', 783),
 ('of', 773),
 ('in', 507),
 ('to', 465),
 ('a', 324),
 ('for', 258),
 ('with', 228),
 ('were', 202),
 ('on', 175),
 ('that', 168),
 ('studies', 157),
 ('search', 156),
 ('is', 152),
 ('or', 150),
 ('term', 140),
 ('was', 135),
 ('are', 104),
 ('long', 95),
 ('we', 94)]
```

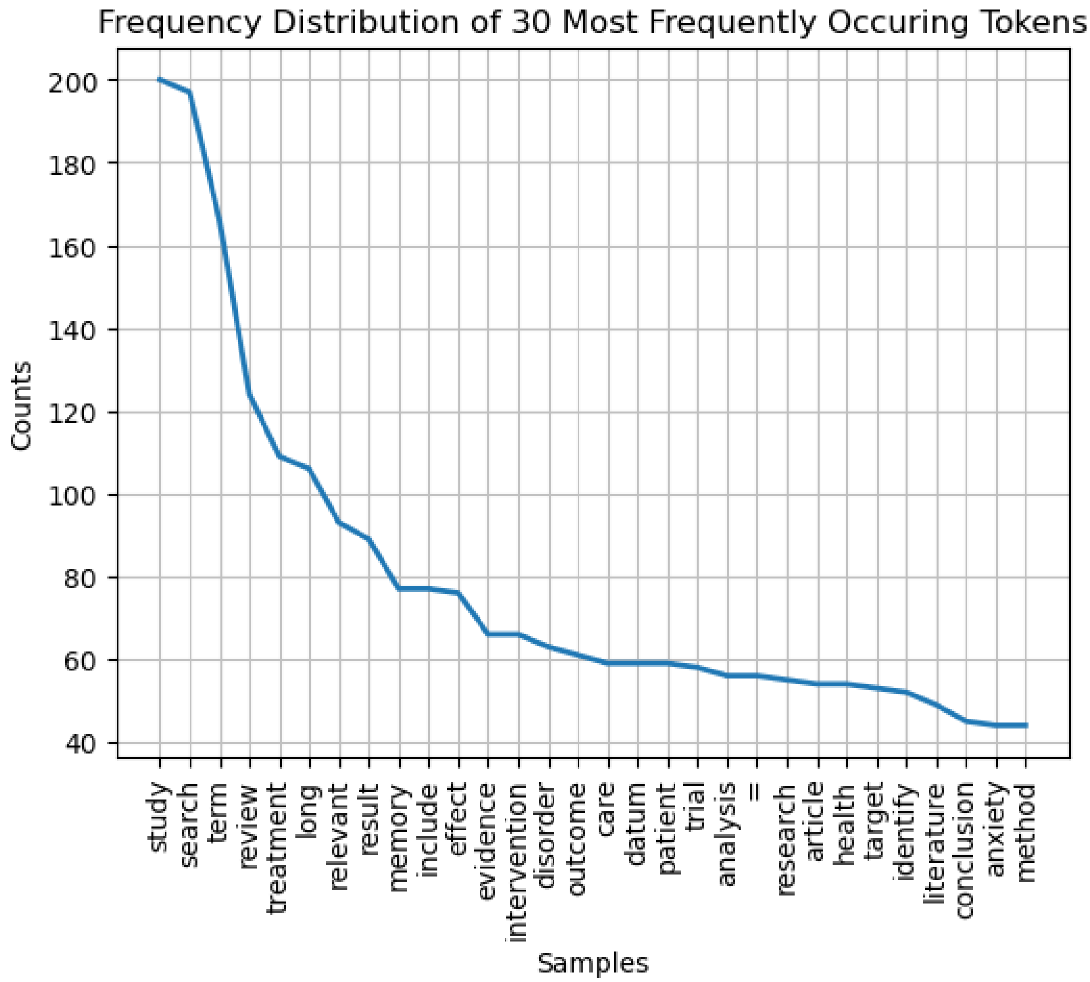
[17]:
```
abstract_corpus_token_normalized = helper_functions.
 ↪normalizeTokens(abstract_corpus_token)
```

[18]:
```
print("Number of tokens for the abstract corpus: {}".
 ↪format(len(abstract_corpus_token)))
print("Number of normalized tokens for the abstract corpus: {}".
 ↪format(len(abstract_corpus_token_normalized)))
```

```
Number of tokens for the abstract corpus: 21069
Number of normalized tokens for the abstract corpus: 12611
```

Then, I will construct the word frequency distribution and word probability distribution.

[19]:
```
abstract_corpus_fdist = nltk.FreqDist(abstract_corpus_token_normalized)
abstract_corpus_fdist.plot(30, title='Frequency Distribution of 30 Most␣
 ↪Frequently Occuring Tokens', cumulative=False)
```

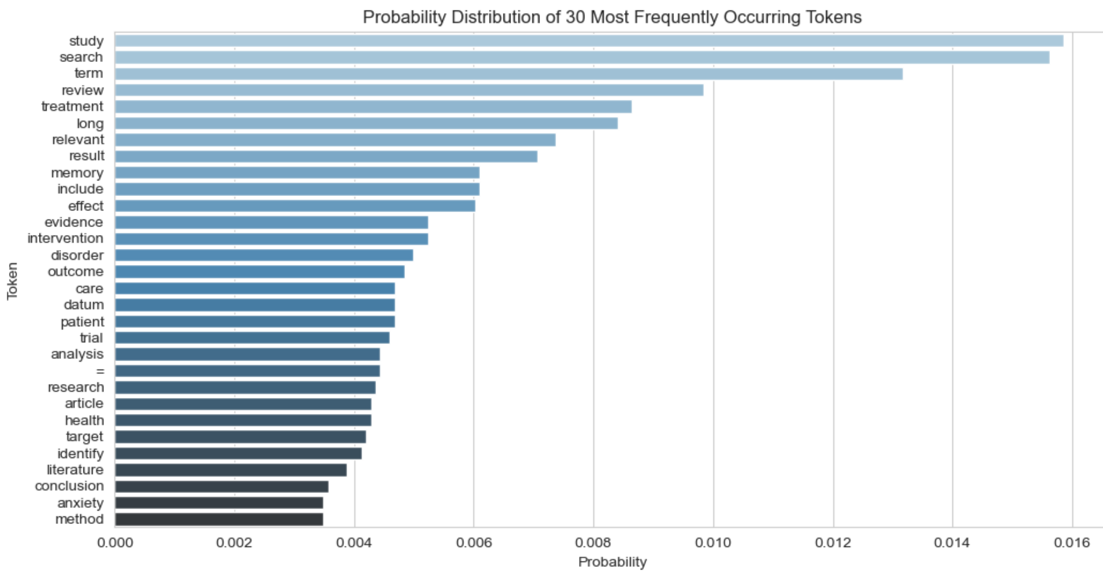


[19]: <Axes: title={'center': 'Frequency Distribution of 30 Most Frequently Occuring Tokens'}, xlabel='Samples', ylabel='Counts'>

[20]:
```
total_token_count = len(abstract_corpus_token_normalized)

# Here, I choose the 30 most frequently occuring tokrns
sample_token = sorted(abstract_corpus_fdist.items(),
                      key=lambda item: item[1], reverse=True)[:30]

# Calculate probabilities
sample_token_list, prob_list = zip(*[(token, freq / total_token_count) for␣
 ↪token, freq in sample_token])
df = pd.DataFrame({
    'Token': sample_token_list,
    'Probability': prob_list
})
```

```
sns.set_style("whitegrid")
plt.figure(figsize=(12, 6))
sns.barplot(x='Probability', y='Token', data=df, palette="Blues_d", orient="h")
plt.title('Probability Distribution of 30 Most Frequently Occurring Tokens')
```

[20]:
```
Text(0.5, 1.0, 'Probability Distribution of 30 Most Frequently Occurring
Tokens')
```



Then, I will construct two conditional probability distributions.

[21]:
```
abstract_corpus_token_normalized_POS = nltk.
 ↪pos_tag(abstract_corpus_token_normalized)
abstract_corpus_token_normalized_POS[:10]
```

[21]:
```
[('question', 'NN'),
 ('target', 'NN'),
 ('selection', 'NN'),
 ('visual', 'JJ'),
 ('search', 'NN'),
 ('effectively', 'RB'),
 ('control', 'VBP'),
 ('simultaneous', 'JJ'),
 ('attentional', 'JJ'),
 ('template', 'NN')]
```

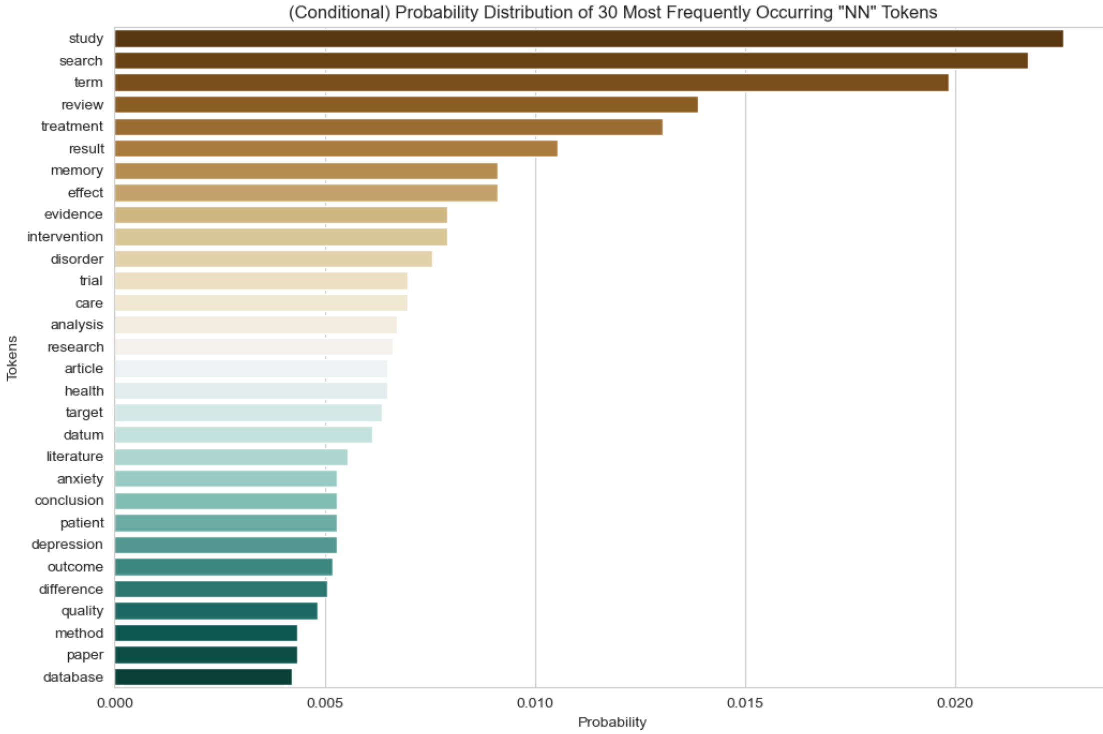
[22]:
```
# Generate the conditional frequency distribution (with POS tag as the feature)
```

```
abstract_corpus_cpfist_WordtoPOS = nltk.ConditionalFreqDist((pos, word) for␣
↪word, pos in abstract_corpus_token_normalized_POS)

# Transform the conditional frequency distribution into conditional probability␣
↪distribution
abstract_corpus_cpdist_WordtoPOS = nltk.
↪ConditionalProbDist(abstract_corpus_cpfist_WordtoPOS, nltk.ELEProbDist)
```

First conditional probability distributions focuses on "NN" POS tag.

```
[23]: nn_token_probs = [(token, abstract_corpus_cpdist_WordtoPOS['NN'].prob(token))
                        for token in abstract_corpus_cpdist_WordtoPOS['NN'].samples()]

      # Sort by probability and take the top 30
      top_30_nn_tokens = sorted(nn_token_probs, key=lambda x: x[1], reverse=True)[:30]

      # Extract words and probabilities for plotting
      token, probabilities = zip(*top_30_nn_tokens)

      df = pd.DataFrame({
          'Token': token,
          'Probability': probabilities
      })

      sns.set_style("whitegrid")
      plt.figure(figsize=(12, 8))
      sns.barplot(x='Probability', y='Token', data=df, palette="BrBG", orient='h')
      plt.title('(Conditional) Probability Distribution of 30 Most Frequently␣
      ↪Occurring "NN" Tokens')
      plt.xlabel('Probability')
      plt.ylabel('Tokens')
```

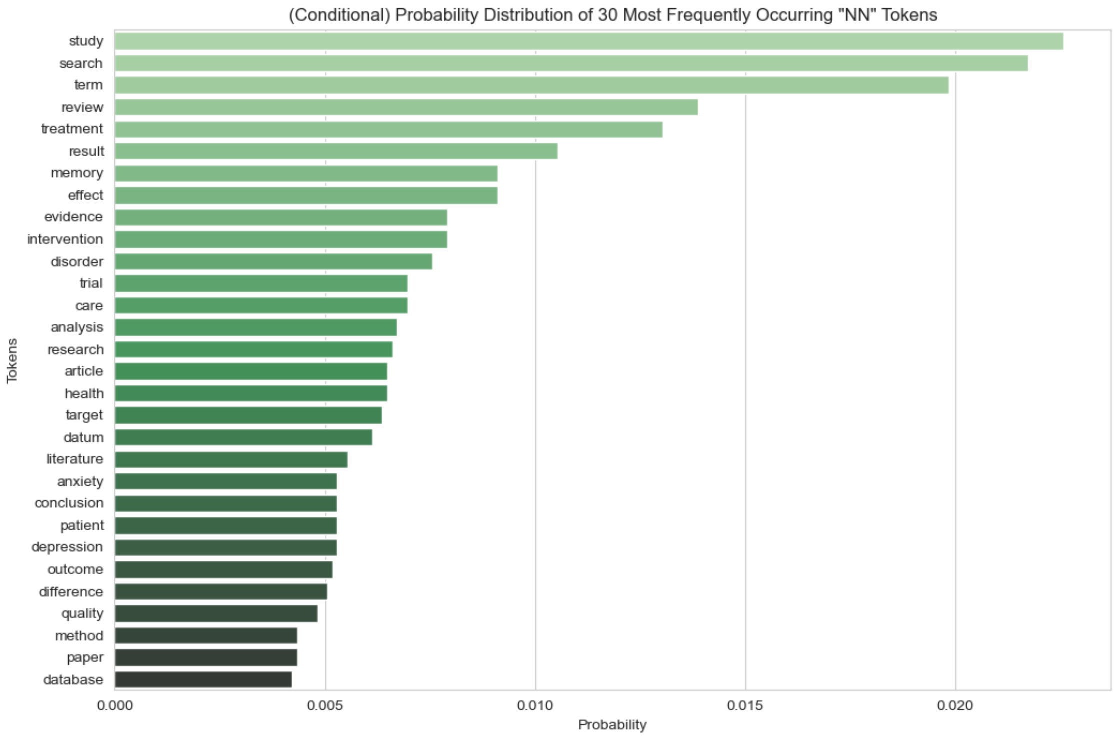
```
[23]: Text(0, 0.5, 'Tokens')
```



Second conditional probability distributions focuses on "JJ" POS tag.

[24]:
```
jj_token_probs = [(token, abstract_corpus_cpdist_WordtoPOS['JJ'].prob(token))
                  for token in abstract_corpus_cpdist_WordtoPOS['JJ'].samples()]

