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.

```
# 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)
```

Scrape abstract from Psychology Papers

	Title	Author	Abstract
0	The Control of Single-color and Multiple-color...	A. Grubert, N. Carlisle, M. Eimer	The question whether target selection in visua...
1	Non-pharmacologic and pharmacologic treatments...	K. Atchison, J. Watt, Delaney Ewert, A. Toohey...	BACKGROUND\nolder adults living in long-term c...
2	Confirmation bias in information search, inter...	Dáša Vedejová, V. Čavojová	Abstract Confirmation bias is often used as an...
3	Meta-analysis on the long-term effectiveness o...	A. Hilbert, D. Petroff, S. Herpertz, R. Pietro...	OBJECTIVE\nLong-term effectiveness is a critic...
4	On the search for a selective and retroactive ...	F. Kalbe, L. Schwabe	Storing motivationally salient experiences pre...

Update the clean_raw_text function about the unicode part:

```
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()
```

```
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
```

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

```
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
```

```
# Construct the nltk Text object
abstract_corpus_Text = nltk.Text(abstract_corpus_token)
```

What relates to the token “memory’?

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.

```
abstract_corpus_Text.count("memory")
```

```
74
```

```
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
```

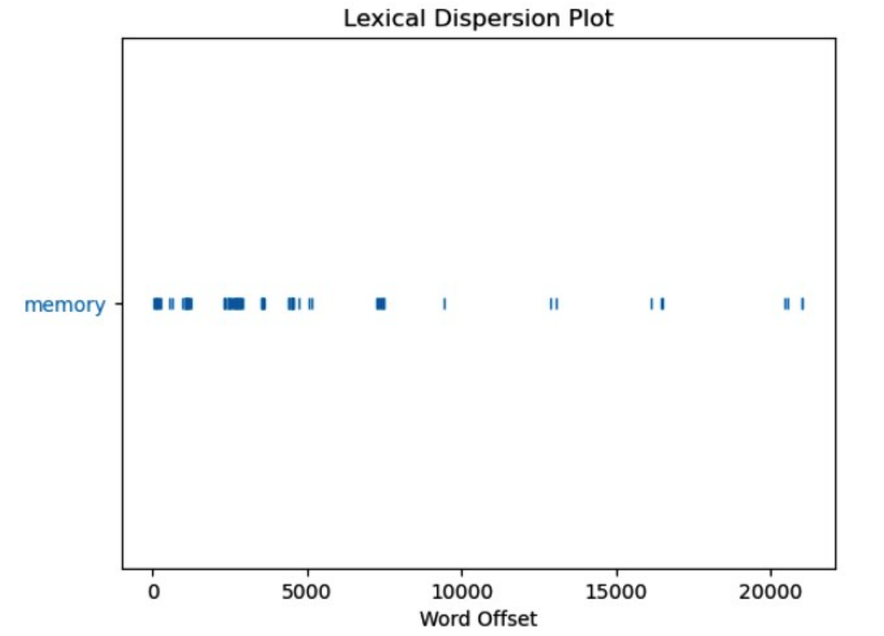
```
abstract_corpus_Text.similar("memory", num=10)
```

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

```
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
```

```
abstract_corpus_Text.dispersion_plot(["memory"])
```




```
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
```

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.

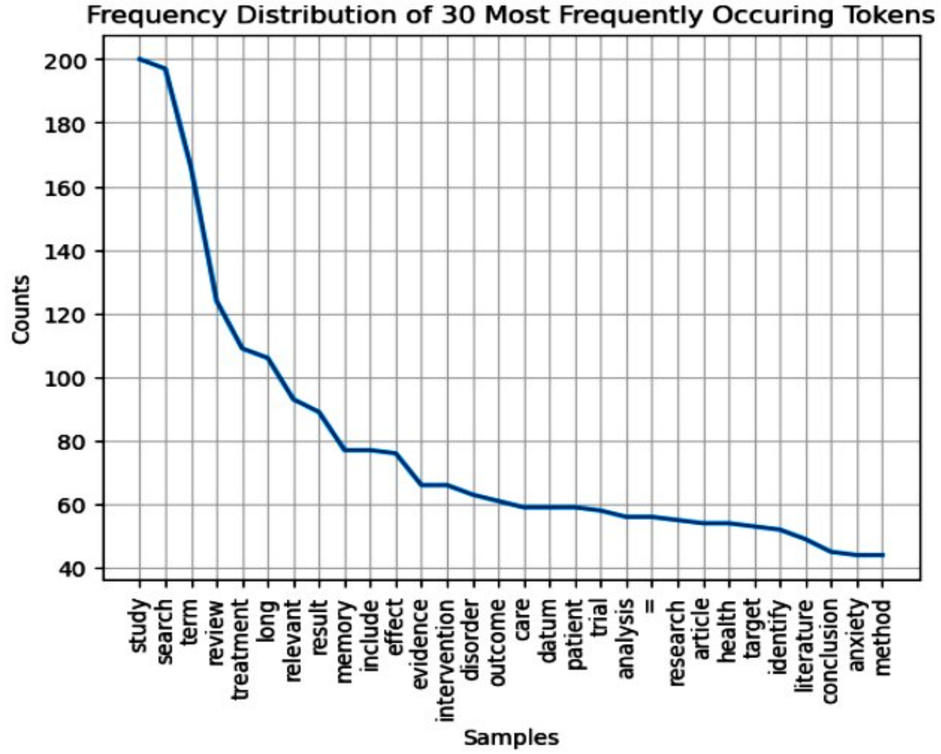
```
abstract_corpus_token_normalized = helper_functions.normalizeTokens(abstract_corpus_token)
```

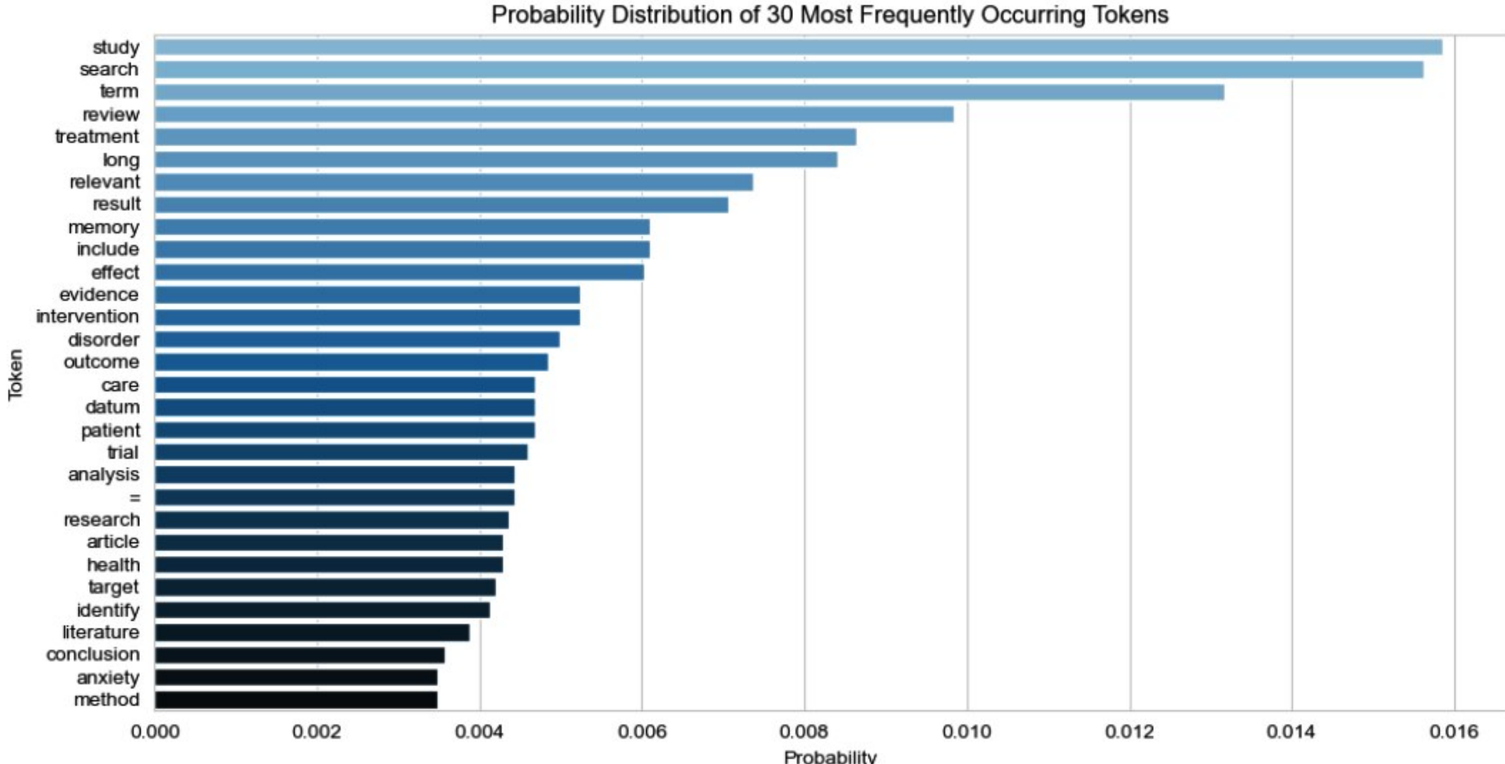
```
print("Number of tokens for the abstract corpus: {}".format(len(abstract_corpus_token)))
print("Number of normalized tokens for the abstract corpus: {}".format(len(abstract_corpus_token_normalized)))
```

