5_pmh_analysis

July 30, 2023

[1]:
```python
import json
import os
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
import numpy as np
import json5
import spacy
import medspacy
import random
from medspacy.ner import TargetMatcher, TargetRule
from medspacy.visualization import visualize_ent, visualize_dep
from spacy.tokens import Span
import sys

parent_dir = os.path.abspath("..")
if parent_dir not in sys.path:
    sys.path.append(parent_dir)
from path import DATA_PROCESSED_DOCUMENTS_DIR
```

[2]:
```python
folder_location = os.path.join(
    DATA_PROCESSED_DOCUMENTS_DIR / "black-or-african-american"
)
b_docs = []
w_docs = []
for filename in os.listdir(folder_location):
    file_location = os.path.join(folder_location, filename)
    if os.path.isfile(file_location):
        with open(file_location) as d:
            try:
                file_contents = d.read()
                content = json.loads(file_contents)
                b_docs.append(content)
            except Exception as e:
                try:
                    # pull of first and last line, gpt sometimes response with␣
 ↪a leading ```json and ends with ```
                    tmp = file_contents.splitlines(True)
```

```
                    while "{" not in tmp[0]:
                        tmp = tmp[1:]
                    while "}" not in tmp[-1]:
                        tmp = tmp[:-1]
                    for i, line in enumerate(tmp):
                        if "{" not in line and "}" not in line:
                            if line[-2:] != ",\n":
                                tmp[i] = line.strip() + ",\n"
                    try:
                        tmp = "".join(tmp)
                        content = json5.loads(tmp)
                        b_docs.append(content)
                    except ValueError as e:
                        try:
                            tmp = file_contents
                            tmp = tmp.replace("\n", " ")
                            tmp = tmp.replace("\r", " ")
                            content = json5.loads(tmp)
                            w_docs.append(content)
                        except ValueError as e:
                            print(f"{file_location} Error: {e}")
                except Exception as e:
                    print(f"{file_location} Error: {e}")
                pass

folder_location = os.path.join(DATA_PROCESSED_DOCUMENTS_DIR /␣
  ↪"white-or-caucasian")
for filename in os.listdir(folder_location):
    file_location = os.path.join(folder_location, filename)
    if os.path.isfile(file_location):
        with open(file_location) as d:
            try:
                file_contents = d.read()
                content = json.loads(file_contents)
                w_docs.append(content)
            except Exception as e:
                try:
                    # pull of first and last line, gpt sometimes response with␣
  ↪a leading ```json and ends with ```
                    tmp = file_contents.splitlines(True)
                    while "{" not in tmp[0]:
                        tmp = tmp[1:]
                    while "}" not in tmp[-1]:
                        tmp = tmp[:-1]
                    for i, line in enumerate(tmp):
                        if "{" not in line and "}" not in line:
```

```
                            # check if line ends with a comma and newline, add␣
↪if not
                            if line[-2:] != ",\n":
                                tmp[i] = line.strip() + ",\n"
                    try:
                        tmp = "".join(tmp)
                        content = json5.loads(tmp)
                        w_docs.append(content)
                    except ValueError as e:
                        try:
                            tmp = file_contents
                            tmp = tmp.replace("\n", " ")
                            tmp = tmp.replace("\r", " ")
                            content = json5.loads(tmp)
                            w_docs.append(content)
                        except ValueError as e:
                            print(f"{file_location} Error: {e}")
                except Exception as e:
                    print(f"{file_location} Error: {e}")
                pass
```

```