# Sort by probability and take the top 30
top_30_jj_tokens = sorted(nn_token_probs, key=lambda x: x[1], reverse=True)[:30]

token, probabilities = zip(*top_30_jj_tokens)

df = pd.DataFrame({
    'Token': token,
    'Probability': probabilities
})

# Plotting
sns.set_style("whitegrid")
plt.figure(figsize=(12, 8))
sns.barplot(x='Probability', y='Token', data=df, palette="Greens_d", orient='h')
plt.title('(Conditional) Probability Distribution of 30 Most Frequently␣
 ↪Occurring "NN" Tokens')
plt.xlabel('Probability')
plt.ylabel('Tokens')
```

[24]: Text(0, 0.5, 'Tokens')



Finally, create a WORD CLOUD.

[25]:
```
abstract_corpus_token_wc = wordcloud.WordCloud(
    background_color="white", max_words=500, width= 1000, height = 1000,
    mode ='RGBA', scale=.5).generate(' '.join(abstract_corpus_token_normalized))

# Display the generated word cloud:
plt.figure(figsize=(10, 10))
plt.imshow(abstract_corpus_token_wc)
plt.axis("off")
```

[25]: (-0.5, 499.5, 499.5, -0.5)

1.5 Exercise 3

Perform POS tagging on a meaningful (but modest) subset of a corpus associated with your final project. Examine the list of words associated with at least three different parts of speech. Consider conditional associations (e.g., adjectives associated with nouns or adverbs with verbs of interest). What do these distributions suggest about your corpus?

[26]:
```
abstract_corpus_df['Sentence'] = abstract_corpus_df["Cleaned_Abstract"].apply(lambda x: [helper_functions.word_tokenize(s) for s in helper_functions.sent_tokenize(x)])
```

[27]:
```
abstract_corpus_df['Sentence'].head()
```

```
[27]: 0    [[The, question, whether, target, selection, i…
      1    [[BACKGROUND, older, adults, living, in, long,…
      2    [[Abstract, Confirmation, bias, is, often, use…
      3    [[OBJECTIVE, Long, term, effectiveness, is, a,…
      4    [[Storing, motivationally, salient, experience…
      Name: Sentence, dtype: object
```

```
[28]: abstract_corpus_df['POS_Sentence'] = abstract_corpus_df['Sentence'].
        ↪apply(lambda x: helper_functions.tag_sents_pos(x))
```

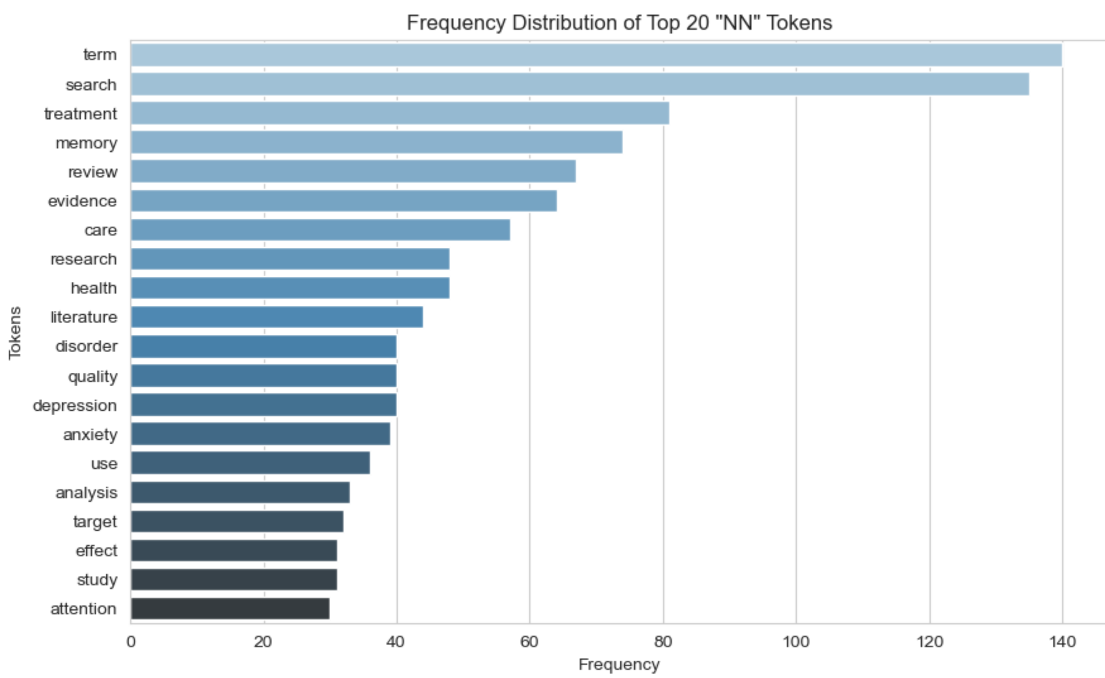
```
[29]: abstract_corpus_df['POS_Sentence'].head()
```

```
[29]: 0    [[(The, DT), (question, NN), (whether, IN), (t…
      1    [[(BACKGROUND, NN), (older, JJR), (adults, NNS…
      2    [[(Abstract, NNP), (Confirmation, NNP), (bias,…
      3    [[(OBJECTIVE, NNP), (Long, JJ), (term, NN), (e…
      4    [[(Storing, VBG), (motivationally, RB), (salie…
      Name: POS_Sentence, dtype: object
```

Examine the list of words associated with at least three POS tags:

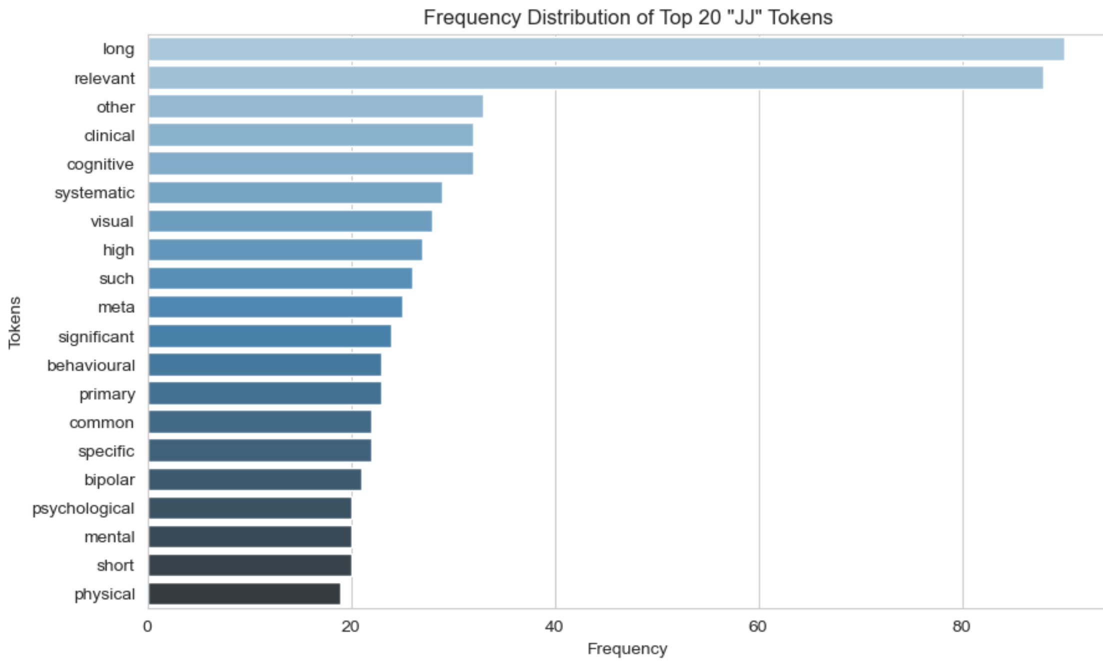
- Common nouns

```
[30]: NN_FreqDist = helper_functions.find_POS(abstract_corpus_df, "NN", 20)
```



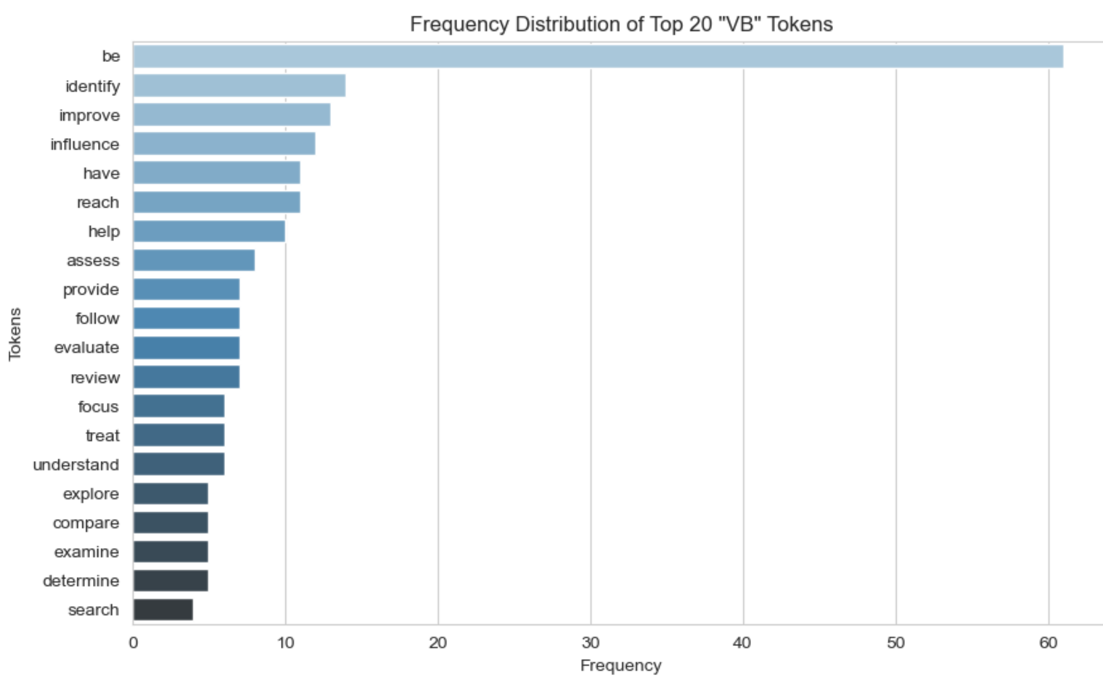
- Adjectives

```
[31]: JJ_FreqDist = helper_functions.find_POS(abstract_corpus_df, "JJ", 20)
```



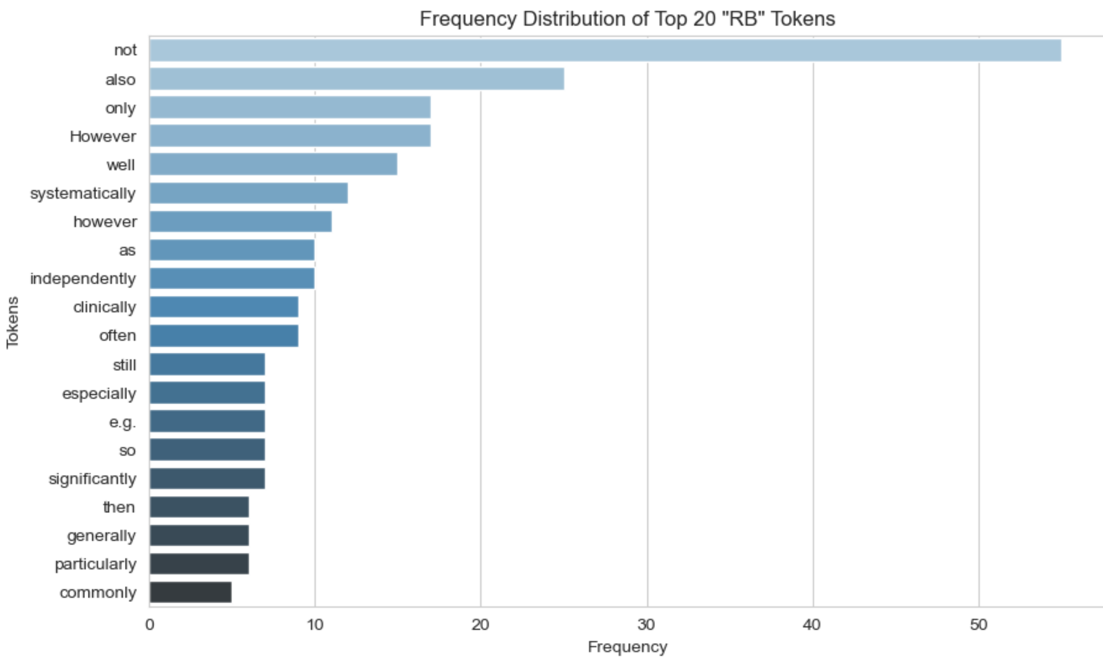
- Verbs

```
[32]: VB_FreqDist = helper_functions.find_POS(abstract_corpus_df, "VB", 20)
```



- Adverbs

[33]:
```
RB_FreqDist = helper_functions.find_POS(abstract_corpus_df, "RB", 20)
```



Consider conditional associations:

- Adjectives and Common Nouns

[34]:
```
jj_nn = helper_functions.find_conditional_associations(abstract_corpus_df,␣
↪("JJ", "NN"))
jj_nn[:10]
```

[34]:
```
[(('long', 'term'), 88),
 (('short', 'term'), 19),
 (('visual', 'search'), 14),
 (('mental', 'health'), 14),
 (('bipolar', 'disorder'), 13),
 (('systematic', 'review'), 10),
 (('meta', 'analysis'), 10),
 (('rapid', 'cycling'), 10),
 (('systematic', 'search'), 9),
 (('high', 'quality'), 9)]
```

- Adverbs and Adjectives

```
[35]: JJ_NN_cfdist = helper_functions.
       ↪find_conditional_associations(abstract_corpus_df, ("RB", "JJ"))
      JJ_NN_cfdist[:10]
```

```
[35]: [(('clinically', 'relevant'), 4),
       (('potentially', 'relevant'), 3),
       (('medically', 'ill'), 3),
       (('clinically', 'significant'), 3),
       (('statistically', 'significant'), 2),
       (('currently', 'available'), 2),
       (('highly', 'flexible'), 2),
       (('spatially', 'specific'), 2),
       (('very', 'low'), 2),
       (('however', 'significant'), 1)]
```

1.6 Exercise 4

Identify statistically significant bigrams and trigrams. Explore whether these collocations are idiomatic and so irreducible to the semantic sum of their component words. You can do this by examination of conditional frequencies (e.g., what else is 'united' besides the 'United States'). If these phrases are idiomatic, what do they suggest about the culture of the world producing them?