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

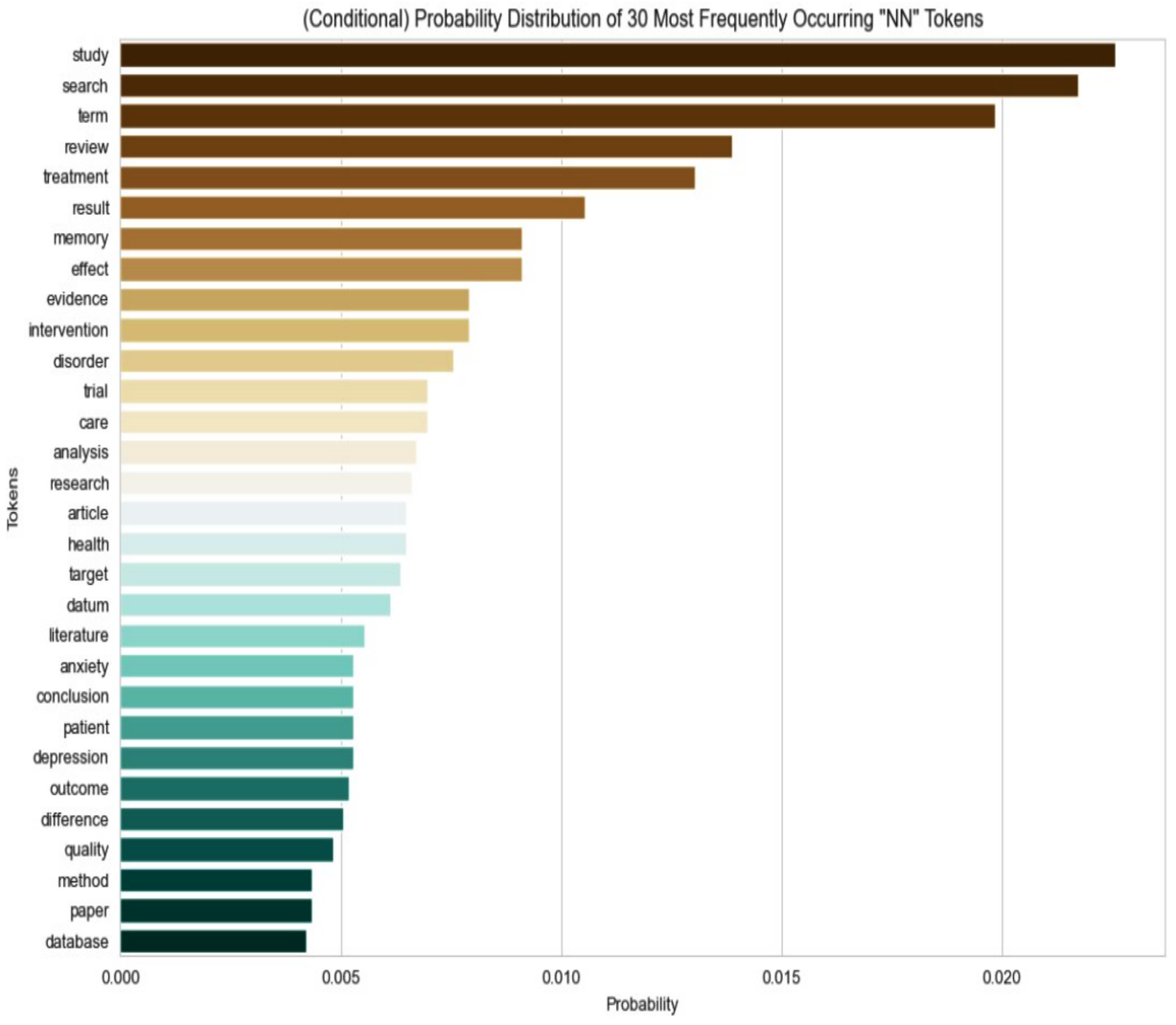
```
abstract_corpus_fdist = nltk.FreqDist(abstract_corpus_token_normalized)
abstract_corpus_fdist.plot(30, title='Frequency Distribution of 30 Most Frequently Occuring Tokens'
```