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_annetta-
williams_61_f_1690475007_h5knGiSKhpP7JtSHSdsyse.txt Error: <string>:1 Unexpected
"," at column 2092
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_keisha-
armstrong_54_f_1690474215_PBEgVYogZUstMp6iSv2Gj5.txt Error: <string>:1
Unexpected """ at column 1014
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_leonard-
douglas_64_m_1690473265_mQHCjxaum947RJx7GwcuZa.txt Error: <string>:1 Unexpected
"c" at column 310
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_earnestine-
roberts_56_f_1690472896_GafFWpG8ow7FpEey7Mouu6.txt Error: <string>:1 Unexpected
"c" at column 370
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_latoya-
lee_40_f_1690474127_RvMdAxCNmK9sheUY3GtUYm.txt Error: <string>:1 Unexpected "w"
at column 411
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_reginald-
burney_58_m_1690472138_a9PF7H7gMP8zvphSj7i2Ex.txt Error: <string>:1 Unexpected
"`" at column 1
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_uriel-
martin_20_m_1690472443_SH7RRw8J6LkftrnbCPqrjd.txt Error: <string>:1 Unexpected
```

```
"r" at column 490
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_effie-
levels_88_f_1690473788_BRgumXrrq2nbaxkt2ydyPp.txt Error: <string>:1 Unexpected
"`" at column 1
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_darnell-
beliard_65_m_1690474490_5HRytSNNKPBebAMpLkXvRY.txt Error: <string>:1 Unexpected
"c" at column 231
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_terra-
clark_36_f_1690474020_m5SoxpjuuwY2tcCXEpAVB8.txt Error: <string>:1 Unexpected
""" at column 965
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_essie-
abera_89_f_1690472710_dyiFdzZmEjQAATB4V7GxSa.txt Error: <string>:1 Unexpected
"`" at column 1
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_emma-
dillard_93_f_1690472115_PS94c8chnfE8ceZo23sLZD.txt Error: <string>:1 Unexpected
"t" at column 375
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/black-or-
african-american/gpt-3.5-turbo-0613_black-or-african-american_raphael-
turner_39_m_1690474289_7w8JJG6gmogaKUAuf64yb5.txt Error: <string>:1 Unexpected
""" at column 1098
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_ava-
kessinger_20_f_1690475550_XnHpDB8FmjkSZeNi4a78h8.txt Error: list index out of
range
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_lon-
wright_62_m_1690475186_9HDhFWiiTtD8arfYemM2pd.txt Error: <string>:1 Unexpected
"r" at column 1029
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_bob-
luhman_65_m_1690477150_Q3KifjDzrQRMhXXtjWxqQi.txt Error: list index out of range
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_alessandra-
hughes_23_f_1690476930_ZqaEARfZXhQVCFKEkd3h27.txt Error: <string>:1 Unexpected
"`" at column 1
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_lorin-
ranta_33_f_1690476153_GLbJsFqBSdtk9Spz9xUeoS.txt Error: <string>:1 Unexpected
""" at column 710
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_elyssa-
shaw_37_f_1690476946_LiDBthZBNfLcTBEUS5X47L.txt Error: <string>:1 Unexpected "R"
at column 713
```

```
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_cathleen-
pitts_57_f_1690476542_Ye76HxZKTFEstiYqNtg7yq.txt Error: <string>:1 Unexpected
"r" at column 693
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_liam-
bowman_20_m_1690476809_KYzgmtj9tHcWWZEGq5gGs3.txt Error: <string>:1 Unexpected
""" at column 599
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_shari-
benedetti_62_f_1690477372_YbdRLZ262uSq5m7tbxvc2t.txt Error: <string>:1
Unexpected """ at column 839
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_tana-
harrell_18_f_1690475752_9ZTdso8gbz4ZnDR4yp4BS6.txt Error: <string>:1 Unexpected
""" at column 913
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_kinga-
mindlin_19_f_1690475593_nZhi5aB4ErfJ4KXEJpVVTU.txt Error: <string>:1 Unexpected
""" at column 691
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_anita-
pace_67_f_1690475526_Yku8scB22Bf7nRw25UuK5W.txt Error: <string>:1 Unexpected """
at column 813
/Users/chris/Documents/gpt-medical-bias/data/processed/documents/white-or-
caucasian/gpt-3.5-turbo-0613_white-or-caucasian_enid-
scott_52_f_1690475894_fpT6BmC2jZNhQmnXn4Z4Sy.txt Error: <string>:1 Unexpected
""" at column 1925
```