```
[36]: whReleases = helper_functions.getGithubFiles('https://api.github.com/repos/
       ↪lintool/GrimmerSenatePressReleases/contents/raw/Whitehouse', maxFiles = 10)
      whReleases['tokenized_text'] = whReleases['text'].apply(lambda x:␣
       ↪helper_functions.word_tokenize(x))
      whReleases['word_counts'] = whReleases['tokenized_text'].apply(lambda x: len(x))
      whReleases['normalized_tokens'] = whReleases['tokenized_text'].apply(lambda x:␣
       ↪helper_functions.normalizeTokens(x))
      whReleases['normalized_tokens_count'] = whReleases['normalized_tokens'].
       ↪apply(lambda x: len(x))
```

```
[37]: whBigrams = nltk.collocations.BigramCollocationFinder.
       ↪from_words(whReleases['normalized_tokens'].sum())
```

```
[38]: bigram_measures = nltk.collocations.BigramAssocMeasures()
```

```
[39]: whTrigrams = nltk.collocations.TrigramCollocationFinder.
       ↪from_words(whReleases['normalized_tokens'].sum())
```

```
[40]: trigram_measures = nltk.collocations.TrigramAssocMeasures()
```

Here, I use student t test to determine whether the ngrams and skipgrams are statistically significant.

In addition, for the critical value for the test, I use the critical value for $\alpha = 0.05$.

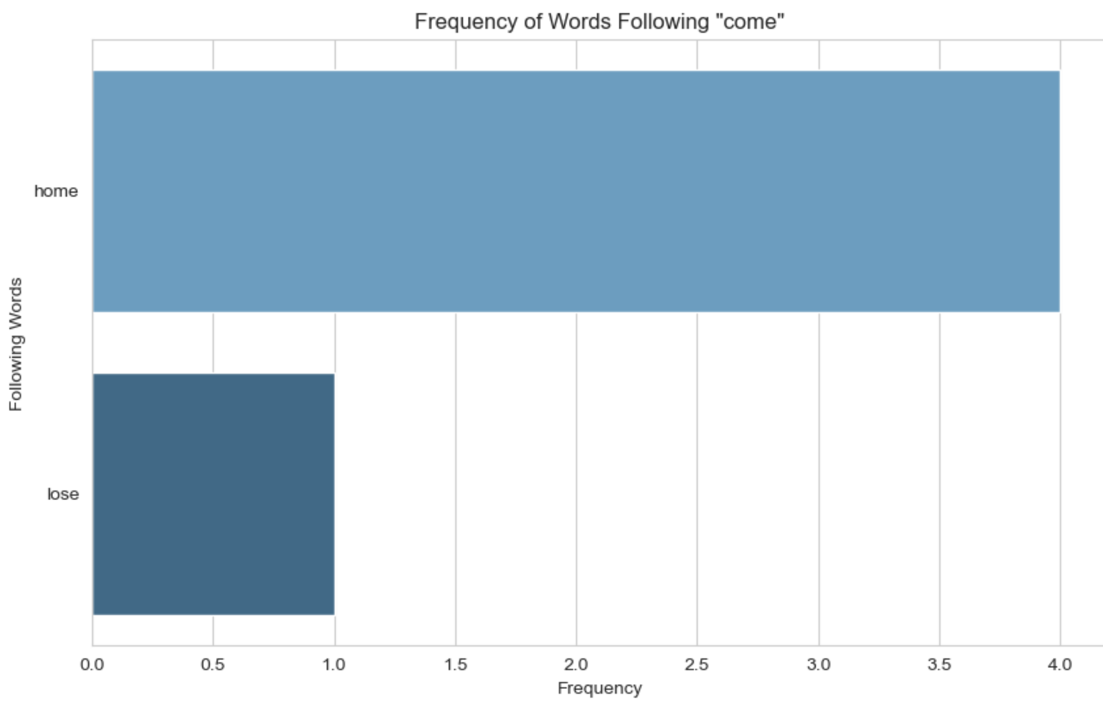
- Statistically significant bigrams

[41]:
```python
t_critical_bigram = helper_functions.find_t_critical(whBigrams.N)

whBigrams_significant = [i for i, j in whBigrams.score_ngrams(bigram_measures.
 ↪student_t) if j >= t_critical_bigram]

whBigrams_significant
```

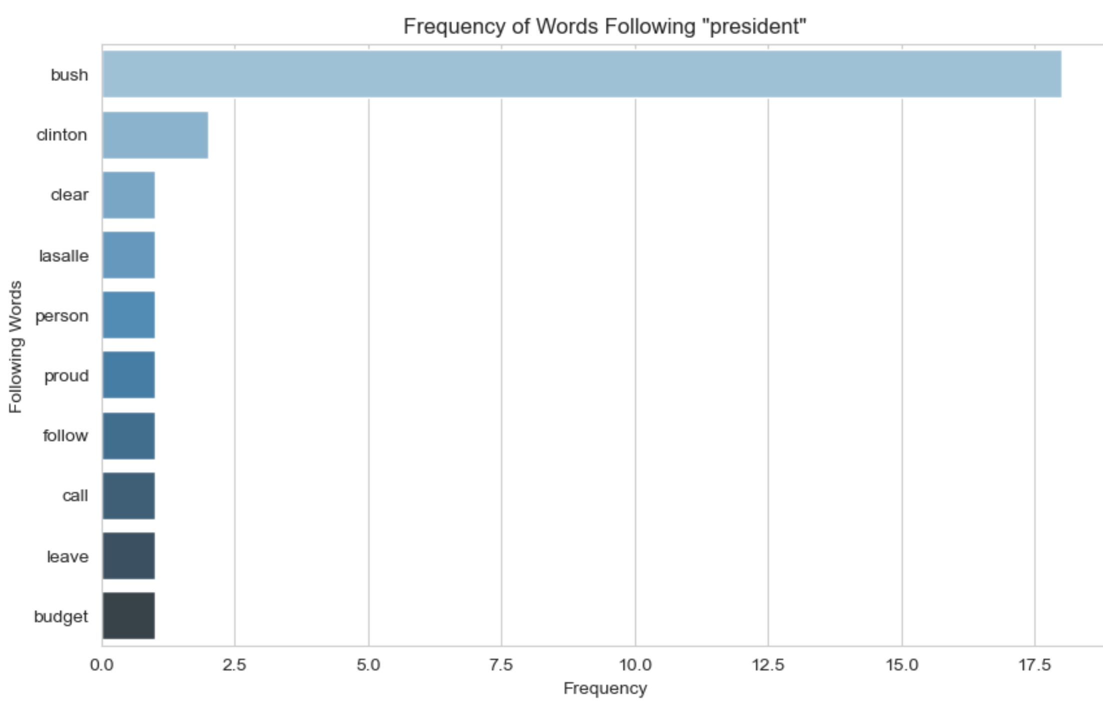
[41]:
```
[('president', 'bush'),
 ('rhode', 'island'),
 ('stem', 'cell'),
 ('sheldon', 'whitehouse'),
 ('whitehouse', 'd'),
 ('d', 'r.i'),
 ('u.s', 'senator'),
 ('bush', 'administration'),
 ('whitehouse', 'say'),
 ('united', 'states'),
 ('senator', 'sheldon'),
 ('american', 'people'),
 ('bring', 'troop'),
 ('troop', 'home'),
 ('cell', 'research'),
 ('sen', 'whitehouse'),
 ('jack', 'reed'),
 ('come', 'home'),
 ('d', 'ri')]
```

[42]:
```python
# Find out conditional frequency distribution for bigrams
bigram_cfd = nltk.ConditionalFreqDist(nltk.
 ↪bigrams(whReleases['normalized_tokens'].sum()))
```

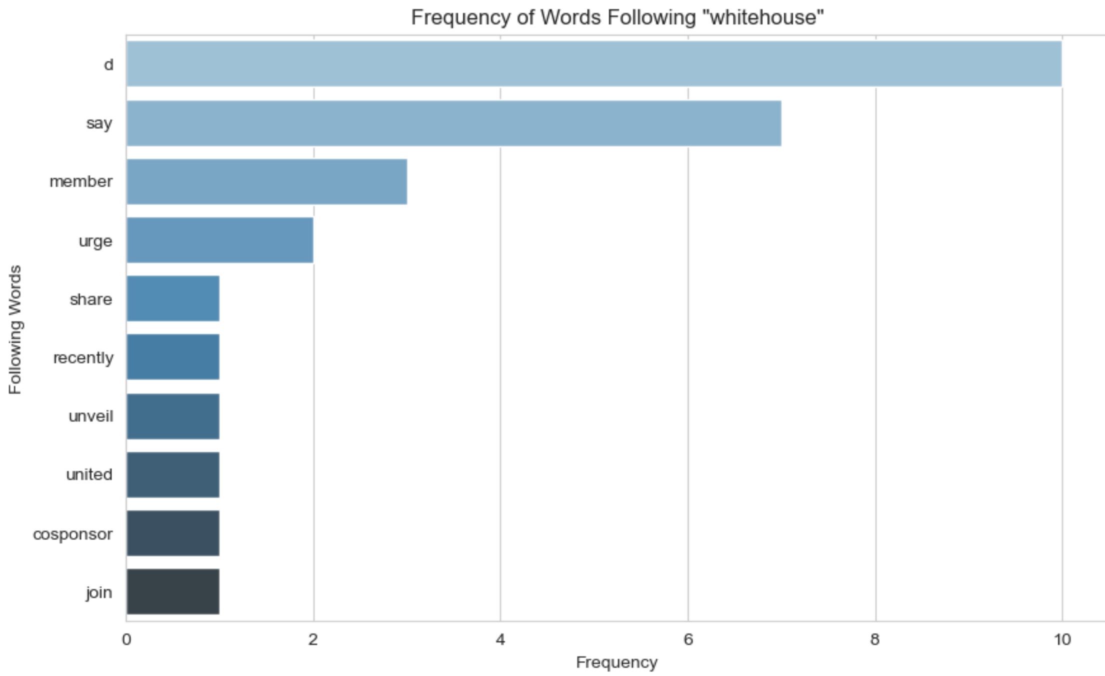
[43]:
```python
helper_functions.plot_ngram_frequencies(("come"), bigram_cfd)
```



[44]:
```
helper_functions.plot_ngram_frequencies(("president"), bigram_cfd)
```



[45]:
```
helper_functions.plot_ngram_frequencies(("whitehouse"), bigram_cfd)
```



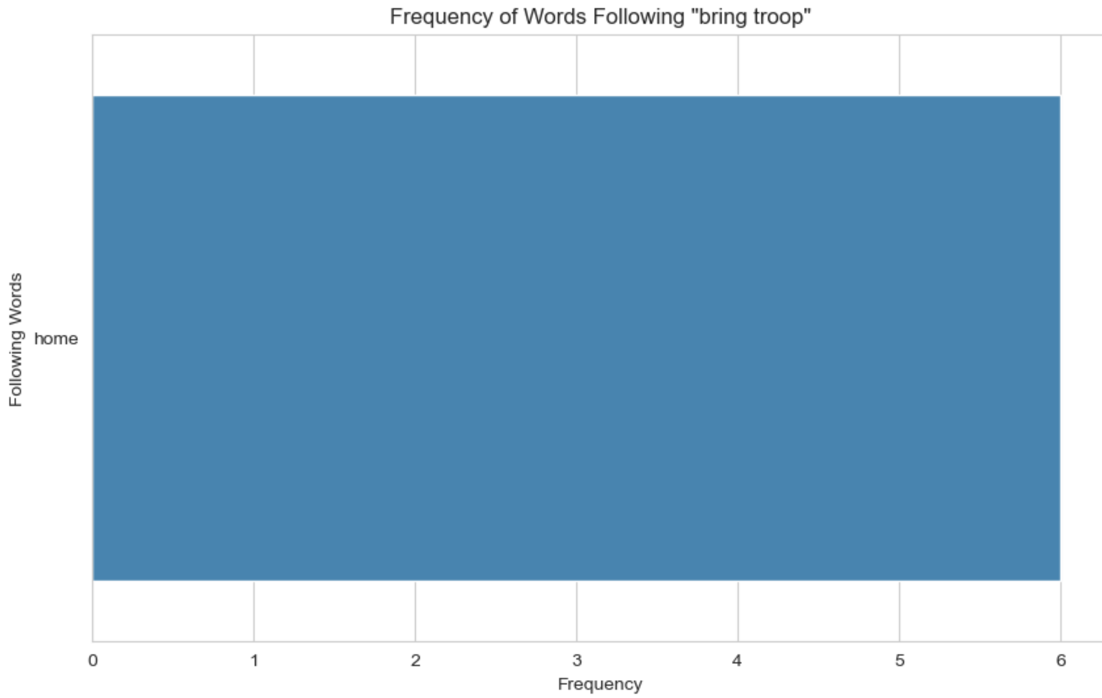
- Statistically significant trigrams

[46]:
```
t_critical_trigram = helper_functions.find_t_critical(whTrigrams.N)

whTrigrams_significant = [i for i, j in whTrigrams.
↪score_ngrams(trigram_measures.student_t) if j >= t_critical_trigram]