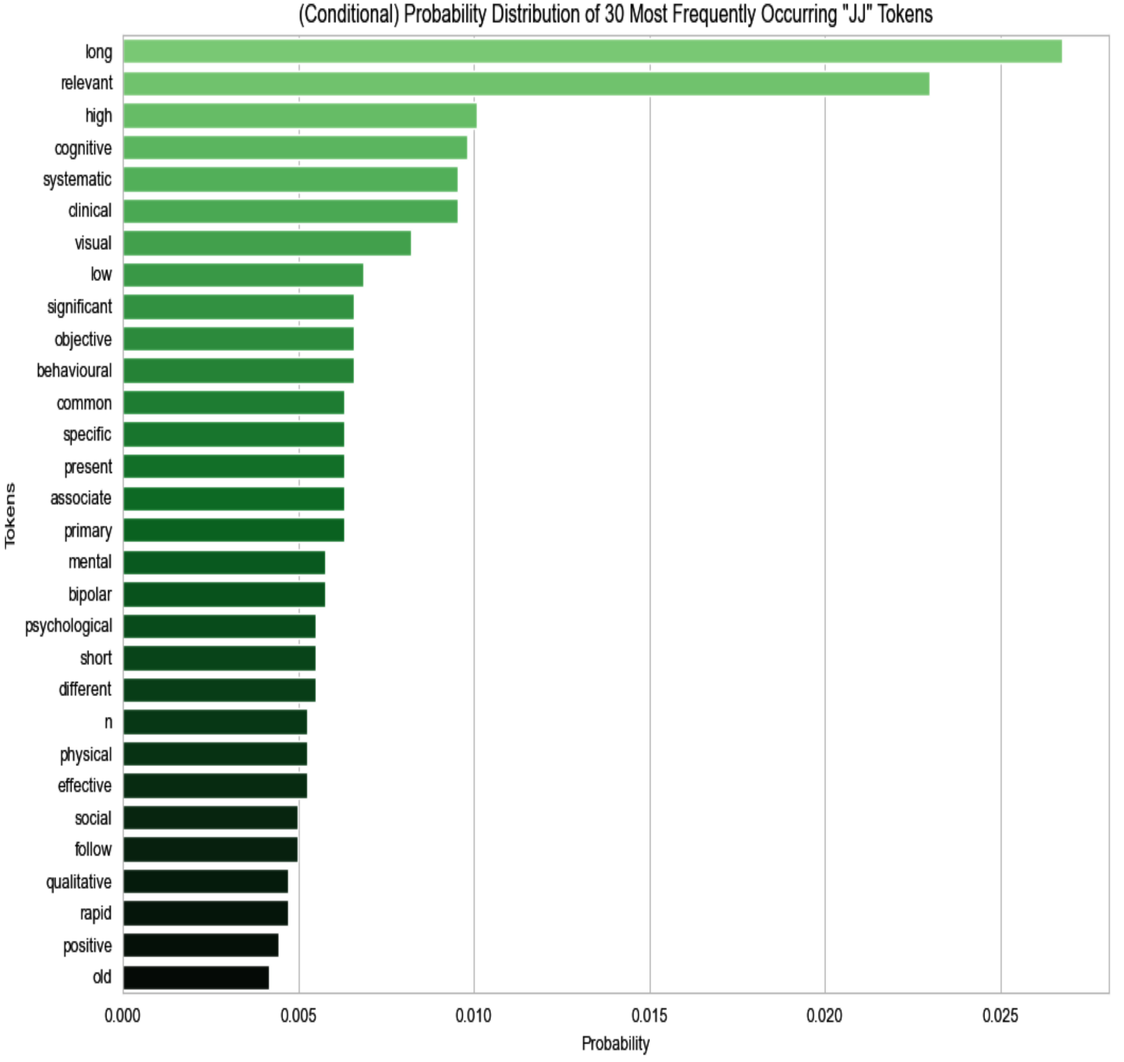
Frequency and Probability Distributions of Normalized Tokens





Conditional Probability Distributions of “NN” and “JJ” tokens





Word Cloud of Normalized Tokens in Psychology Paper Abstract

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?

Three Types of POS Tags of Normalized Tokens (1)

```
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)])
```

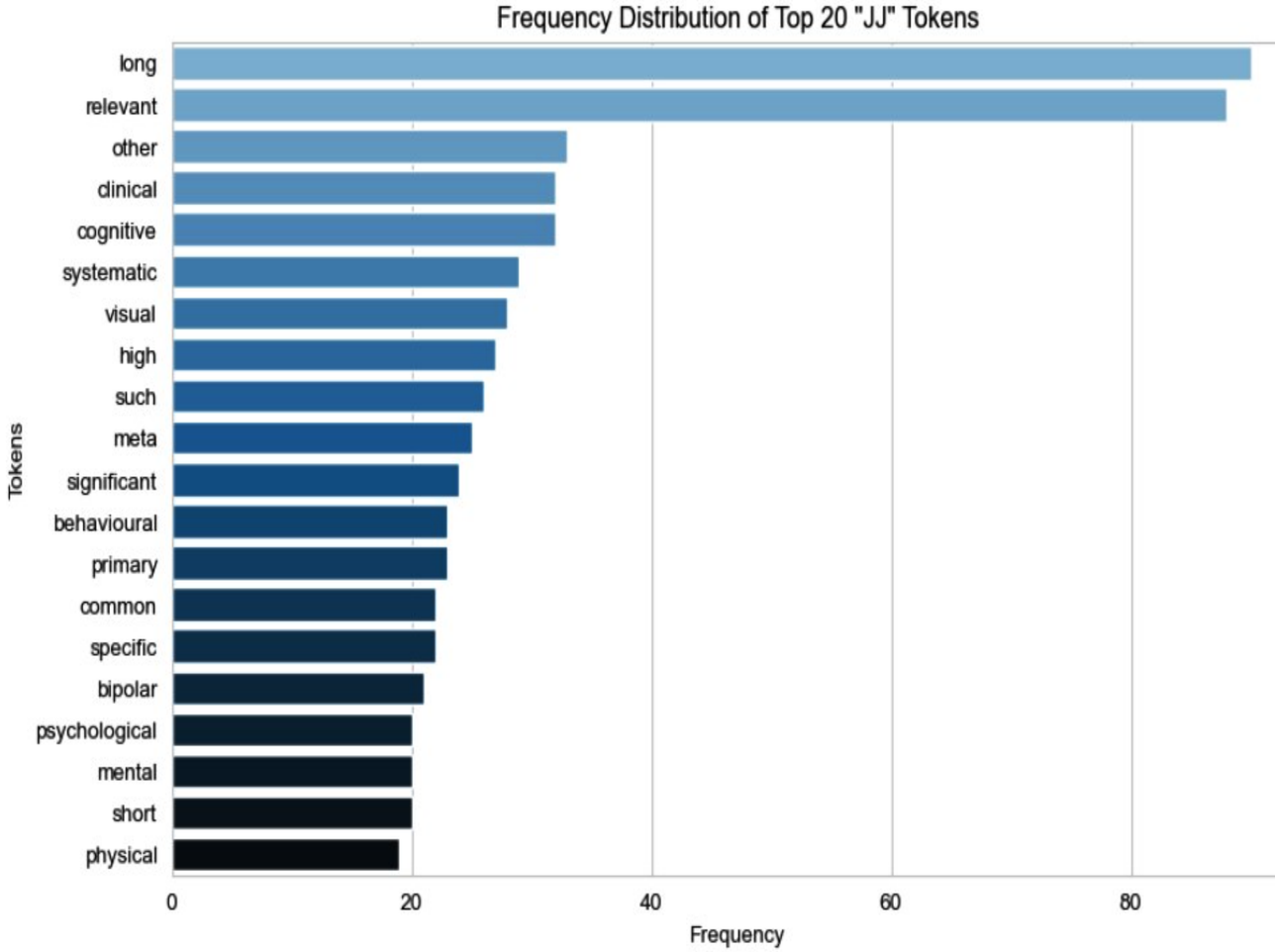
```
abstract_corpus_df['Sentence'].head()
```

```
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
```