[3]:
```python
print(len(b_docs))
print(len(w_docs))
```

```
4982
4992
```

[4]:
```python
# Grab the text from each document's past medical history section
b_pmh = []
for doc in b_docs:
    if doc.get("past_medical_history") is not None:
        b_pmh.append(doc.get("past_medical_history"))

w_pmh = []
for doc in w_docs:
    if doc.get("past_medical_history") is not None:
        w_pmh.append(doc.get("past_medical_history"))
```

[5]:
```python
# We want to see if each patient has a history of any of the following␣
↪conditions
```

```
nlp = medspacy.load()
print(nlp.pipe_names)

Span.set_extension("icd10_code", default="")

ICD_TO_TEXT_MAP = {
    "I10": "hypertension",
    "E78.5": "hyperlipidemia",
    "M19.90": "osteoarthritis",
    "E11.9": "type ii diabetes mellitus",
    "E78.00": "hypercholesterolemia",
    "J45": "asthma",
    "I48.91": "atrial fibrillation",
    "M81. 0": "osteoporosis",
    "K21.9": "gastroesophageal reflux disease ",
    "I21.9": "myocardial infarction",
    "I25.10": "coronary artery disease",
}

# Add rules for target concept extraction
target_matcher = nlp.get_pipe("medspacy_target_matcher")
target_rules = [
    TargetRule("hypertension", category="CONDITION", attributes={"icd10_code":␣
 ↪"I10"}),
    TargetRule(
        "hyperlipidemia", category="CONDITION", attributes={"icd10_code": "E78.
 ↪5"}
    ),
    TargetRule(
        "osteoarthritis", category="CONDITION", attributes={"icd10_code": "M19.
 ↪90"}
    ),
    TargetRule(
        "osteoporosis", category="CONDITION", attributes={"icd10_code": "M81.␣
 ↪0"}
    ),
    TargetRule(
        "dyslipidemia", category="CONDITION", attributes={"icd10_code": "E78.5"}
    ),
    TargetRule(
        literal="Type II Diabetes Mellitus",
        category="CONDITION",
        attributes={"icd10_code": "E11.9"},
    ),
    TargetRule(
        literal="diabetes mellitus type 2",
        category="CONDITION",
        pattern=[
            {"LOWER": "diabetes"},
            {"LOWER": "mellitus"},
            {"LOWER": "type"},
            {"LOWER": {"IN": ["two", "ii", "2"]}},
        ],
        attributes={"icd10_code": "E11.9"},
    ),
    TargetRule(
        literal="gerd",
        category="CONDITION",
        pattern=[
            {"LOWER": "gastroesophageal"},
            {"LOWER": "reflux"},
            {"LOWER": "disease"},
        ],
        attributes={"icd10_code": "K21.9"},
    ),
    TargetRule(
        literal="GERD", category="CONDITION", attributes={"icd10_code": "K21.9"}
    ),
    TargetRule(
        literal="Type II Diabetes Mellitus",
        category="CONDITION",
        pattern=[
            {"LOWER": "type"},
            {"LOWER": {"IN": ["two", "ii", "2"]}},
            {
                "LOWER": {
                    "IN": [
                        "dm",
                        "diabetes mellitus",
                        "diabetes",
                    ]
                }
            },
        ],
        attributes={"icd10_code": "E11.9"},
    ),
    TargetRule("asthma", category="CONDITION", attributes={"icd10_code":␣
↪"J45"}),
    TargetRule(
        "atrial fibrillation",
        category="CONDITION",
        attributes={"icd10_code": "I48.91"},
    ),
    TargetRule(
        "hypercholesterolemia",
        category="CONDITION",
        attributes={"icd10_code": "E78.00"},
    ),
    TargetRule(
        "high cholesterol",
        category="CONDITION",
        pattern=[{"LOWER": {"IN": ["high", "elevated"]}}, {"LOWER":␣
↪"cholesterol"}],
        attributes={"icd10_code": "E78.00"},
    ),
    TargetRule(
        "hypertriglyceridemia", category="CONDITION", attributes={"icd10_code":␣
↪"E78.1"}
    ),
    TargetRule(
        "myocardial infarction",
        category="CONDITION",
        pattern=[
            {"LOWER": "myocardial"},
            {"LOWER": "infarction"},
        ],
        attributes={"icd10_code": "I21.9"},
    ),
    TargetRule(
        "coronary artery disease",
        category="CONDITION",
        attributes={"icd10_code": "I25.10"},
    )
]
target_matcher.add(target_rules)
```

```
['medspacy_pyrush', 'medspacy_target_matcher', 'medspacy_context']
```

[6]:
```
# Extract conditions from PMH
b_nlp_pmh = []
for doc in b_pmh:
    doc = nlp(doc)
    b_nlp_pmh.append(doc)
```

[7]:
```
w_nlp_pmh = []
for doc in w_pmh:
    doc = nlp(doc)
    w_nlp_pmh.append(doc)
```