whTrigrams_significant
```

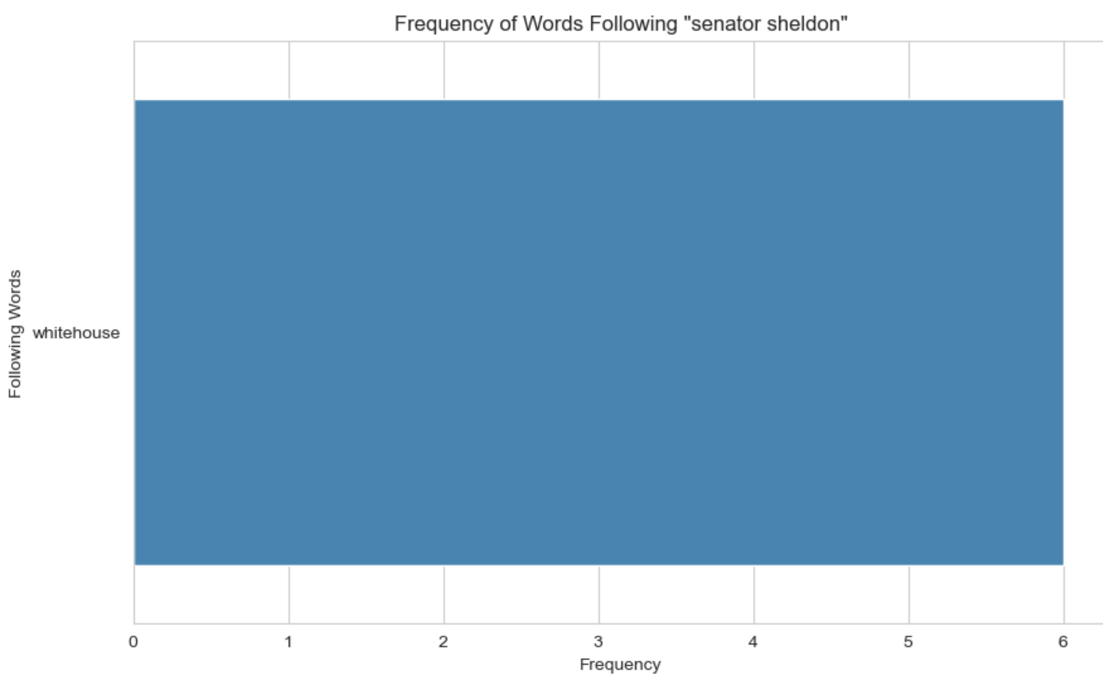
[46]:
```
[('sheldon', 'whitehouse', 'd'),
 ('whitehouse', 'd', 'r.i'),
 ('bring', 'troop', 'home'),
 ('senator', 'sheldon', 'whitehouse'),
 ('stem', 'cell', 'research'),
 ('u.s', 'senator', 'sheldon')]
```

[47]:
```
# Find out conditional frequency distribution for trigrams
trigram_list = list(nltk.trigrams(whReleases['normalized_tokens'].sum()))
trigram_cfd = nltk.ConditionalFreqDist(((w1, w2), w3) for w1, w2, w3 in␣
↪trigram_list)
```

[48]:
```
helper_functions.plot_ngram_frequencies(("bring", "troop"), trigram_cfd)
```



[49]:
```
helper_functions.plot_ngram_frequencies(("senator", "sheldon"), trigram_cfd)
```



1.7 Exercise 5

Perform NER on a (modest) subset of your corpus of interest. List all of the different kinds of entities tagged? What does their distribution suggest about the focus of your corpus? For a subset of your corpus, tally at least one type of named entity and calculate the Precision, Recall and F-score for the NER classification just performed.

First, create enetity tags for the sentence:

[50]:
```
abstract_corpus_df['classified_sents'] = abstract_corpus_df['Sentence'].
 ↪apply(lambda x: helper_functions.tag_sents_ner(x))
abstract_corpus_df['classified_sents'].head()
```

[50]:
```
0    [[(one two or three, CARDINAL), (CDA, ORG), (C…
1    [[(LTC, PERSON), (five, CARDINAL), (Cochrane C…
2    [[(Confirmation, ORG), (three, CARDINAL)], [(t…
3    [[(BED, ORG), (BED, ORG), (METHOD, ORG), (Febr…
4    [[(one, CARDINAL), (four, CARDINAL), (only one…
Name: classified_sents, dtype: object
```

Find out the different kinds of entities tagged:

[51]:
```
abstract_corpus_entity_kind_counts = {}
for entry in abstract_corpus_df['classified_sents']:
    for sentence in entry:
        for _, kind in sentence:
            abstract_corpus_entity_kind_counts[kind] =␣
 ↪abstract_corpus_entity_kind_counts.get(kind, 0) + 1

abstract_corpus_entity_kind_counts = sorted(abstract_corpus_entity_kind_counts.
 ↪items(), key = lambda x: x[1], reverse = True)

len(abstract_corpus_entity_kind_counts)
```

[51]:
```
18
```

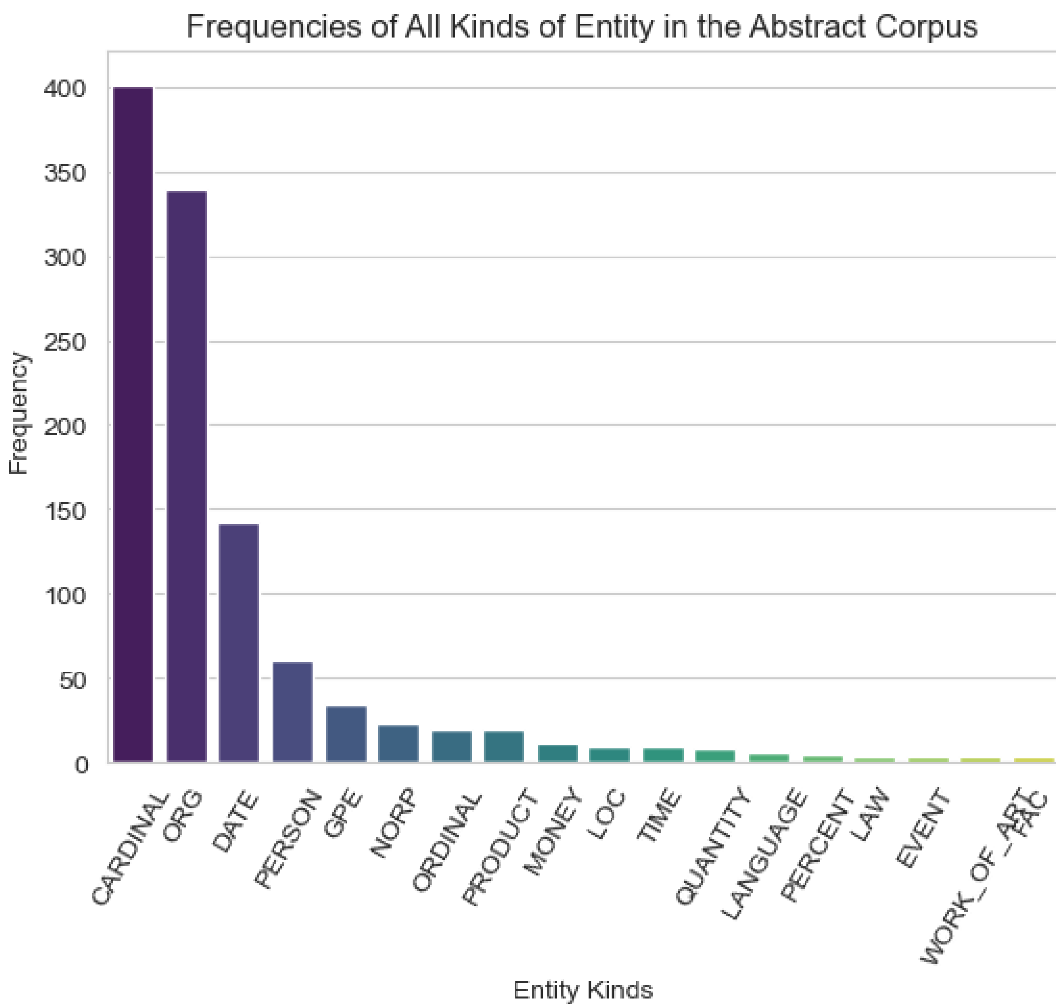
[52]:
```
abstract_corpus_entity_kind_counts
```

[52]:
```
[('CARDINAL', 401),
 ('ORG', 339),
 ('DATE', 143),
 ('PERSON', 61),
 ('GPE', 34),
 ('NORP', 23),
 ('ORDINAL', 20),
 ('PRODUCT', 19),
 ('MONEY', 12),
 ('LOC', 9),
 ('TIME', 9),
```

```
 ('QUANTITY', 8),
 ('LANGUAGE', 6),
 ('PERCENT', 5),
 ('LAW', 4),
 ('EVENT', 4),
 ('WORK_OF_ART', 4),
 ('FAC', 3)]
```

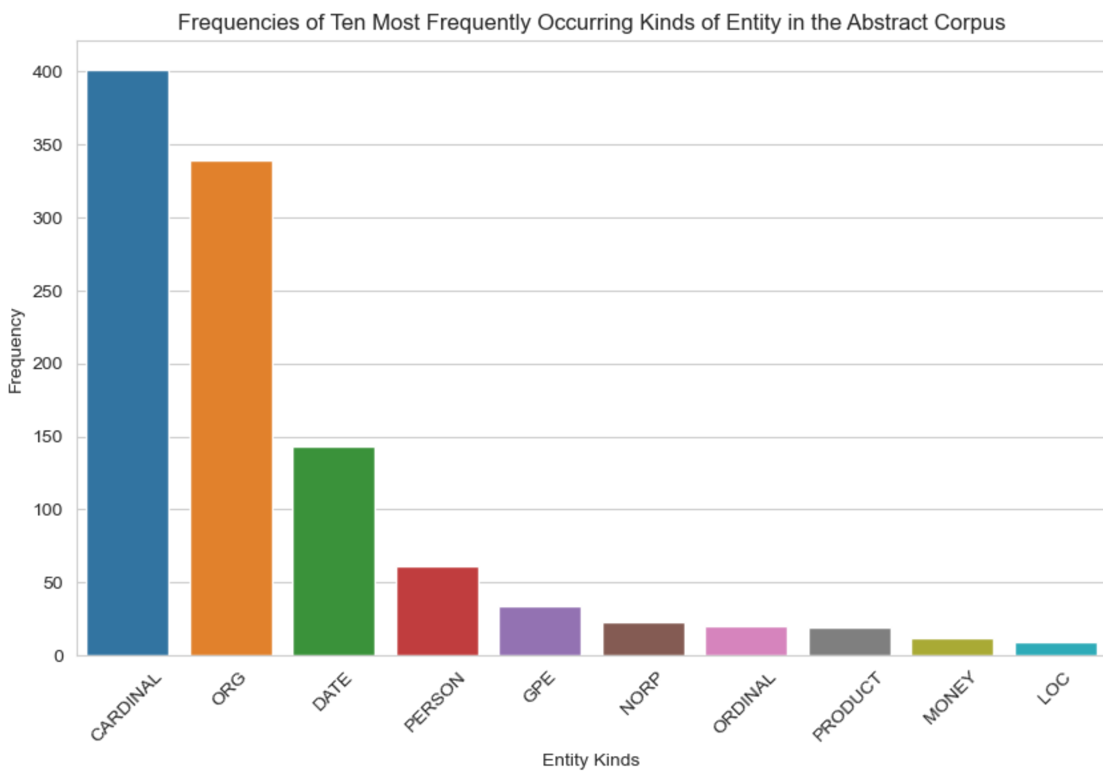
[53]:
```
sns.barplot(x='Entity Kinds', y='Frequency', data=pd.
 ↪DataFrame(abstract_corpus_entity_kind_counts, columns=['Entity Kinds',␣
 ↪'Frequency']), palette="viridis")
plt.title('Frequencies of All Kinds of Entity in the Abstract Corpus')
plt.xticks(rotation=60)
```

[53]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17]),
 [Text(0, 0, 'CARDINAL'),
  Text(1, 0, 'ORG'),
  Text(2, 0, 'DATE'),
  Text(3, 0, 'PERSON'),
  Text(4, 0, 'GPE'),
  Text(5, 0, 'NORP'),
  Text(6, 0, 'ORDINAL'),
  Text(7, 0, 'PRODUCT'),
  Text(8, 0, 'MONEY'),
  Text(9, 0, 'LOC'),
  Text(10, 0, 'TIME'),
  Text(11, 0, 'QUANTITY'),
  Text(12, 0, 'LANGUAGE'),
  Text(13, 0, 'PERCENT'),
  Text(14, 0, 'LAW'),
  Text(15, 0, 'EVENT'),
  Text(16, 0, 'WORK_OF_ART'),
  Text(17, 0, 'FAC')])
```



[54]:
```
kinds, counts = zip(*abstract_corpus_entity_kind_counts[:10])
df = pd.DataFrame({
    'Entity Kinds': kinds,
    'Frequency': counts
})

sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))
sns.barplot(x='Entity Kinds', y='Frequency', data=df)
plt.title('Frequencies of Ten Most Frequently Occurring Kinds of Entity in the␣
 ↪Abstract Corpus')
plt.xlabel('Entity Kinds')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
```

[54]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'CARDINAL'),
  Text(1, 0, 'ORG'),
  Text(2, 0, 'DATE'),
  Text(3, 0, 'PERSON'),
  Text(4, 0, 'GPE'),
  Text(5, 0, 'NORP'),
  Text(6, 0, 'ORDINAL'),
  Text(7, 0, 'PRODUCT'),
  Text(8, 0, 'MONEY'),
  Text(9, 0, 'LOC')])
```



For Exercise 5, I choose to tally "ORG" entity named and calculate the Precision, Recall and F-score for the NER classification just performed on a subset of corpus

[55]:
```
# Create a set of unique set of
org_entity_ner = set()

# Set random seed
seed = 2333

for s in abstract_corpus_df['classified_sents'].sample(50, random_state=seed).
 ↪sum():
```

```
    for entity, kind in s:
        if kind == "ORG":
            org_entity_ner.add(entity)

print("There are in total {} unique ORG labels from a sample of 50 sentences.".
 ↪format(len(org_entity_ner)))
```

```
There are in total 120 unique ORG labels from a sample of 50 sentences.
```

[56]:
```
print(org_entity_ner)
```