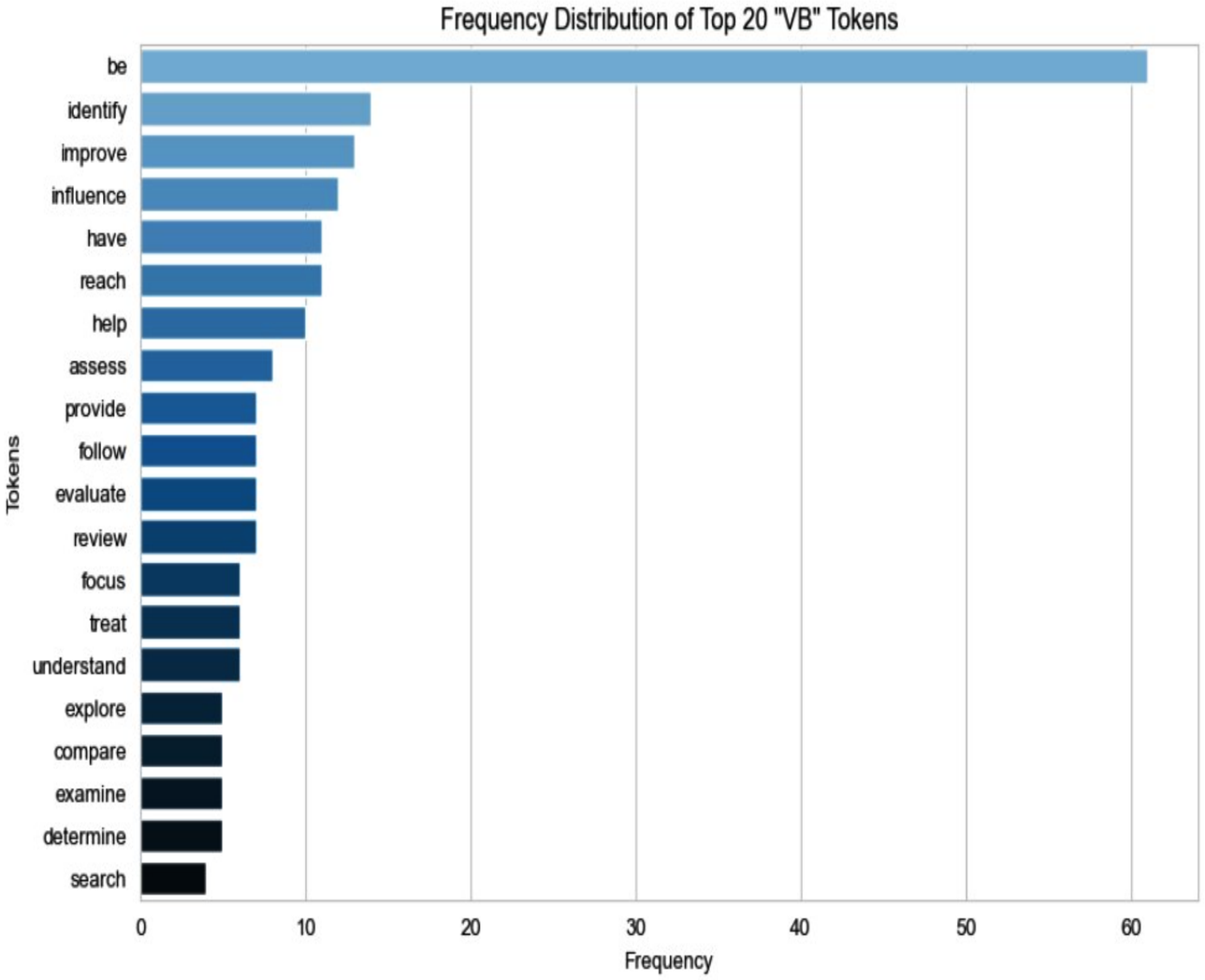
```
abstract_corpus_df['POS_Sentence'] = abstract_corpus_df['Sentence']./
    apply(lambda x: helper_functions.tag_sents_pos(x))
```

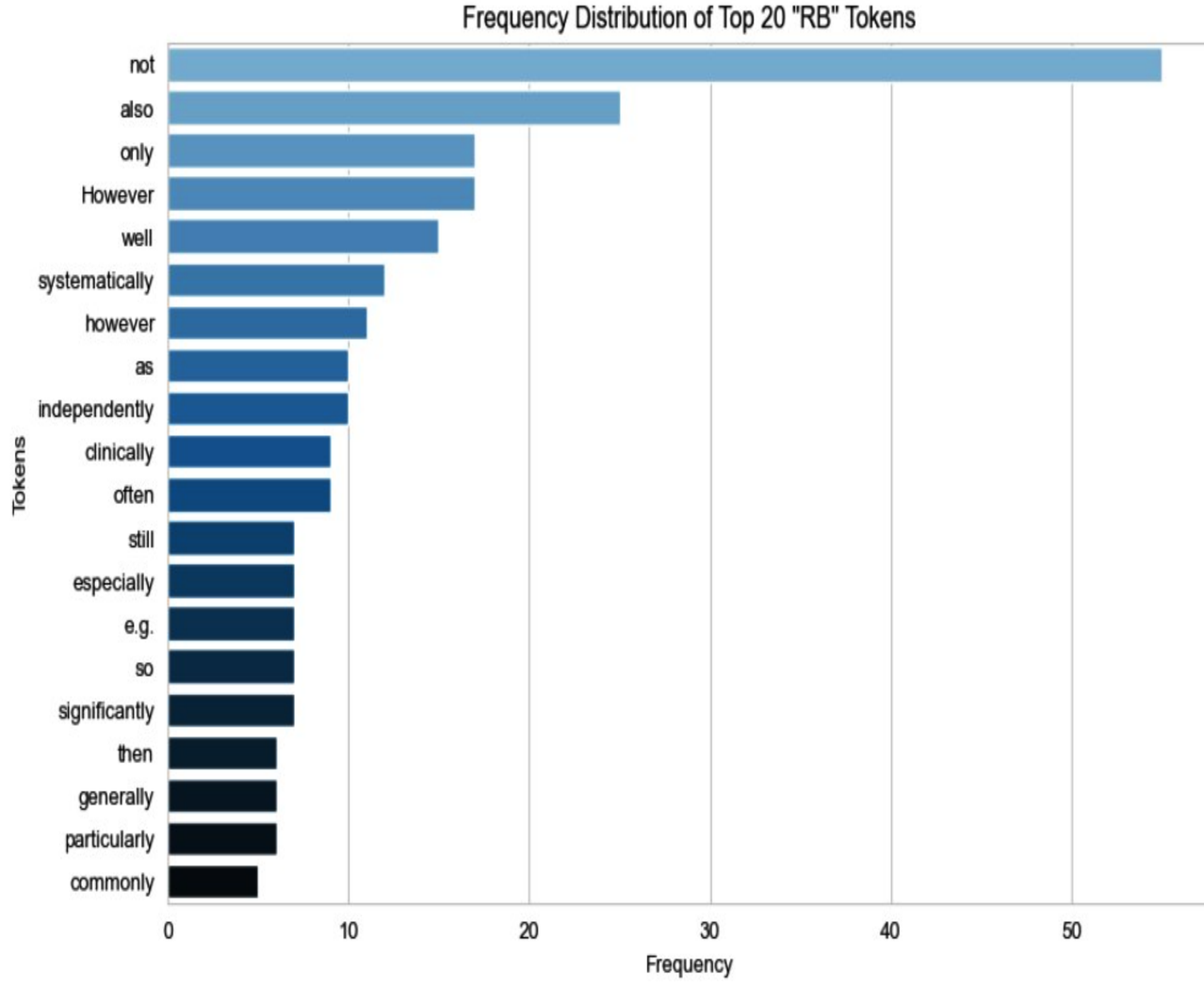
```
abstract_corpus_df['POS_Sentence'].head()
```

```
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
```



Three Types of POS Tags of Normalized Tokens (2 & 3)





Conditional Association of POS tags

JJ and NN

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

```
[(('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)]
```

RB and JJ

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

```
[(('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)]
```