[8]:
```
# Quick test to make sure negation detection works
# negation test
```

```
test = "patient admits to type 2 diabetes but denies any hypertension."
doc = nlp(test)
visualize_ent(doc)
for ent in doc.ents:
    print(ent._.is_negated)
```

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

False
True
```

[9]:
```
# Quick visualization of entity extraction
for doc in w_nlp_pmh[:3]:
    visualize_ent(doc)
```

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

<IPython.core.display.HTML object>

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

[10]:
```
for doc in b_nlp_pmh[:3]:
    visualize_ent(doc)
```

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

<IPython.core.display.HTML object>

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

[11]:
```
# Test entity extraction, making sure to ignore negated entities
test = b_nlp_pmh[0:2]
test.append(nlp("patient admits to type 2 diabetes but denies any hypertension.␣
 ↪he takes metformin for his type 2 diabetes."))
print(list(map(lambda x: [y for y in list(x.ents) if y._.is_negated == False],␣
 ↪test)))
print(
    list(
        map(
            lambda x: [y._.icd10_code for y in list(x.ents) if y._.is_negated␣
 ↪== False],
            test,
        )
    )
)
# De-dup
print(
    list(
        map(
            lambda x: set([y._.icd10_code for y in list(x.ents) if y._.
 ↪is_negated == False]),
            test,
        )
    )
)
```

```
[[hypertension, osteoarthritis], [high cholesterol, hypertension], [type 2
diabetes, type 2 diabetes]]
[['I10', 'M19.90'], ['E78.00', 'I10'], ['E11.9', 'E11.9']]
[{'I10', 'M19.90'}, {'E78.00', 'I10'}, {'E11.9'}]
```

[12]:
```
# Do entity extraction on the PMH section of the notes, skipping negated␣
 ↪entities. Make sure to de-duplicate the entities.
b_just_names = list(
    map(
        lambda x: set([y._.icd10_code for y in list(x.ents) if y._.is_negated␣
 ↪== False]),
        b_nlp_pmh,
    )
)

b_normalized_conditions_names = [
    element for sublist in b_just_names for element in sublist
]
w_just_names = list(
    map(
        lambda x: set([y._.icd10_code for y in list(x.ents) if y._.is_negated␣
 ↪== False]),
        w_nlp_pmh,
    )
)
w_normalized_conditions_names = [
    element for sublist in w_just_names for element in sublist
]
print(len(b_normalized_conditions_names))
print(len(w_normalized_conditions_names))
```

```
6379
6313
```

[13]:
```
# Count the instances of each word in the black and white conditions.␣
 ↪Conditions are de-duped, so if a condition appears multiple times in a␣
 ↪single participant's data, it is only counted once.
# We fix this later before doing statistical analysis.
from collections import Counter