```
{'FDA', 'WhatsApp Groups', 'Meta Analyses', 'Childhood', 'Thematic Analysis',
'Comparing', 'CCMDCTR', 'Groups', 'Norris', 'WhatsApp', 'the Australian
Government Targeting', 'VSTM', 'Eleven', 'Electrophysiological', 'PsycInfo
Database Record', 'Recognition', 'UC', 'IVF', 'Hedges', 'AnthroSource
Reference', 'Purpose Objective Research Question Focus of Study Deploying',
'ASD', 'IBM', 'ANS', 'RESULTS Overall', 'National Health Service Direct',
'Hand', 'Publications', 'the European Convention on Human Rights', 'PubMed
CINAHL PsychINFO', 'CST Findings', 'AMSTAR', 'Preferred Reporting Items for
Systematic Reviews', "the Cochrane Common Mental Disorders Group 's", 'AIM',
'the Joanna Briggs Institute JBI Critical', 'Background Persons',
'Confidentiality', 'ACM', 'Confirmation', 'CONCLUSIONS Data', 'Conclusion
Anxiety', 'COVID-19 infection Panic', 'Limited', 'the Cochrane Central Register
of Controlled Trials', 'Telemedicine', 'APA', 'Medline Embase', 'CBT', 'CTM',
'PRISMA', 'CI', 'Google Scholar', 'METHODS Pubmed Scopus Science Direct',
'Method Medline PsycInfo SciSearch SocScisearch', 'WhatsApp Telemedicine',
'COVID-19', 'HCI', 'RA', 'Medline', 'METHOD Systematic', 'the Preferred
Reporting Items for Systematic Reviews', 'Review', 'Search', 'MedLine Psycinfo
CINAHL', 'Cochrane', 'PubMed', 'Social Abstracts', 'RESULTS Adverse', 'the
European Court of Human Rights ECtHR', 'Data Collection and Analysis', 'METHOD',
'PPC', 'NNTB 5 Comparing', 'schizophrenia Discussion Persons', 'linear',
'Significance Theories', 'BACKGROUND Paediatric', 'OUD', 'RESULTS Nineteen',
'Science and Cochrane', 'Continuing', 'The Cochrane Risk of Bias', 'Weight',
'Continued', 'Systematic Review Registration', 'Wandering', 'MedLine Scopus
Cochrane', 'the Centre for Reviews and Dissemination', 'Results Individual',
'Youngsters', 'PILOTS', 'Reciprocal', 'Results A', 'BACKGROUND Depressive',
'Background Anxiety', 'Specifically', 'CONCLUSION', 'Rheumatoid', 'ICSI',
'Results Bipolar', 'S. Inamdar', '9/47 Conclusions Recommendations',
'OBJECTIVES', 'Recommendations', 'PubMed Medline', 'Oliva and Torralba', 'VLTM',
'The Cochrane Library Reference', 'ASL', "the National Library of Medicine 's
PubMed Database", 'BNI', 'PE', 'Future', 'Participants N', 'CST', 'Baseline',
'EIBI', 'MEDLINE', 'Conclusions The'}
```

Manually code whether the tags are indeed ORG

[57]:
```
org_entity_manual = {
    "WhatsApp", "The Cochrane Library Reference", "the Cochrane Common Mental␣
 ↪Disorders Group",
```

```
    "IBM", "MEDLINE", "APA", "AMSTAR", "the Joanna Briggs Institute JBI␣
↪Critical",
    "METHODS Pubmed Scopus Science Direct", "Preferred Reporting Items for␣
↪Systematic Reviews",
    "Method Medline PsycInfo SciSearch SocScisearch", "BNI", "CCMDCTR", "FDA",␣
↪"the European Court of Human Rights ECtHR",
    "Google Scholar", "National Health Service Direct", "PsycInfo Database␣
↪Record", "The Cochrane Risk of Bias",
    "AnthroSource Reference", "the Cochrane Central Register of Controlled␣
↪Trials", "ACM",
    "MedLine Scopus Cochrane", "the European Convention on Human Rights",␣
↪"Cochrane", "PubMed Medline",
    "PubMed CINAHL PsychINFO", "PRISMA", "MedLine Psycinfo CINAHL", "PubMed",␣
↪"the Centre for Reviews and Dissemination",
    "the National Library of Medicine 's PubMed Database"
}
```

Calculate the Precision, Recall and F-score

[58]:
```
# Calculate true_positive, false_positive, and false_positive
true_positive = len(org_entity_ner & org_entity_manual)
false_positive = len(org_entity_ner - org_entity_manual)
false_negative = len(org_entity_manual - org_entity_ner)

precision = true_positive / (true_positive + false_positive) if true_positive +␣
↪false_positive > 0 else 0
recall = true_positive / (true_positive + false_negative) if true_positive +␣
↪false_negative > 0 else 0
f_score = 2 * precision * recall / (precision + recall) if precision + recall >␣
↪0 else 0

org_classification_performance = pd.DataFrame({"ORG Precision": precision,
                                               "ORG Recall": recall,
                                               "ORG F-score":␣
↪f_score},index=["ORG Classification Performance"])
org_classification_performance
```

[58]:

	ORG Precision	ORG Recall	ORG F-score
ORG Classification Performance	0.258333	0.96875	0.407895

1.8 Exercise 6

Parse a (modest) subset of your corpus of interest. How deep are the phrase structure and dependency parse trees nested? How does parse depth relate to perceived sentence complexity? What are five things you can extract from these parses for subsequent analysis? (e.g., nouns collocated in a noun phrase; adjectives that modify a noun; etc.) Capture these sets of things for a focal set of words (e.g., “Bush”, “Obama”, “Trump”). What do they reveal about the roles that these entities are perceive to play in the social world inscribed by your texts?

First, parse the corpus:

[59]:
```
abstract_corpus_df.head()
```

[59]:
```
                                               Title  \
0  The Control of Single-color and Multiple-color…
1  Non-pharmacologic and pharmacologic treatments…
2  Confirmation bias in information search, inter…
3  Meta-analysis on the long-term effectiveness o…
4  On the search for a selective and retroactive …

                                              Author  \
0                A. Grubert, N. Carlisle, M. Eimer
1  K. Atchison, J. Watt, Delaney Ewert, A. Toohey…
2                          Dáša Vedejová, V. Čavojová
3  A. Hilbert, D. Petroff, S. Herpertz, R. Pietro…
4                                F. Kalbe, L. Schwabe

                                            Abstract  \
0  The question whether target selection in visua…
1  BACKGROUND\nolder adults living in long-term c…
2  Abstract Confirmation bias is often used as an…
3  OBJECTIVE\nLong-term effectiveness is a critic…
4  Storing motivationally salient experiences pre…

                                    Cleaned_Abstract  \
0  The question whether target selection in visua…
1  BACKGROUND\nolder adults living in long-term c…
2  Abstract Confirmation bias is often used as an…
3  OBJECTIVE\nLong-term effectiveness is a critic…
4  Storing motivationally salient experiences pre…

                                      Abstract_Token  \
0  [The, question, whether, target, selection, in…
1  [BACKGROUND, older, adults, living, in, long, …
2  [Abstract, Confirmation, bias, is, often, used…
3  [OBJECTIVE, Long, term, effectiveness, is, a, …
4  [Storing, motivationally, salient, experiences…

                                            Sentence  \
0  [[The, question, whether, target, selection, i…
1  [[BACKGROUND, older, adults, living, in, long,…
2  [[Abstract, Confirmation, bias, is, often, use…
3  [[OBJECTIVE, Long, term, effectiveness, is, a,…
4  [[Storing, motivationally, salient, experience…
```

```
                                        POS_Sentence  \
0  [[(The, DT), (question, NN), (whether, IN), (t…
1  [[(BACKGROUND, NN), (older, JJR), (adults, NNS…
2  [[(Abstract, NNP), (Confirmation, NNP), (bias,…
3  [[(OBJECTIVE, NNP), (Long, JJ), (term, NN), (e…
4  [[(Storing, VBG), (motivationally, RB), (salie…

                                     classified_sents
0  [[(one two or three, CARDINAL), (CDA, ORG), (C…
1  [[(LTC, PERSON), (five, CARDINAL), (Cochrane C…
2  [[(Confirmation, ORG), (three, CARDINAL)], [(t…
3  [[(BED, ORG), (BED, ORG), (METHOD, ORG), (Febr…
4  [[(one, CARDINAL), (four, CARDINAL), (only one…
```

[60]:
```python
sentence_list = [" ".join(i) for i in  abstract_corpus_df["Sentence"].sum()]

parsed_corpus = [nlp(sentence) for sentence in sentence_list]

parsed_corpus[:10]
```

[60]:
```
[The question whether target selection in visual search can be effectively
controlled by simultaneous attentional templates for multiple features is still
under dispute,
 We investigated whether multiple color attentional guidance is possible when
target colors remain constant and can thus be represented in long term memory
but not when they change frequently and have to be held in working memory,
 Participants searched for one two or three possible target colors that were
specified by cue displays at the start of each trial,
 In constant color blocks the same colors remained task relevant throughout,
 In variable color blocks target colors changed between trials,
 The contralateral delay activity CDA to cue displays increased in amplitude as
a function of color memory load in variable color blocks which indicates that
cued target colors were held in working memory,
 In constant color blocks the CDA was much smaller suggesting that color
representations were primarily stored in long term memory,
 N2pc components to targets were measured as a marker of attentional target
selection,
 Target N2pcs were attenuated and delayed during multiple color search
demonstrating less efficient attentional deployment to color defined target
objects relative to single color search,
 Importantly these costs were the same in constant color and variable color
blocks]
```

- Deph of the phrase structure and dependency parse trees nested:

[61]:
```python
sentence_depth_df = {"Sentence": [], "Depth": []}

for sentence in parsed_corpus:
```

```
    if len(sentence) > 0:
        root = [token for token in sentence if token.head == token][0]
        sentence_depth_df["Sentence"].append(sentence.text)
        sentence_depth_df["Depth"].append(helper_functions.max_depth(root))

sentence_depth_df = pd.DataFrame(sentence_depth_df)

sentence_depth_df["Depth"].describe()
```

[61]:
```
count    959.000000
mean       7.312826
std        3.042523
min        1.000000
25%        5.000000
50%        7.000000
75%        9.000000
max       30.000000
Name: Depth, dtype: float64
```

- Relation between parse depth and perceived sentence complexity: In a nutshell, the deeper the more complex of a sentence

[62]:
```
min_depth = sentence_depth_df["Depth"].min()
mean_depth = round(sentence_depth_df["Depth"].mean())
max_depth = sentence_depth_df["Depth"].max()
```

[63]:
```
# Randomly select one sentence with minimum depth
sample_sentence_min_depth = nlp(sentence_depth_df[sentence_depth_df["Depth"] ==␣
 ↪min_depth]["Sentence"].sample(1, random_state=seed).to_list()[0])
spacy.displacy.render(sample_sentence_min_depth, style='dep')
```

```
<IPython.core.display.HTML object>
```

[64]:
```
# Randomly select one sentence with rounded value of mean depth
sample_sentence_mean_depth = nlp(sentence_depth_df[sentence_depth_df["Depth"]␣
 ↪== mean_depth]["Sentence"].sample(1, random_state=seed).to_list()[0])
spacy.displacy.render(sample_sentence_mean_depth, style='dep')
```

```
<IPython.core.display.HTML object>
```

[65]:
```
sample_sentence_max_depth =␣
 ↪nlp(str(sentence_depth_df[sentence_depth_df["Depth"] ==␣
 ↪max_depth]["Sentence"].sample(1, random_state=seed).to_list()[0]))
spacy.displacy.render(sample_sentence_max_depth, style='dep')
```

```
<IPython.core.display.HTML object>
```

- Extract five linguisitic features from parses and apply them to a focal set of words

[66]:
```
focal_words = {"memory", "disorder"}
```

[67]:
```
linguistic_features = {word: {'noun_phrases': [], 'adjectives': [], 'verbs':␣
↪[], 'dependencies': [], 'co_occurring': []} for word in focal_words}

for doc in parsed_corpus:
    # Feature 1: noun phrases containing the focal words
    for np in doc.noun_chunks:
        np_words = set(np.text.lower().split())
        common_words = focal_words.intersection(np_words)
        for word in common_words:
            linguistic_features[word]['noun_phrases'].append(np.text)

    for token in doc:
        # Feature 2: adjectives modifying the focal words
        if token.pos_ == "ADJ" and token.head.text.lower() in focal_words:
            linguistic_features[token.head.text.lower()]['adjectives'].
↪append(token.text)

        # Feature 3: verbs associated with the focal Words (either subjects or␣
↪objects)
        if token.head.text.lower() in focal_words and token.pos_ == "VERB":
            linguistic_features[token.head.text.lower()]['verbs'].append(token.
↪text)
        if token.text.lower() in focal_words and token.head.pos_ == "VERB":
            linguistic_features[token.text.lower()]['verbs'].append(token.head.
↪text)

        # Feature 4: dependency relations where the focal words are involved
        if token.text.lower() in focal_words:
            linguistic_features[token.text.lower()]['dependencies'].
↪append((token.text, token.dep_, token.head.text))

    # Feature 5: named entities that frequently co-occur with the focal words␣
↪in the same sentences
    for ent in doc.ents:
        sentence_words = set(token.text.lower() for token in ent.sent)
        if focal_words.intersection(sentence_words):
            for focal_word in focal_words.intersection(sentence_words):
                linguistic_features[focal_word]['co_occurring'].append(ent.text)

linguistic_features_df = pd.DataFrame(linguistic_features)
```

[68]:
```
linguistic_features_df
```

[68]:
```
                                                          memory  \
noun_phrases  [long term memory, working memory, color memor…
adjectives    [enhanced, high, weak, visual, visual, visual,…
verbs         [working, working, working, showed, working, e…
```

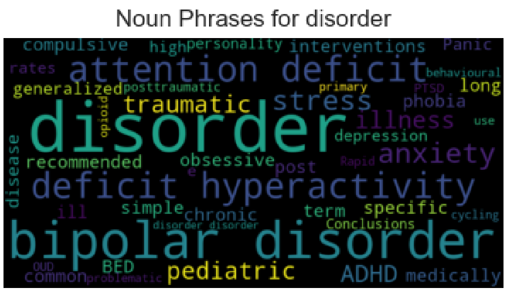
```
dependencies  [(memory, pobj, in), (memory, pobj, in), (memo…
co_occurring  [CDA, CDA, three, four, only one, four, Bayesi…

                                                        disorder
noun_phrases  [disorder BED, disorder, simple phobia obsessi…
adjectives    [obsessive, compulsive, traumatic, specific, c…
verbs         [eating, excluding, impairs, generalized, obse…
dependencies  [(disorder, compound, BED), (disorder, dobj, e…
co_occurring  [BED, RESULTS Effectiveness, up to 12 months, …
```

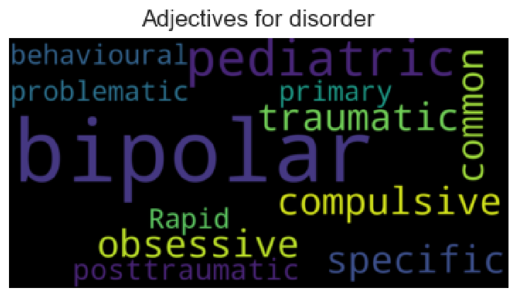
Visualization of the results

[69]:
```python
from wordcloud import WordCloud

for word, feats in linguistic_features.items():
    noun_phrases = ' '.join(feats['noun_phrases'])
    adjectives = ' '.join(feats['adjectives'])

    # Create word clouds
    np_cloud = WordCloud(width=800, height=400).generate(noun_phrases)
    adj_cloud = WordCloud(width=800, height=400).generate(adjectives)