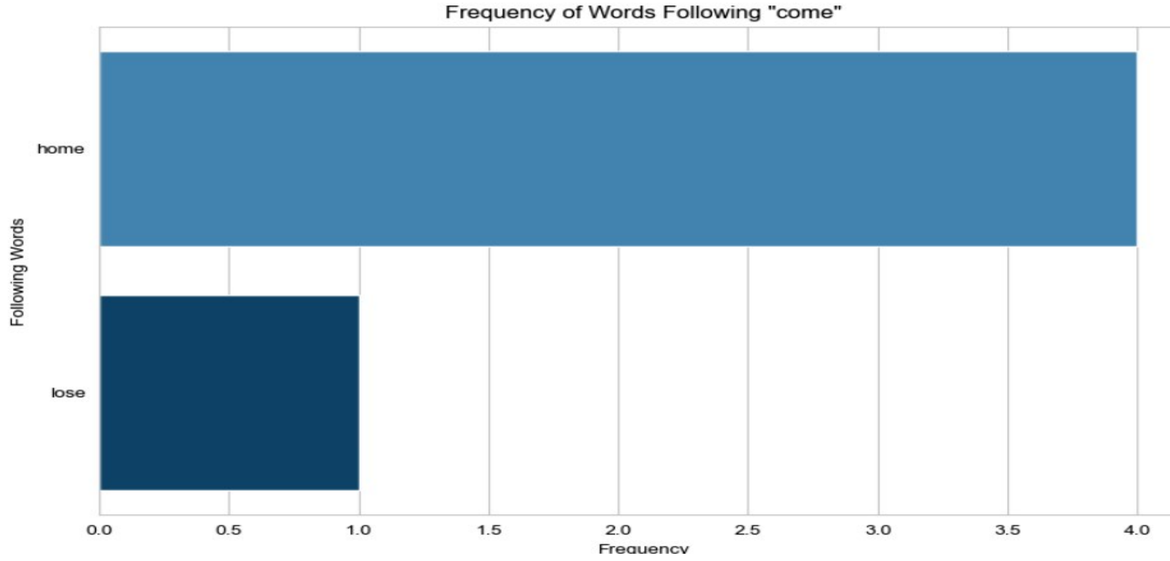
Exercise 4

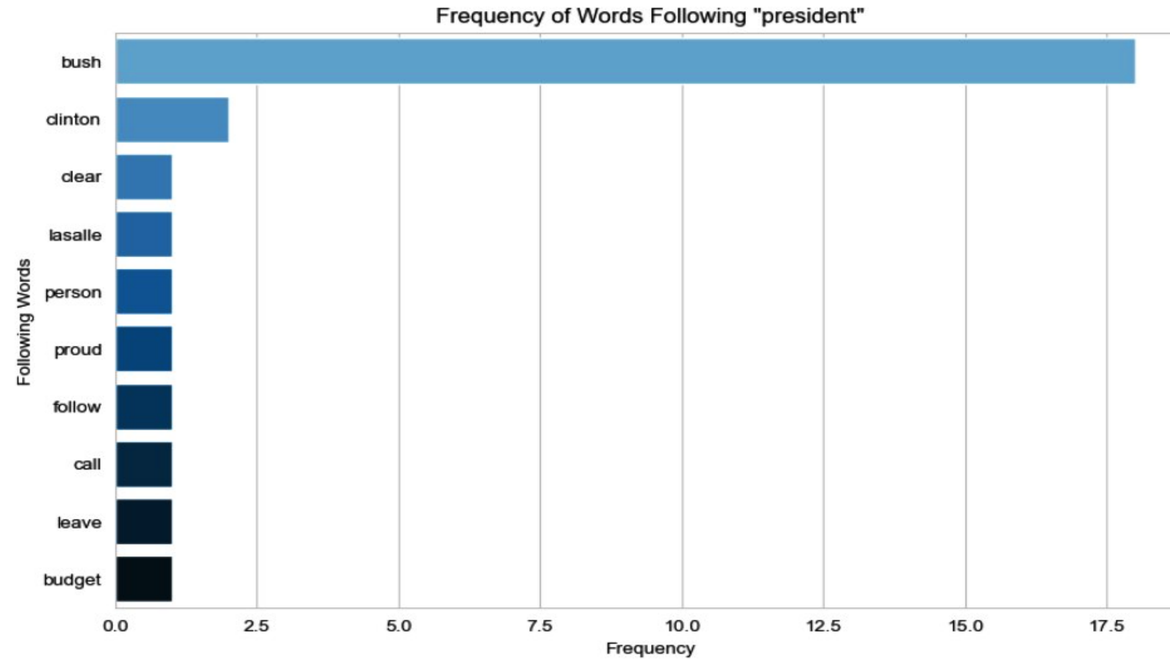
Identify statistically significant bigrams and trigrams. Explore whether these collocations are idiomatic and so irreducible to the semantic sum of their component words. You can do this by examination of conditional frequencies (e.g., what else is 'united' besides the 'United States'). If these phrases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

Statistically significant bigrams

```
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
```

```
[('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')]
```