b_word_freq = Counter(b_normalized_conditions_names)
w_word_freq = Counter(w_normalized_conditions_names)
```

[14]:
```python
b_word_freq_df = pd.DataFrame(
    b_word_freq.items(), columns=["word", "b.frequency"]
).sort_values(by="b.frequency", ascending=False)
w_word_freq_df = pd.DataFrame(
    w_word_freq.items(), columns=["word", "w.frequency"]
).sort_values(by="w.frequency", ascending=False)
```

[15]:
```python
wf_df = w_word_freq_df.merge(b_word_freq_df, how="inner", on="word")
wf_df
```

[15]:
```
      word  w.frequency  b.frequency
0      I10         3201         3245
1    E78.5         2347         2334
2   M19.90          286          265
3    E11.9          167          188
4   E78.00          112          129
5      J45           70           72
6   I25.10           69           73
7    I21.9           34           48
8    K21.9           12            9
9   M81. 0            8           11
10  I48.91            7            5
```

[16]:
```python
wf_df["w.frequency_pct"] = wf_df["w.frequency"] / wf_df["w.frequency"].sum()
wf_df["b.frequency_pct"] = wf_df["b.frequency"] / wf_df["b.frequency"].sum()
wf_df["frequency_pct_diff"] = wf_df["b.frequency_pct"] - wf_df["w.
 ↪frequency_pct"]
wf_df["frequency_pct_diff_abs"] = wf_df["frequency_pct_diff"].abs()
# Sort by largest values in absolue difference
wf_df.sort_values(by="frequency_pct_diff", ascending=False).head(25)
```

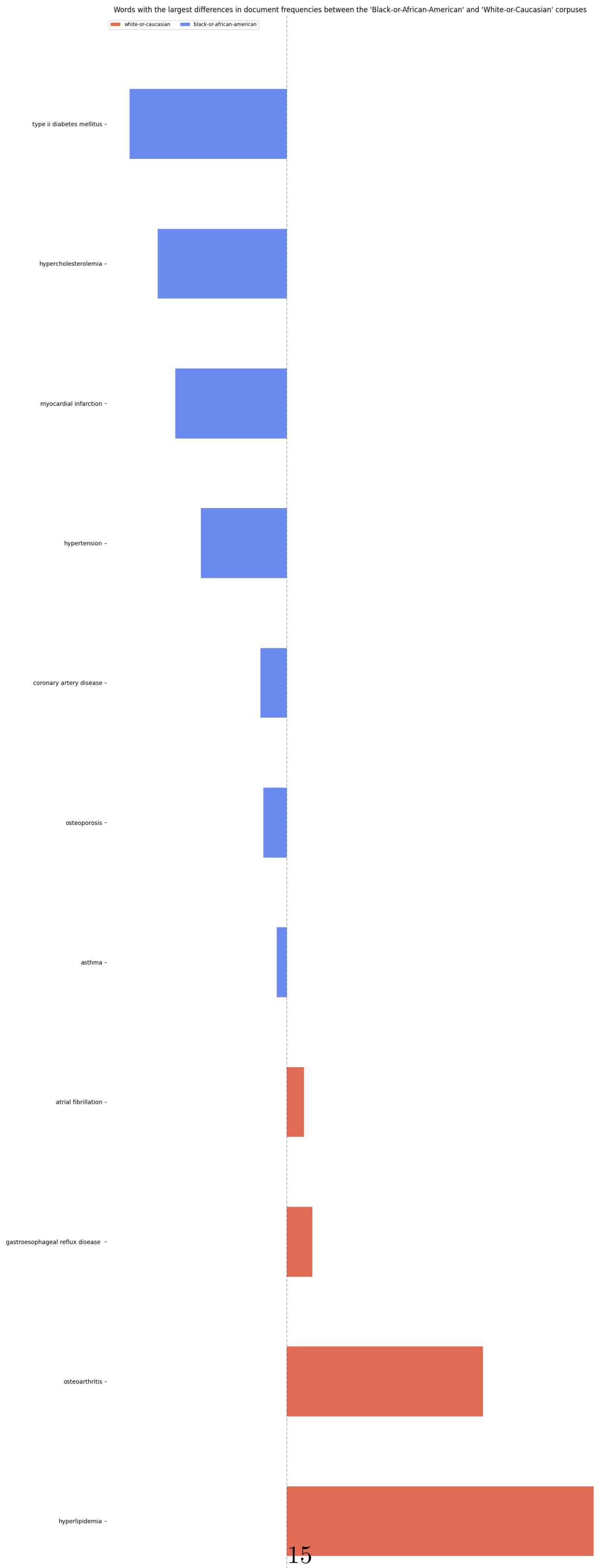
[16]:
```
      word  w.frequency  b.frequency  w.frequency_pct  b.frequency_pct  \
3    E11.9          167          188         0.026453         0.029472
4   E78.00          112          129         0.017741         0.020223
7    I21.9           34           48         0.005386         0.007525
0      I10         3201         3245         0.507049         0.508700
6   I25.10           69           73         0.010930         0.011444
9   M81. 0            8           11         0.001267         0.001724
5      J45           70           72         0.011088         0.011287
10  I48.91            7            5         0.001109         0.000784
8    K21.9           12            9         0.001901         0.001411
2   M19.90          286          265         0.045303         0.041543
1    E78.5         2347         2334         0.371773         0.365888

    frequency_pct_diff  frequency_pct_diff_abs
3             0.003018                0.003018
4             0.002481                0.002481
```