    # Display word clouds
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(np_cloud, interpolation='bilinear')
    plt.title(f"Noun Phrases for {word}")
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(adj_cloud, interpolation='bilinear')
    plt.title(f"Adjectives for {word}")
    plt.axis('off')
```

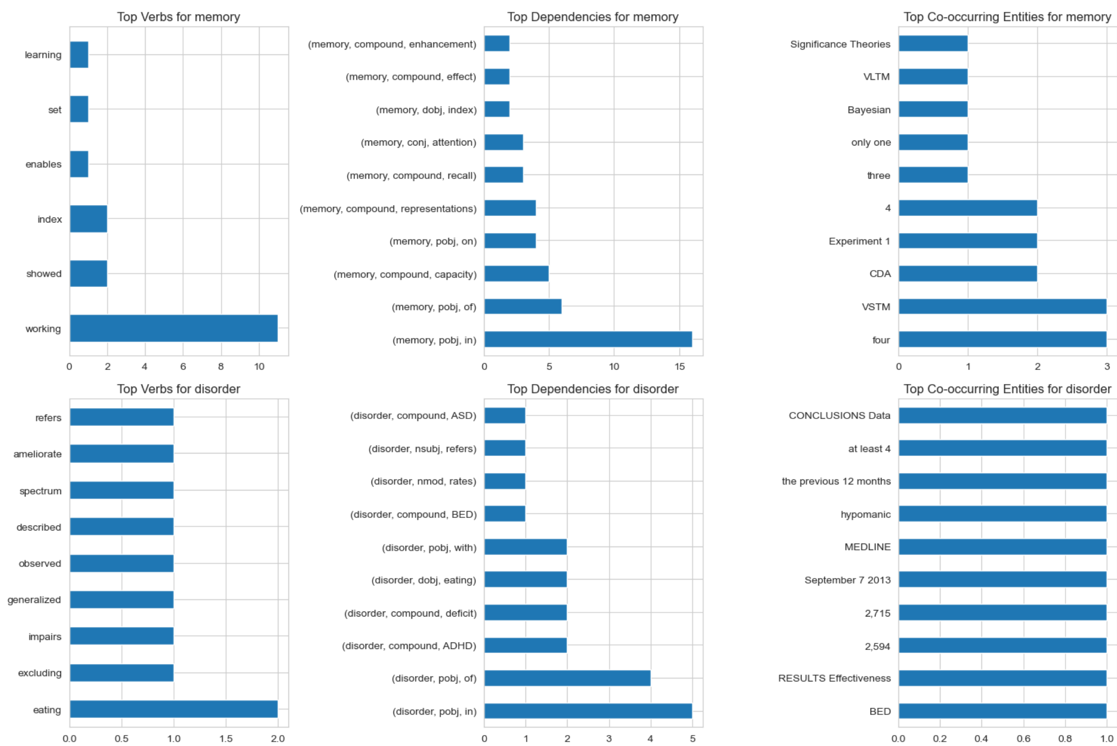
Noun Phrases for memory

Adjectives for memory





[70]:
```python
fig, axs = plt.subplots(len(focal_words), 3, figsize=(15, 5 * len(focal_words)))

for i, word in enumerate(focal_words):
    helper_functions.plot_bar_chart(linguistic_features[word]['verbs'], f"Top Verbs for {word}", axs[i, 0])
    helper_functions.plot_bar_chart(linguistic_features[word]['dependencies'], f"Top Dependencies for {word}", axs[i, 1])
    helper_functions.plot_bar_chart(linguistic_features[word]['co_occurring'], f"Top Co-occurring Entities for {word}", axs[i, 2])

plt.tight_layout()
```



1.9 Exercise 7

Calculate the KL and ^{2} divergences, and the KS and Wasserstein distances between four separate corpora, plot these with heatmaps, and then array them in two dimensions with multidimensional scaling as above. What does this reveal about relations between the corpora? Which analysis (and divergence or distribution) distinguishes the authors or documents better?

For Exercise 7, I scraped abstracts from four fields of study to create four separate corpora.

[71]:
```
fieldsOfStudy = ["Engineering", "Physics", "Philosophy", "History"]

abstract_corpus_collection = {f: helper_functions.scrape_abstract(f) for f in
  ↪fieldsOfStudy}
```

[72]:
```
abstract_corpus_collection["Engineering"].head()
```

[72]:
```
  FieldOfStudy                                              Title  \
0  Engineering      Engineering Psychology and Human Performance
1  Engineering  Active learning increases student performance …
2  Engineering  Multiplex Genome Engineering Using CRISPR/Cas …
3  Engineering  The new frontier of genome engineering with CR…
4  Engineering      RNA-Guided Human Genome Engineering via Cas9

                                              Author  \
0  C. Wickens, J. G. Hollands, S. Banbury, R. Par…
1  S. Freeman, Sarah L. Eddy, Miles McDonough, Mi…
2  Le Cong, F. Ran, David B. T. Cox, Shuailiang L…
3                        J. Doudna, E. Charpentier
4  P. Mali, Luhan Yang, K. Esvelt, J. Aach, M. Gu…

                                            Abstract  \
0  1. Introduction to Engineering Psychology and …
1  Significance The President's Council of Adviso…
2  Genome Editing Clustered regularly interspaced…
3  Background Technologies for making and manipul…
4  Genome Editing Clustered regularly interspaced…

                                      tokenized_text  \
0  [1, Introduction, to, Engineering, Psychology,…
1  [Significance, The, President, 's, Council, of…
2  [Genome, Editing, Clustered, regularly, inters…
3  [Background, Technologies, for, making, and, m…
4  [Genome, Editing, Clustered, regularly, inters…

                                   normalized_tokens     stopwords  \
0  [introduction, engineering, psychology, human,…        [make]
1  [significance, president, council, advisor, sc…  [call, call]
2  [genome, editing, cluster, regularly, interspa…        [show]
```

```
3  [background, technology, make, manipulate, dna…  [make, have]
4  [genome, editing, cluster, regularly, interspa…            []

                                       non_stopwords
0  [introduction, engineering, psychology, human,…
1  [significance, president, council, advisor, sc…
2  [genome, editing, cluster, regularly, interspa…
3  [background, technology, manipulate, dna, enab…
4  [genome, editing, cluster, regularly, interspa…
```

[73]:
```
abstract_corpus_collection["Physics"].head()
```

[73]:
```
  FieldOfStudy                                              Title  \
0      Physics             Plasma Physics via Computer Simulation
1      Physics              CRC Handbook of Chemistry and Physics
2      Physics                Introduction to solid state physics
3      Physics                                     Plasma Physics
4      Physics  Atmospheric Chemistry and Physics: From Air Po…

                               Author  \
0          C. Birdsall, A. Langdon
1                      W. M. Haynes
2                         C. Kittel
3               Richard Fitzpatrick
4  J. Seinfeld, S. Pandis, K. Noone

                                            Abstract  \
0  PART 1: PRIMER Why attempting to do plasma phy…
1  CRC handbook of chemistry and physics , CRC ha…
2  Mathematical Introduction Acoustic Phonons Pla…
3  Several approaches are commonly used to study …
4  Expanded and updated with new findings and new…

                                      tokenized_text  \
0  [PART, 1, PRIMER, Why, attempting, to, do, pla…
1  [CRC, handbook, of, chemistry, and, physics, C…
2  [Mathematical, Introduction, Acoustic, Phonons…
3  [Several, approaches, are, commonly, used, to,…
4  [Expanded, and, updated, with, new, findings, …

                                   normalized_tokens     stopwords  \
0  [primer, attempt, plasma, physics, computer, s…        [make]
1  [crc, handbook, chemistry, physics, crc, handb…            []
2  [mathematical, introduction, acoustic, phonon,…            []
3  [approach, commonly, study, system, o, f, char…  [call, give]
4  [expand, update, new, finding, new, feature, n…            []
```

```
                                       non_stopwords
0  [primer, attempt, plasma, physics, computer, s…
1  [crc, handbook, chemistry, physics, crc, handb…
2  [mathematical, introduction, acoustic, phonon,…
3  [approach, commonly, study, system, o, f, char…
4  [expand, update, new, finding, new, feature, n…
```

[74]:
```
abstract_corpus_collection["Philosophy"].head()
```

[74]:
```
  FieldOfStudy                                              Title  \
0   Philosophy     Space-Perception and the Philosophy of Science
1   Philosophy             Towards a Transformation of Philosophy
2   Philosophy  Speech Acts: An Essay in the Philosophy of Lan…
3   Philosophy  Philosophy in the flesh : the embodied mind an…
4   Philosophy                      The Philosophy of Philosophy

                      Author  \
0     P. Heelan, James L. Park
1  K. Apel, G. Adey, D. Frisby
2                    J. Searle
3   G. Lakoff, Mark L. Johnson
4                T. Williamson

                                            Abstract  \
0  Drawing on the phenomenological tradition in t…
1  As Apel himself notes in his preface, the expr…
2  Part I. A Theory of Speech Acts: 1. Methods an…
3  * Introduction: Who Are We? How The Embodied M…
4  Preface. Acknowledgments. Introduction. 1. The…

                                      tokenized_text  \
0  [Drawing, on, the, phenomenological, tradition…
1  [As, Apel, himself, notes, in, his, preface, t…
2  [Part, I., A, Theory, of, Speech, Acts, 1, Met…
3  [Introduction, Who, Are, We, How, The, Embodie…
4  [Preface, Acknowledgments, Introduction, 1, Th…

                                   normalized_tokens      stopwords  \
0  [draw, phenomenological, tradition, philosophy…             []
1  [apel, note, preface, expression, transformati…  [name, take]
2  [i., theory, speech, act, method, scope, expre…             []
3  [introduction, embody, mind, challenge, wester…             []
4  [preface, acknowledgment, introduction, lingui…  [take, well]

                                       non_stopwords
0  [draw, phenomenological, tradition, philosophy…
1  [apel, note, preface, expression, transformati…
```

```
2  [i., theory, speech, act, method, scope, expre…
3  [introduction, embody, mind, challenge, wester…
4  [preface, acknowledgment, introduction, lingui…
```

[75]:
```python
abstract_corpus_collection["History"].head()
```

[75]:
```
  FieldOfStudy                                              Title  \
0      History  Capital in the twenty-first century: a multidi…
1      History                           A short history of SHELX.
2      History        The MovieLens Datasets: History and Context
3      History                         History on Film/Film on History
4      History                        A Brief History of Neoliberalism

                               Author  \
0                         T. Piketty
1                       G. Sheldrick
2  F. M. Harper, J. Konstan, J. A.
3              Robert A. Rosenstone
4                          D. Harvey

                                            Abstract  \
0  I am most grateful to the editors of the Briti…
1  An account is given of the development of the …
2  The MovieLens datasets are widely used in educ…
3  Chapter 1: History on film. Chapter 2: To see …
4  Neoliberalism - the doctrine that market excha…

                                      tokenized_text  \
0  [I, am, most, grateful, to, the, editors, of, …
1  [An, account, is, given, of, the, development,…
2  [The, MovieLens, datasets, are, widely, used, …
3  [Chapter, 1, History, on, film, Chapter, 2, To…
4  [Neoliberalism, the, doctrine, that, market, e…

                                   normalized_tokens    stopwords  \
0  [grateful, editor, british, journal, sociology…  [put, well]
1  [account, give, development, shelx, system, co…      [give]
2  [movielen, dataset, widely, education, researc…   [hundred]
3  [chapter, history, film, chapter, past, chapte…          []
4  [neoliberalism, doctrine, market, exchange, et…      [show]

                                       non_stopwords
0  [grateful, editor, british, journal, sociology…
1  [account, development, shelx, system, computer…
2  [movielen, dataset, widely, education, researc…
3  [chapter, history, film, chapter, past, chapte…
4  [neoliberalism, doctrine, market, exchange, et…
```

[76]:
```python
# Concatenate the lists (of normalized tokens, stopwords and nonstop words)
# into one big list for each field of study and then combine each field of study
abstract_corpus_concatenated = pd.DataFrame()

# Define the indices for the type of corpora tokens in the dataframe
df_index = ['normalized_tokens', 'stopwords', 'non_stopwords']

for f in fieldsOfStudy:
    f_df = pd.DataFrame(abstract_corpus_collection[f][df_index].sum()).T
    f_df.insert(0, 'Fields_of_study', f)
    abstract_corpus_concatenated = pd.concat([abstract_corpus_concatenated,␣
↪f_df])

abstract_corpus_concatenated.reset_index(drop=True, inplace=True)
```

[77]:
```python
abstract_corpus_concatenated
```

[77]:
```
  Fields_of_study                                  normalized_tokens  \
0     Engineering  [introduction, engineering, psychology, human,…
1         Physics  [primer, attempt, plasma, physics, computer, s…
2      Philosophy  [draw, phenomenological, tradition, philosophy…
3         History  [grateful, editor, british, journal, sociology…

                                        stopwords  \
0  [make, call, call, show, make, have, make, mak…
1  [make, call, give, one, part, see, one, take, …
2  [name, take, take, well, call, keep, name, go,…
3  [put, well, give, hundred, show, name, being, …