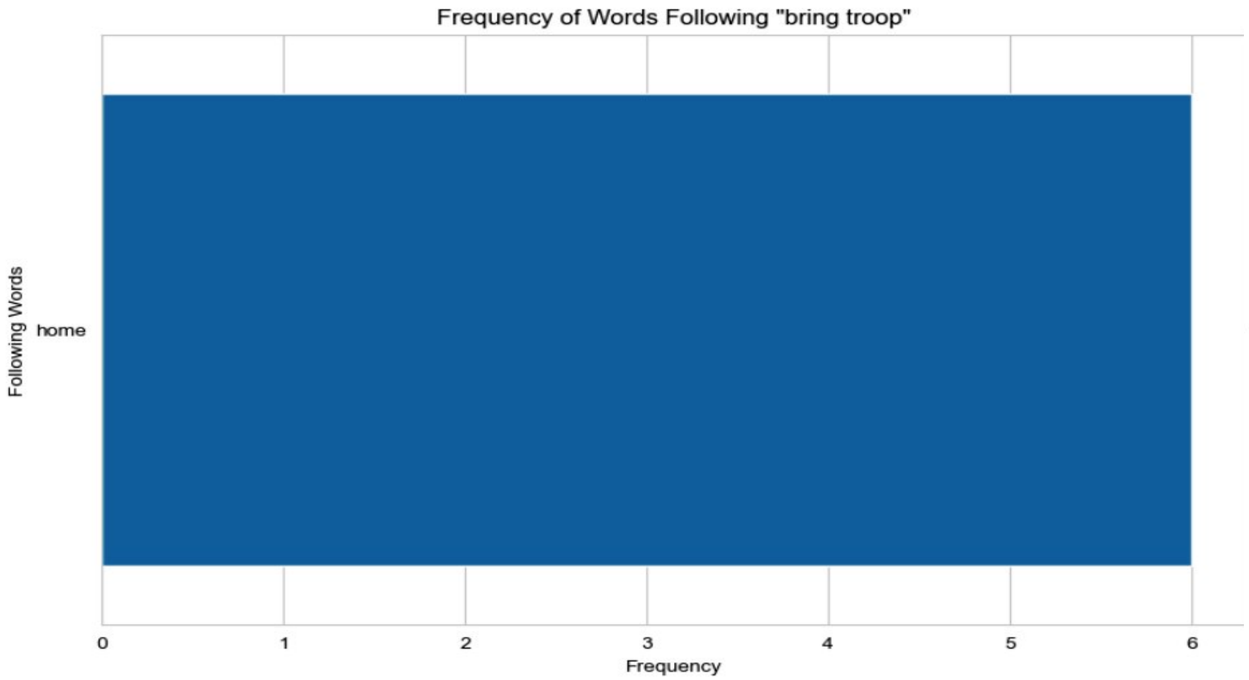
Statistically significant trigrams

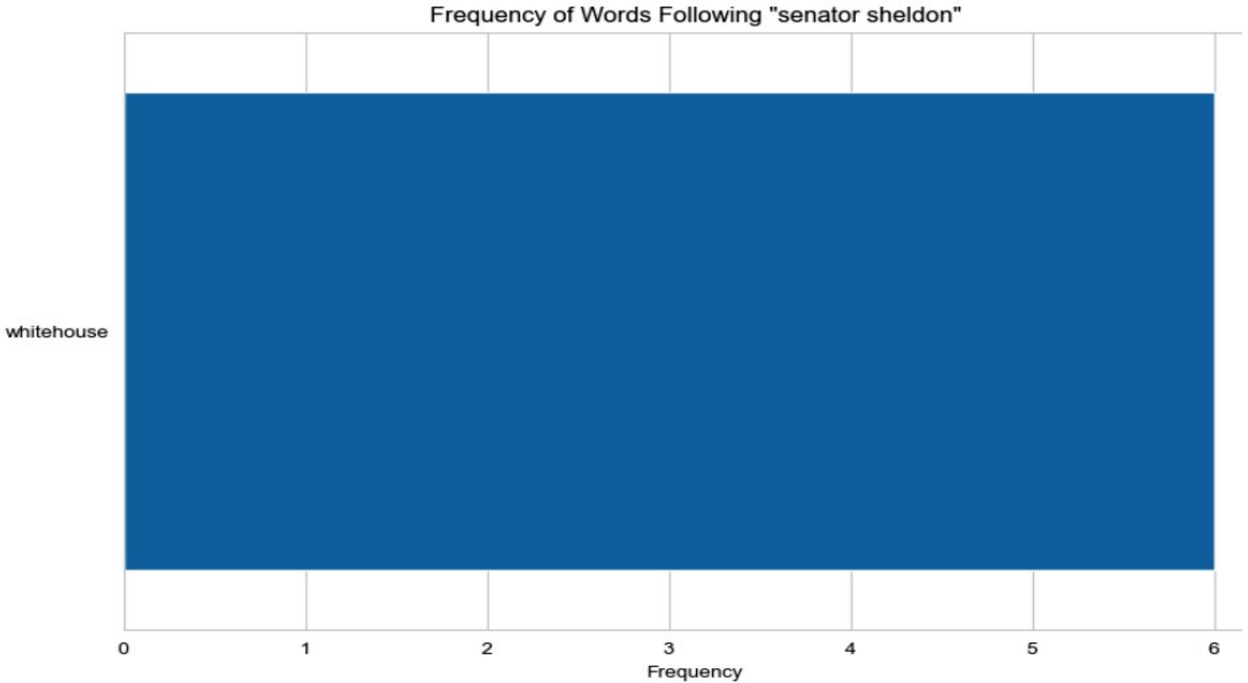
```
t_critical_trigram = helper_functions.find_t_critical(whTrigrams.N)

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

whTrigrams_significant
```

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

```
# 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)
```

```
helper_functions.plot_ngram_frequencies(("bring", "troop"), trigram_cfd)
```





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.

Perform NER

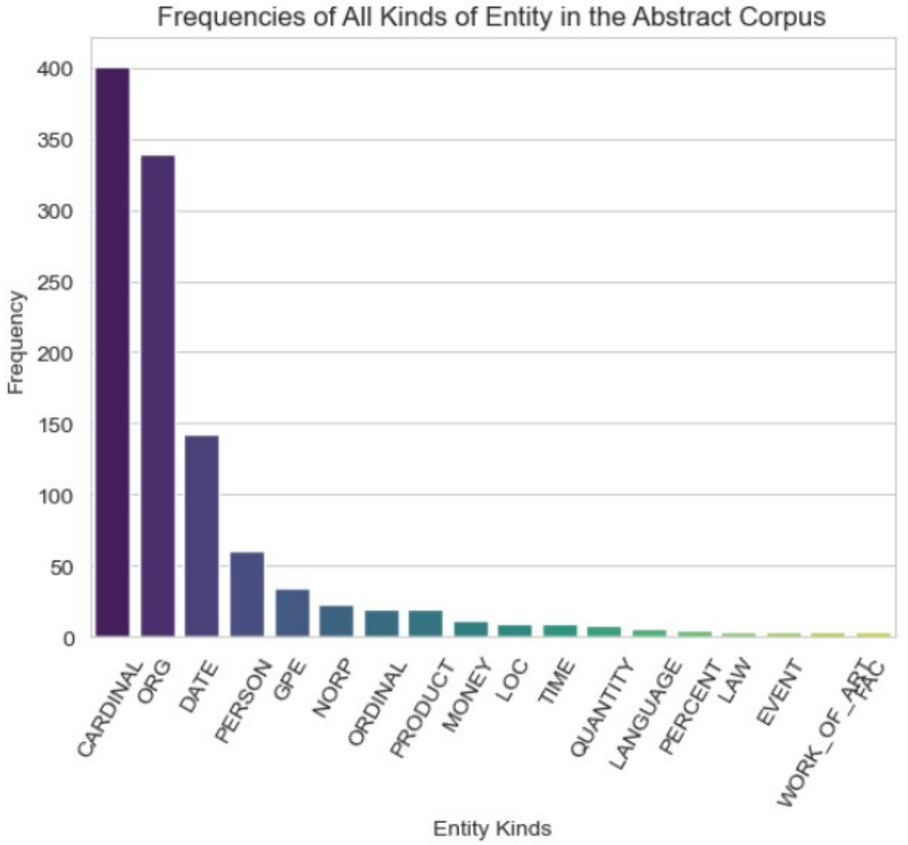
```
abstract_corpus_df['classified_sents'] = abstract_corpus_df['Sentence']./
    apply(lambda x: helper_functions.tag_sents_ner(x))
abstract_corpus_df['classified_sents'].head()
```

```
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:

```
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)
```

```
18
```



Calculate Precision, Recall, and F-Score for NER Classification (ORG tags as the sample)

```python
# 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.
```

Manually code whether the tags are indeed ORG

```python
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"
}
```

```python
# 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
```

ORG Precision	ORG Recall	ORG F-Score
0.26	0.97	0.41

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?

Parse the Corpus

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

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

parsed_corpus[:10]
```

```
[The question whether target selection in visual search can be effectively controlled by simultaneous attentional t
emplates 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 l
ong 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 dep
loyment to color defined target objects relative to single color search,
 Importantly these costs were the same in constant color and variable color blocks]
```

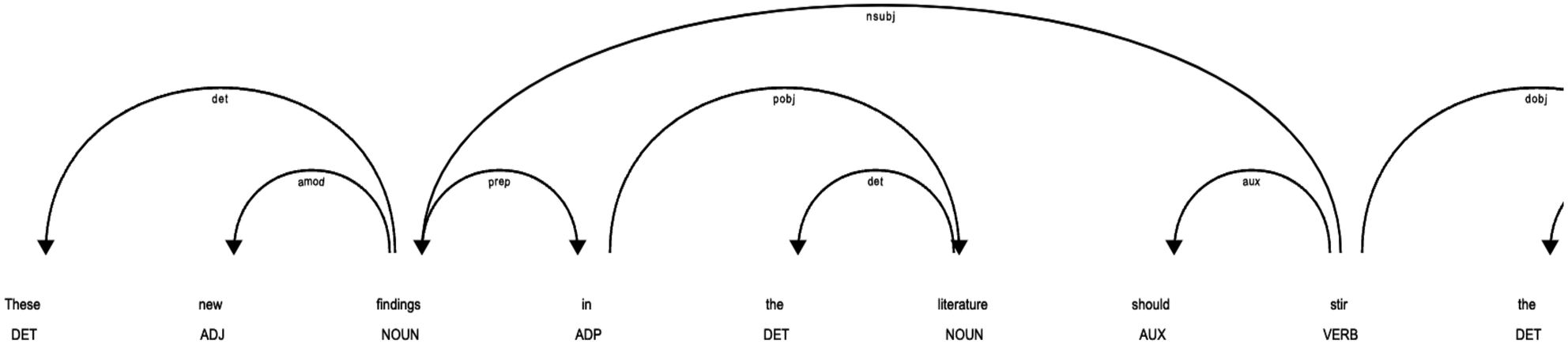
Depth of the Phrase Structure and Dependency Parse Trees Nested

```
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()
```

```
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
```

Depth of 1

Depth of 7

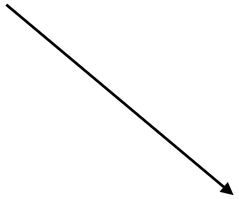


Extract Five Linguistic Features and Apply Them to A Focal Set of Words

```
linguistic_features = {word: {'noun_phrases': [], 'adjectives': [], 'verbs': [], 'dependencies': [], 'co_occurring':

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)
```