```
7           0.002139           0.002139
0           0.001651           0.001651
6           0.000514           0.000514
9           0.000457           0.000457
5           0.000199           0.000199
10         -0.000325           0.000325
8          -0.000490           0.000490
2          -0.003761           0.003761
1          -0.005884           0.005884
```

[17]:
```
# First order frequencies by magnitude of difference (absolute value), take the␣
↪top 200 words with the greatest difference,
# then re-sort by actual difference so when we plot the values will be␣
↪sequential from smallest to largest bars
most = (
    wf_df.sort_values(by="frequency_pct_diff_abs", ascending=False)
    .head(200)
    .sort_values(by="frequency_pct_diff", ascending=False)
)

chart_data = {}

# Create a map with the word as the frequency, and the magnitude vector as the␣
↪value\
# a vector of [0, n] will plot a blue bar
# a vector of [n, 0] will plot an orange bar
# a vector with a negative n [-n, 0] will plot a bar on the left
# a vector with a positive n [n, 0] will plot a bar on the right
# {"word": [-1, 0]} will plot an orange bar for "word" on the left of 0 with␣
↪length 1
# {"word": [0, 0.5]} will plot a blue bar for "word" on the right of 0 with␣
↪length 0.5
# in order to generate a good Positive Negative bar chart, we assign b freq to␣
↪the left side (negative)
# and w freq to the right side (positive)
for row in most.iterrows():
    if row[1]["w.frequency_pct"] > row[1]["b.frequency_pct"]:
        # orange bars
        chart_data[row[1]["word"]] = [
            row[1]["w.frequency_pct"] - row[1]["b.frequency_pct"],
            0,
        ]
    else:
        # blue bars
        chart_data[row[1]["word"]] = [
            0,
            -(row[1]["b.frequency_pct"] - row[1]["w.frequency_pct"]),
        ]
```

[18]:
```
# Positive Negative Bar Chart to better visualize where word frequencies␣
↪diverge between data sets
# Based on https://stackoverflow.com/a/69976552/11407943
import numpy as np
import matplotlib.pyplot as plt


category_names = ["white-or-caucasian", "black-or-african-american"]
results = chart_data


def survey(results, category_names):
    """
    Parameters
    ----------
    results : dict
        A mapping from question labels to a list of answers per category.
        It is assumed all lists contain the same number of entries and that
        it matches the length of *category_names*. The order is assumed
        to be from 'Strongly disagree' to 'Strongly aisagree'
    category_names : list of str
        The category labels.
    """

    labels = list(map(lambda i: ICD_TO_TEXT_MAP.get(i), results.keys()))
    data = np.array(list(results.values()))
    data_cum = data.cumsum(axis=1)
    middle_index = data.shape[1] // 2
    offsets = 0  # data[:, range(middle_index)].sum(axis=1) # + data[:,␣
↪middle_index]/2

    # Color Mapping
    category_colors = plt.get_cmap("coolwarm_r")(np.linspace(0.15, 0.85, data.
↪shape[1]))

    fig, ax = plt.subplots(figsize=(15, 50))

    # Plot Bars
    for i, (colname, color) in enumerate(zip(category_names, category_colors)):
        widths = data[:, i]
        starts = data_cum[:, i] - widths - offsets
        rects = ax.barh(
            labels, widths, left=starts, height=0.5, label=colname, color=color
        )
```

```python
    # Add Zero Reference Line
    ax.axvline(0, linestyle="--", color="black", alpha=0.25)

    # X Axis
    # ax.set_xlim(-0.006, 0.006)
    # ax.set_xticks(np.arange(-0.0035, 0.0035, 0.003))
    ax.xaxis.set_major_formatter(lambda x, pos: str(x))

    # Y Axis
    ax.invert_yaxis()

    # Remove spines
    ax.spines["right"].set_visible(False)
    ax.spines["top"].set_visible(False)
    ax.spines["left"].set_visible(False)

    # Ledgend
    ax.legend(
        ncol=len(category_names),
        bbox_to_anchor=(0, 0.99),
        loc="lower left",
        fontsize="small",
    )

    # Set Background Color
    fig.set_facecolor("#FFFFFF")

    return fig, ax


fig, ax = survey(results, category_names)
plt.title(
    "Words with the largest differences in document frequencies between the␣
 ↪'Black-or-African-American' and 'White-or-Caucasian' corpuses"
)
plt.show()
```


Words with the largest differences in document frequencies between the 'Black-or-African-American' and 'White-or-Caucasian' corpuses
white-or-caucasian
black-or-african-american
type ii diabetes mellitus
hypercholesterolemia
myocardial infarction
hypertension
coronary artery disease
osteoporosis
asthma
atrial fibrillation
gastroesophageal reflux disease
osteoarthritis
hyperlipidemia
-0.002
0.0
0.002
0.004