                                    non_stopwords
0  [introduction, engineering, psychology, human,…
1  [primer, attempt, plasma, physics, computer, s…
2  [draw, phenomenological, tradition, philosophy…
3  [grateful, editor, british, journal, sociology…
```

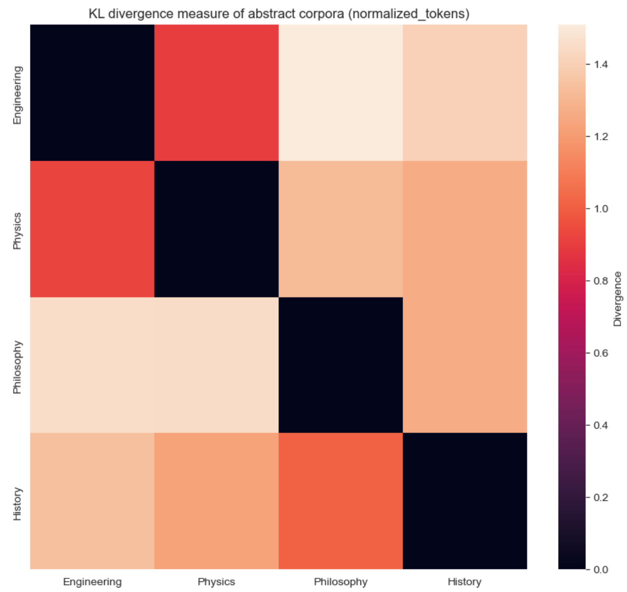
Then, I calculated the KL and ² divergences, and the KS and Wasserstein distances between four separate corpora and plot these with heatmaps

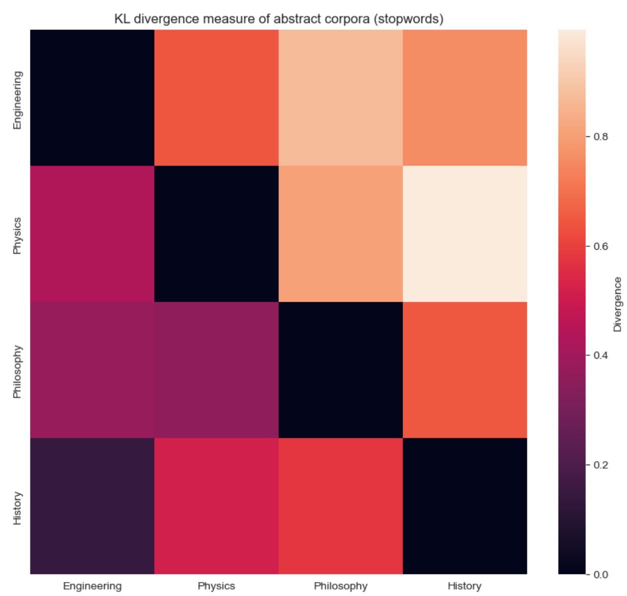
- KL divergence and multidimensional scaling:

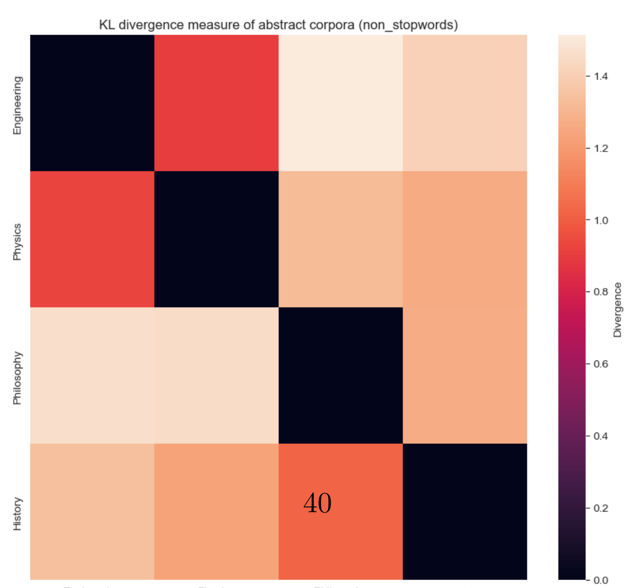
[78]:
```python
helper_functions.distributional_distance(abstract_corpus_concatenated, "KL",␣
↪fieldsOfStudy, df_index)
```

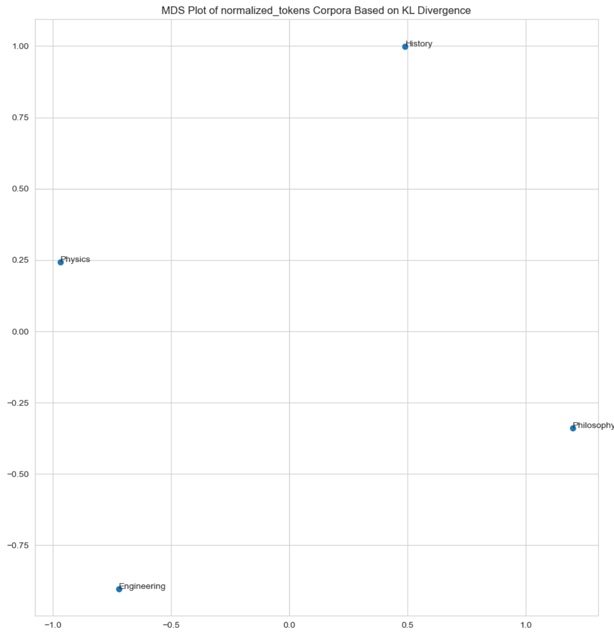
```
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
```

```
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:380: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig_heatmap.show()
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:381: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig_mds.show()
```

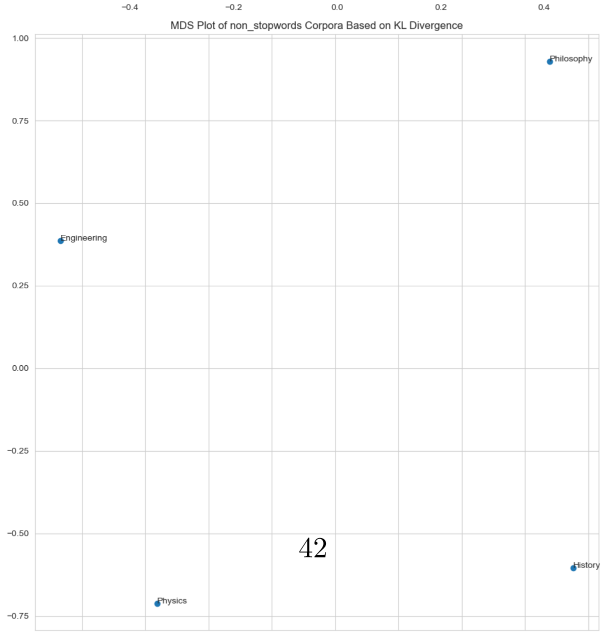

KL divergence measure of abstract corpora (normalized_tokens)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


KL divergence measure of abstract corpora (stopwords)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8


KL divergence measure of abstract corpora (non_stopwords)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


MDS Plot of normalized_tokens Corpora Based on KL Divergence
History
Physics
Philosophy
Engineering
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0


MDS Plot of stopwords Corpora Based on KL Divergence
History
Philosophy
Engineering
Physics
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4

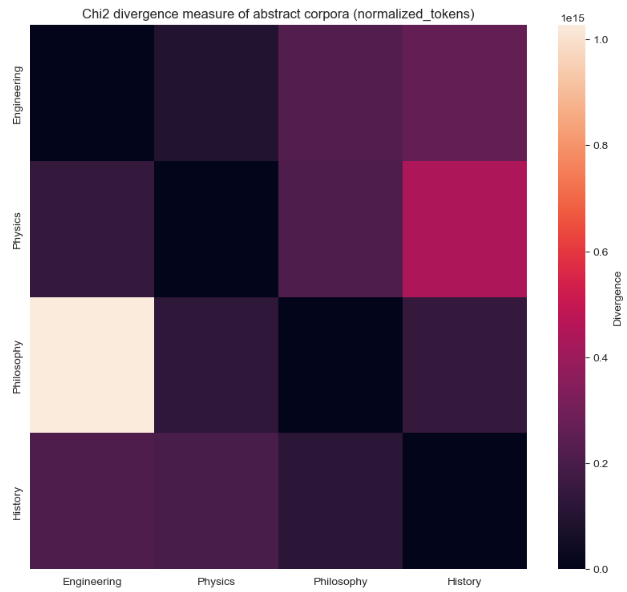

MDS Plot of non_stopwords Corpora Based on KL Divergence
Philosophy
Engineering
History
Physics
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

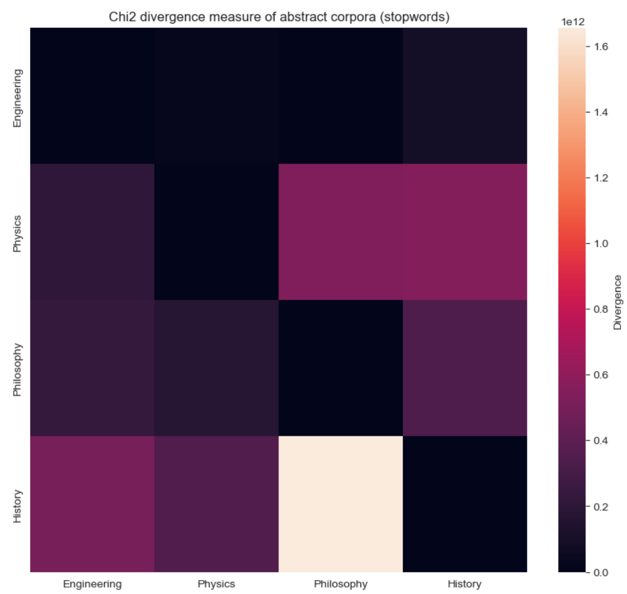
- ^{2} divergences and multidimensional scaling:

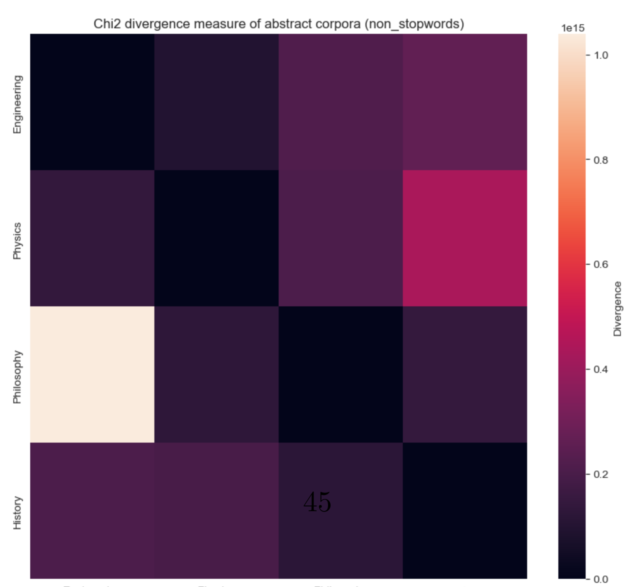
[79]:
```
helper_functions.distributional_distance(abstract_corpus_concatenated, "Chi2",␣
↪fieldsOfStudy, df_index)
```

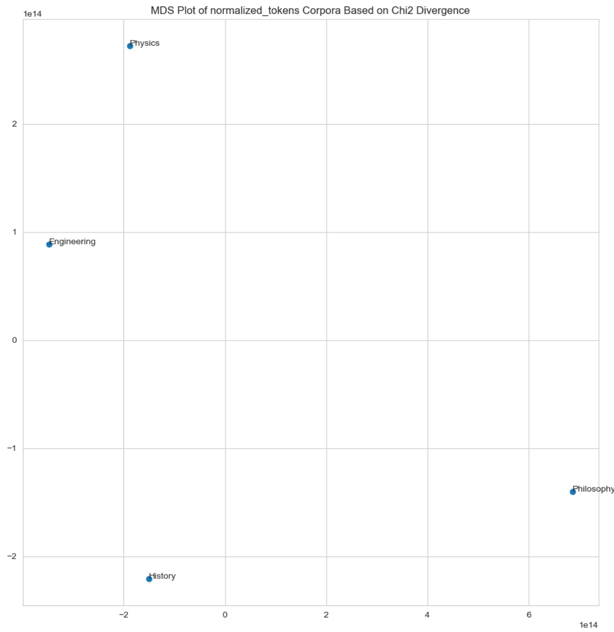
```
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:380: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig_heatmap.show()
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:381: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```

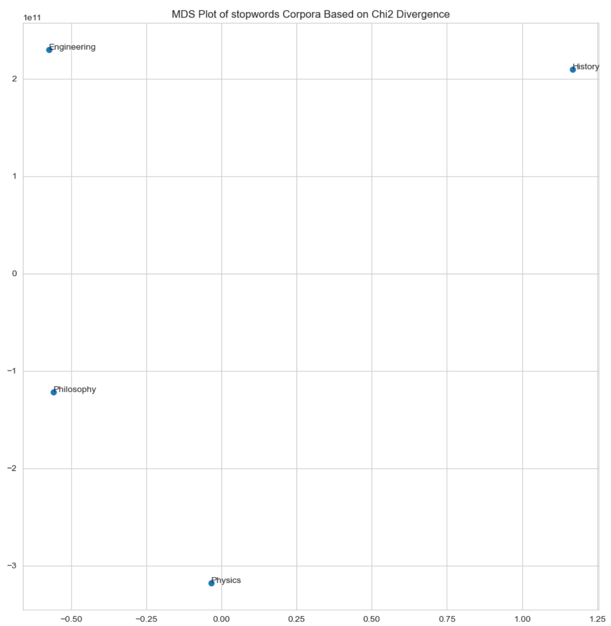
```
fig_mds.show()
```


Chi2 divergence measure of abstract corpora (normalized_tokens)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
1e15
1.0
0.8
0.6
0.4
0.2
0.0
Divergence


Chi2 divergence measure of abstract corpora (stopwords)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
1e12
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Divergence


Chi2 divergence measure of abstract corpora (non_stopwords)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
1e15
1.0
0.8
0.6
0.4
0.2
0.0
Divergence


MDS Plot of normalized_tokens Corpora Based on Chi2 Divergence
1e14
Physics
Engineering
Philosophy
History
2
1
0
−1
−2
−2
0
2
4
6
1e14


MDS Plot of stopwords Corpora Based on Chi2 Divergence
1e11
Engineering
History
Philosophy
Physics
2
1
0
−1
−2
−3
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1e12

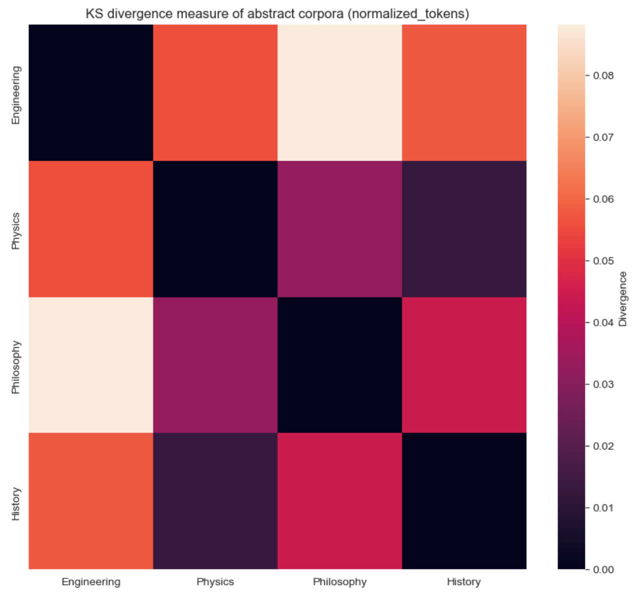

MDS Plot of non_stopwords Corpora Based on Chi2 Divergence
1e14
Philosophy
History
Engineering
Physics
4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
1e14