Five Linguistic Features

1. Noun phrases containing the focal words
2. Adjectives modifying the focal words
3. Verbs associated with the focal Words (either subjects or objects)
4. Dependency relations where the focal words are involved
5. Named entities that frequently co-occur with the focal words

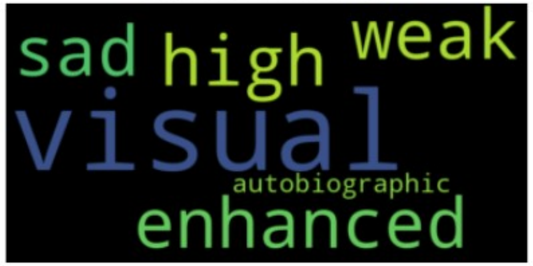


	memory	disorder
noun_phrases	[long term memory, working memory, color memor...	[disorder BED, disorder, simple phobia obsessi...
adjectives	[enhanced, high, weak, visual, visual, visual,...	[obsessive, compulsive, traumatic, specific, c...
verbs	[working, working, working, showed, working, e...	[eating, excluding, impairs, generalized, obse...
dependencies	[(memory, pobj, in), (memory, pobj, in), (memo...	[(disorder, compound, BED), (disorder, dobj, e...
co_occurring	[CDA, CDA, three, four, only one, four, Bayesi...	[BED, RESULTS Effectiveness, up to 12 months, ...

Visualization of Results

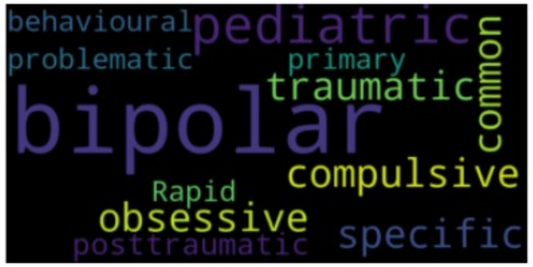
Noun Phrases for memory

Adjectives for memory

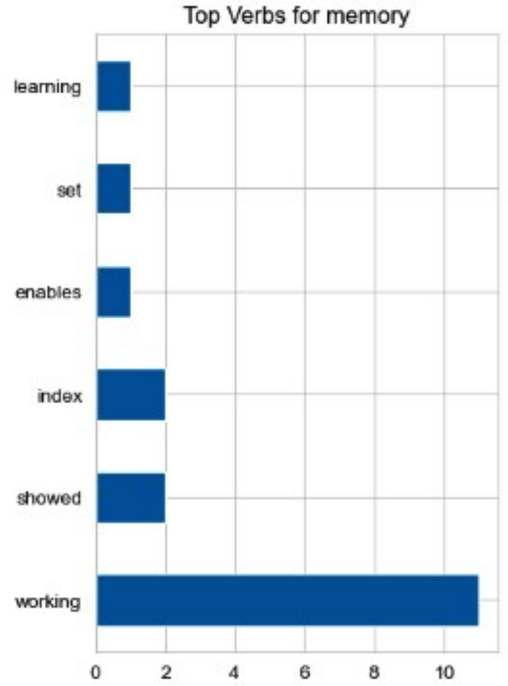


Noun Phrases for disorder

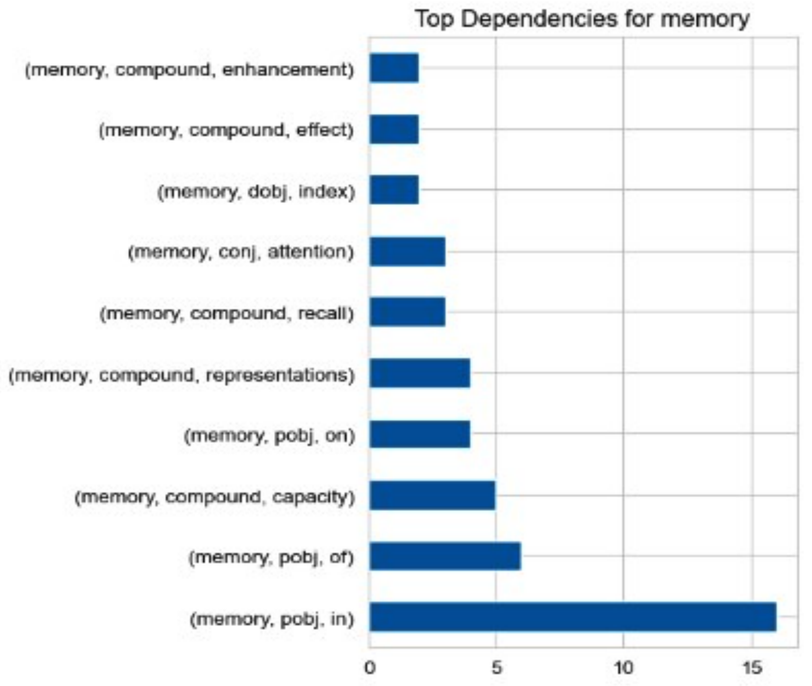
Adjectives for disorder



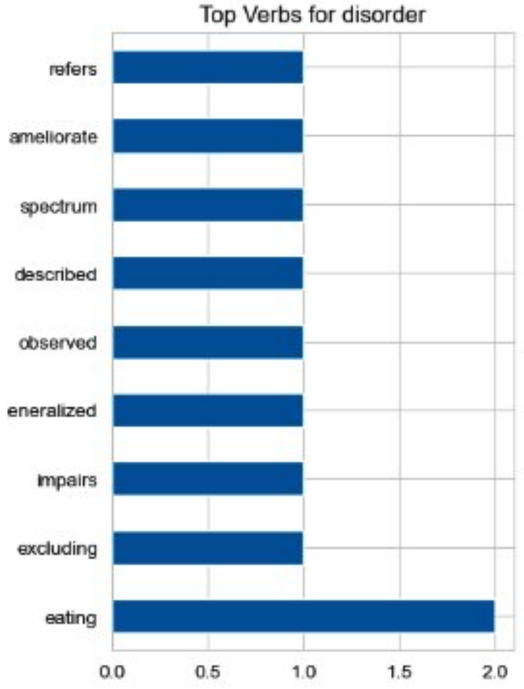
Top Verbs for memory



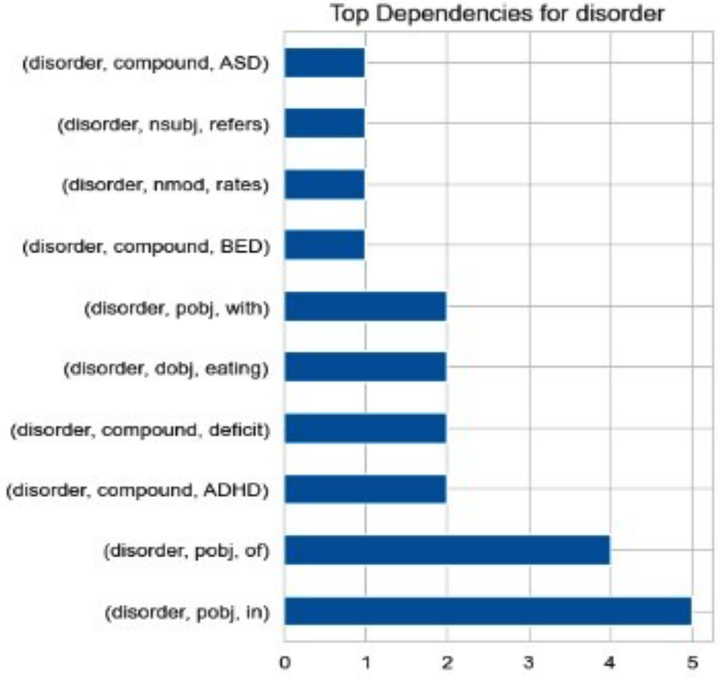
Top Dependencies for memory



Top Verbs for disorder



Top Dependencies for disorder



Exercise 7

Calculate the KL and χ^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?