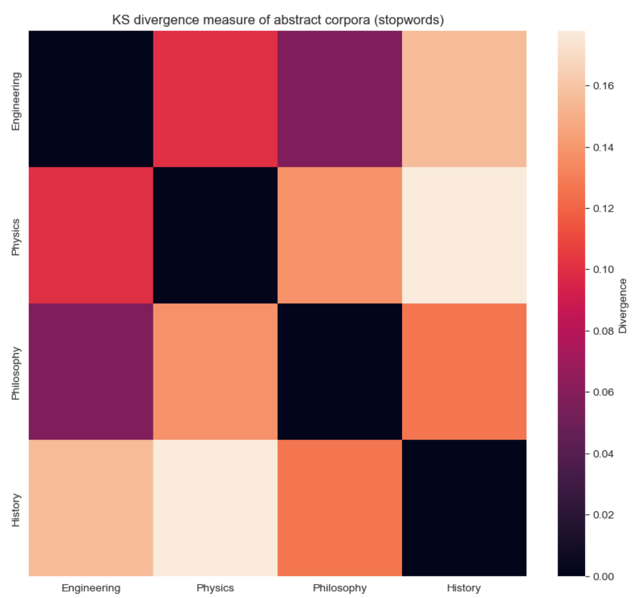
- KS distances multidimensional scaling:

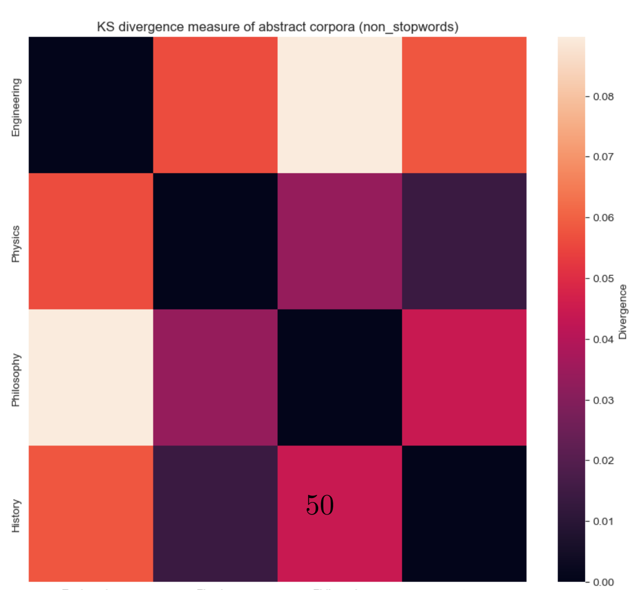
[80]:
```
helper_functions.distributional_distance(abstract_corpus_concatenated, "KS",␣
↪fieldsOfStudy, df_index)
```

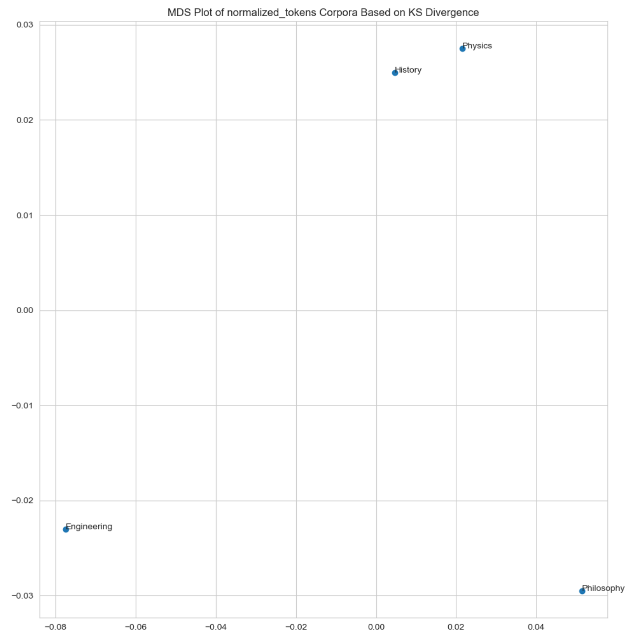
```
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:380: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig_heatmap.show()
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:381: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```

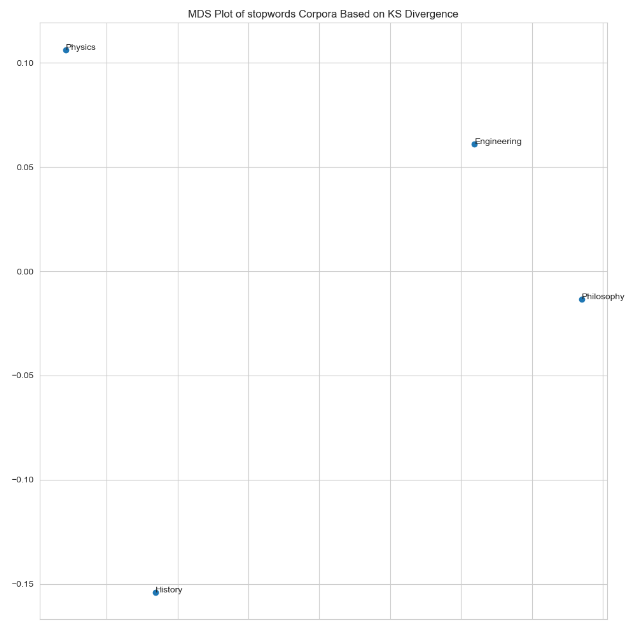
```
fig_mds.show()
```

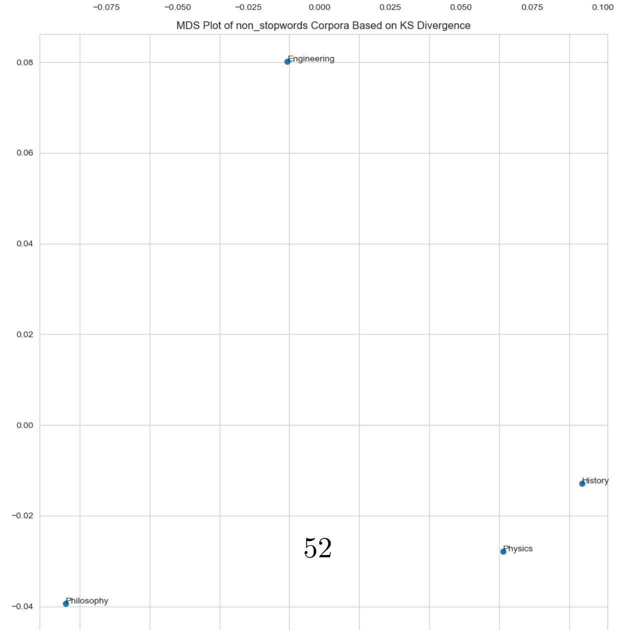

KS divergence measure of abstract corpora (normalized_tokens)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
Divergence
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


KS divergence measure of abstract corpora (stopwords)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
Divergence
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16


KS divergence measure of abstract corpora (non_stopwords)
Engineering
Physics
Philosophy
History
Engineering
Physics
Philosophy
History
Divergence
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


MDS Plot of normalized_tokens Corpora Based on KS Divergence
Physics
History
Engineering
Philosophy


MDS Plot of stopwords Corpora Based on KS Divergence
Physics
Engineering
Philosophy
History

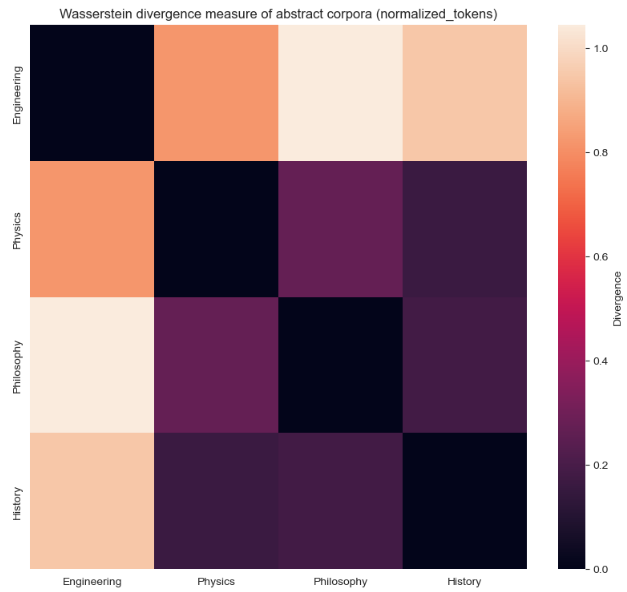

MDS Plot of non_stopwords Corpora Based on KS Divergence
Engineering
History
Physics
Philosophy

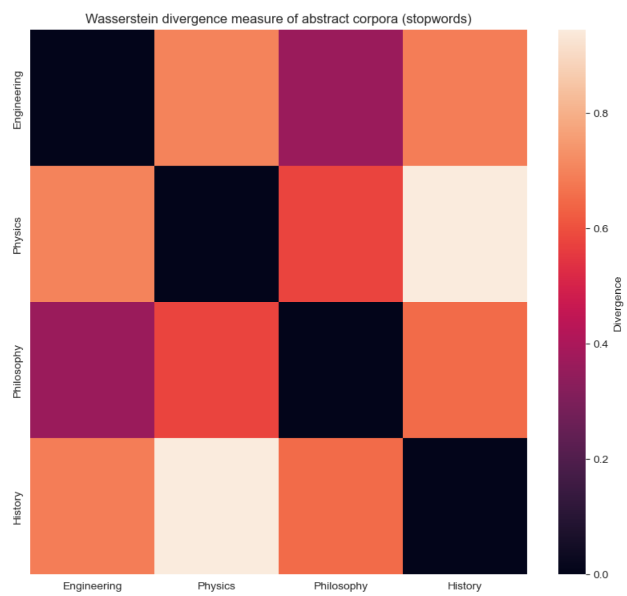
- Wasserstein distances and multidimensional scaling:

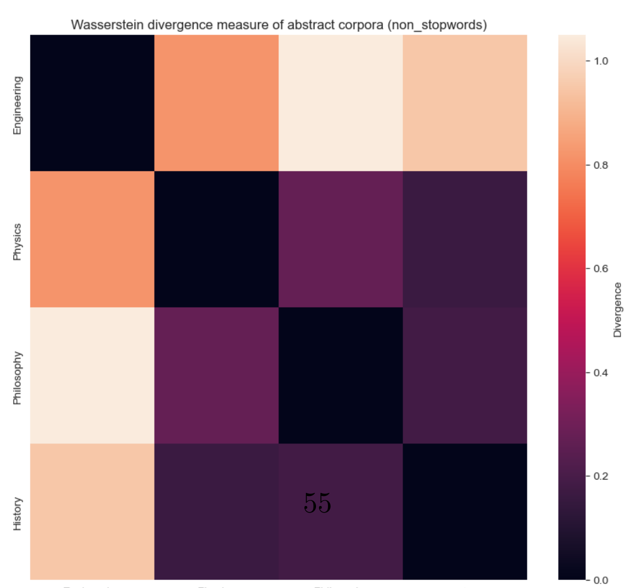
[81]:
```
helper_functions.distributional_distance(abstract_corpus_concatenated,␣
↪"Wasserstein", fieldsOfStudy, df_index)
```

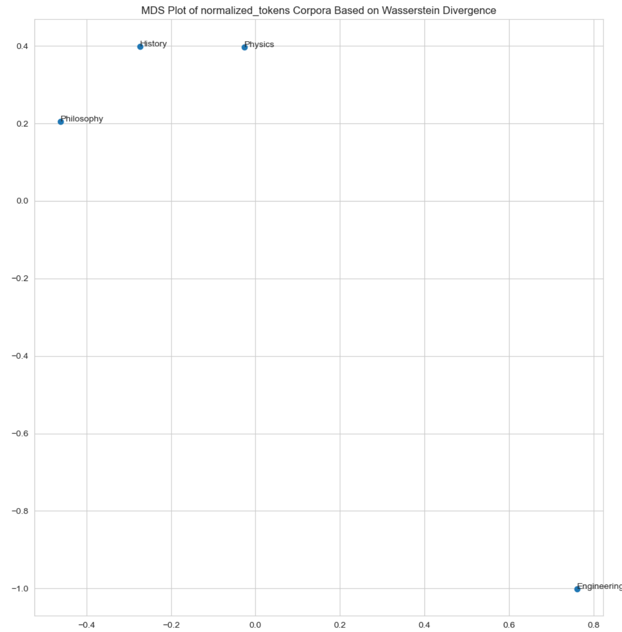
```
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:601: UserWarning: The MDS API has changed.
``fit`` now constructs an dissimilarity matrix from data. To use a custom
dissimilarity matrix, set ``dissimilarity='precomputed'``.
  warnings.warn(
/Users/samcong/anaconda3/lib/python3.11/site-
packages/sklearn/manifold/_mds.py:298: FutureWarning: The default value of
`normalized_stress` will change to `'auto'` in version 1.4. To suppress this
warning, manually set the value of `normalized_stress`.
  warnings.warn(
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:380: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig_heatmap.show()
/Users/samcong/Library/CloudStorage/OneDrive-TheUniversityofChicago/Winter
2024/MACS 60000 Computational Content Analysis/Weekly_Exercises/week
2/helper_functions.py:381: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```

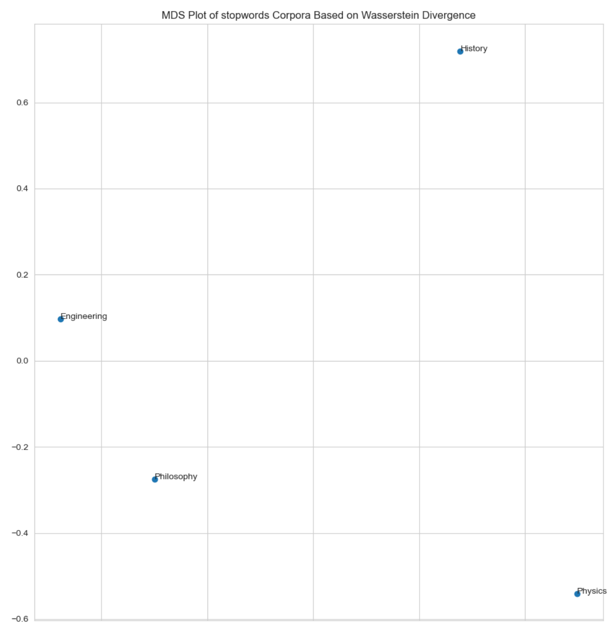
```
fig_mds.show()
```


Wasserstein divergence measure of abstract corpora (normalized_tokens)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8
1.0


Wasserstein divergence measure of abstract corpora (stopwords)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8


Wasserstein divergence measure of abstract corpora (non_stopwords)
Engineering
Physics
Philosophy
History
Divergence
0.0
0.2
0.4
0.6
0.8
1.0


MDS Plot of normalized_tokens Corpora Based on Wasserstein Divergence
History
Physics
Philosophy
Engineering
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


MDS Plot of stopwords Corpora Based on Wasserstein Divergence
History
Engineering
Philosophy
Physics
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4


MDS Plot of non_stopwords Corpora Based on Wasserstein Divergence
Philosophy
History
Engineering
Physics
0.3
0.2
0.1
0.0
−0.1
−0.2
−1.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50