Scrape Abstract from Multiple Fields of Study

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

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

```
# 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)
```

Fields_of_study	normalized_tokens	stopwords	non_stopwords
Engineering	[introduction, engineering, psychology, human,...	[make, call, call, show, make, have, make, mak...	[introduction, engineering, psychology, human,...
Physics	[primer, attempt, plasma, physics, computer, s...	[make, call, give, one, part, see, one, take, ...	[primer, attempt, plasma, physics, computer, s...
Philosophy	[draw, phenomenological, tradition, philosophy...	[name, take, take, well, call, keep, name, go,...	[draw, phenomenological, tradition, philosophy...
History	[grateful, editor, british, journal, sociology...	[put, well, give, hundred, show, name, being, ...	[grateful, editor, british, journal, sociology...

Calculate Four Types of Distributional Divergence

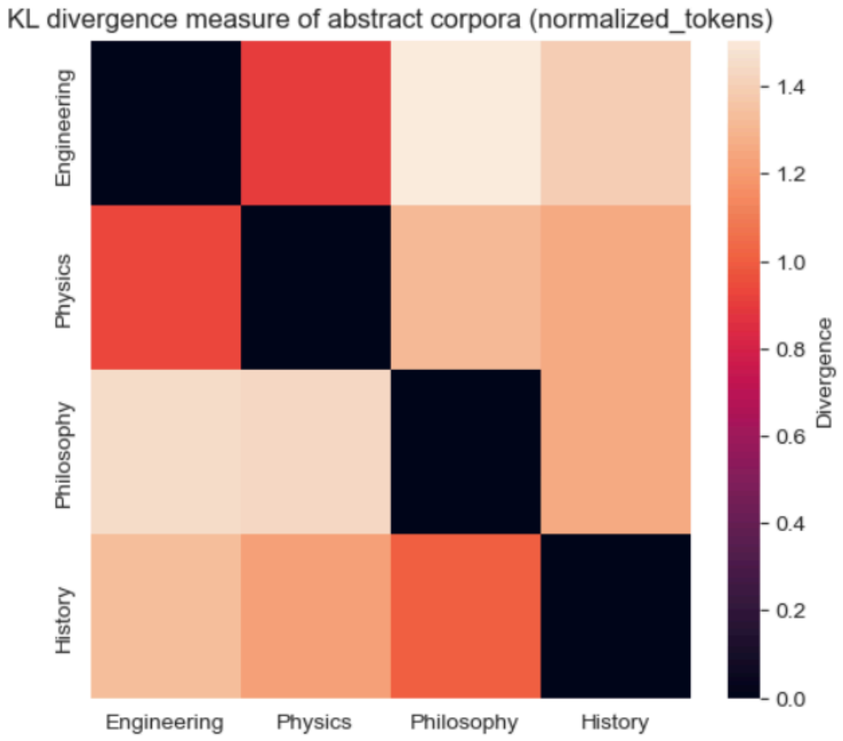
```
helper_functions.distributional_distance(abstract_corpus_concatenated, "KL", fieldsOfStudy, df_index)
```

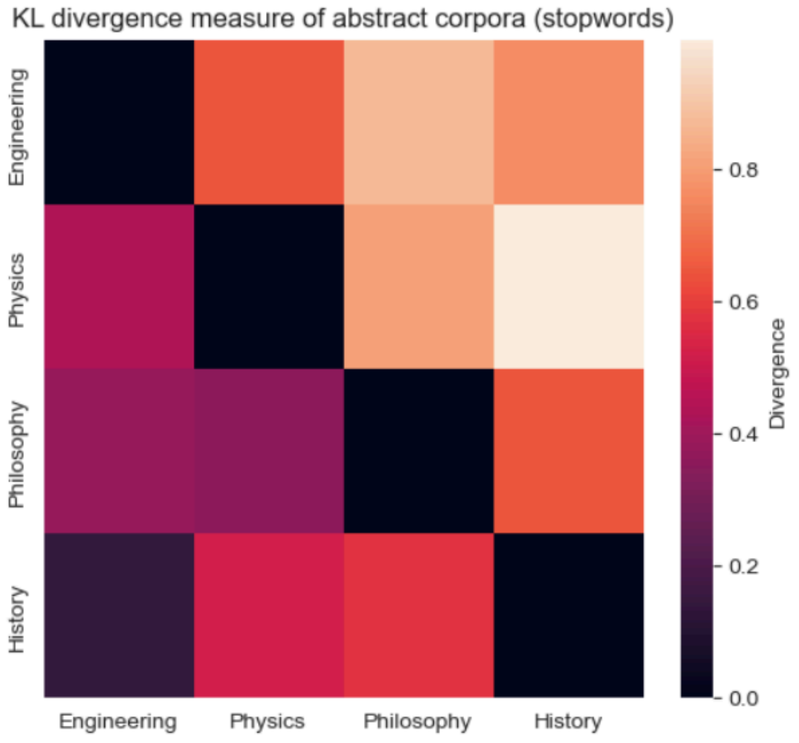
```
helper_functions.distributional_distance(abstract_corpus_concatenated, "Chi2", fieldsOfStudy, df_index)
```

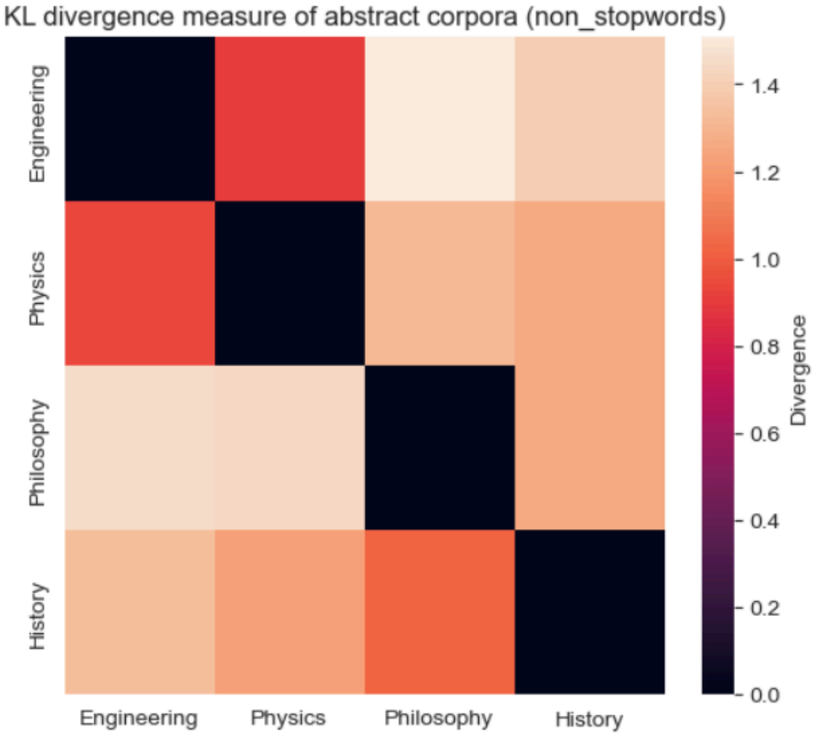
```
helper_functions.distributional_distance(abstract_corpus_concatenated, "KS", fieldsOfStudy, df_index)
```

```
helper_functions.distributional_distance(abstract_corpus_concatenated, "Wasserstein", fieldsOfStudy, df_index)
```

KL Measure Heatmap







KL Measure Multidimensional Scaling