```
[19]: import scipy
      from sklearn.feature_extraction import text
      from collections import Counter
```

```
[20]: b_just_names_lower = [
          list(map(lambda x: ICD_TO_TEXT_MAP.get(x), arr)) for arr in b_just_names
      ]
      b_list_of_doc_counter = list(map(Counter, b_just_names_lower))
      # element for sublist in w_just_names for element in sublist
      w_just_names_lower = [
          list(map(lambda x: ICD_TO_TEXT_MAP.get(x), arr)) for arr in w_just_names
      ]
      w_list_of_doc_counter = list(map(Counter, w_just_names_lower))
      b_conditions_names_counter = Counter(
          [element for sublist in b_just_names_lower for element in sublist]
      )
      w_conditions_names_counter = Counter(
          [element for sublist in w_just_names_lower for element in sublist]
      )
```

```
[21]: b_conditions_names_counter
```

```
[21]: Counter({'hypertension': 3245,
               'hyperlipidemia': 2334,
               'osteoarthritis': 265,
               'type ii diabetes mellitus': 188,
               'hypercholesterolemia': 129,
               'coronary artery disease': 73,
               'asthma': 72,
               'myocardial infarction': 48,
               'osteoporosis': 11,
               'gastroesophageal reflux disease ': 9,
               'atrial fibrillation': 5})
```

```
[22]: w_conditions_names_counter
```

```
[22]: Counter({'hypertension': 3201,
               'hyperlipidemia': 2347,
               'osteoarthritis': 286,
               'type ii diabetes mellitus': 167,
               'hypercholesterolemia': 112,
               'asthma': 70,
               'coronary artery disease': 69,
               'myocardial infarction': 34,
               'gastroesophageal reflux disease ': 12,
               'osteoporosis': 8,
               'atrial fibrillation': 7})
```

[23]:
```
total_keys = list(
    set(
        list(w_conditions_names_counter.keys())
        + list(b_conditions_names_counter.keys())
    )
)
new_counts = {}
aa = []
ca = []
for k in total_keys:
    # [aa,ca]
    new_counts[k] = [
        b_conditions_names_counter.get(k, 0),
        w_conditions_names_counter.get(k, 0),
    ]
    aa.append(b_conditions_names_counter.get(k, 0))
    ca.append(w_conditions_names_counter.get(k, 0))

c_table = pd.DataFrame.from_dict(new_counts)
c_table.rename(index={0: "b.freq"}, inplace=True)
c_table.rename(index={1: "w.freq"}, inplace=True)
c_table
```

[23]:
```
        asthma  hyperlipidemia  osteoporosis  hypercholesterolemia  \
b.freq      72            2334            11                   129
w.freq      70            2347             8                   112

        myocardial infarction  gastroesophageal reflux disease   \
b.freq                     48                                9
w.freq                     34                               12

        osteoarthritis  coronary artery disease  atrial fibrillation  \
b.freq             265                       73                    5
w.freq             286                       69                    7

        type ii diabetes mellitus  hypertension
b.freq                        188          3245
w.freq                        167          3201
```

[24]:
```
class bcolors:
    HEADER = "\033[95m"
    OKBLUE = "\033[94m"
    OKCYAN = "\033[96m"
    OKGREEN = "\033[92m"
    WARNING = "\033[93m"
    FAIL = "\033[91m"
    ENDC = "\033[0m"
    BOLD = "\033[1m"
    UNDERLINE = "\033[4m"
```

[25]:
```
sig_results = []
# Chi square independence test
# https://www.dir.uniupo.it/pluginfile.php/138296/mod_resource/content/0/
 ↪22-colloc-bw.pdf
for k in list(set(total_keys)):
    # For AA [Number of instances of current word, Number of instances of all␣
 ↪other words]
    x1 = [c_table[k].iloc[0], c_table.iloc[0].sum() - c_table[k].iloc[0]]
    # For CA [Number of instances of current word, Number of instances of all␣
 ↪other words]
    y1 = [c_table[k].iloc[1], c_table.iloc[1].sum() - c_table[k].iloc[1]]
    test = scipy.stats.chi2_contingency([x1, y1])
    word = c_table[k].name
    if test.pvalue < 0.05:
        sig_results.append(word)
        print(f"{bcolors.BOLD}Condition: {k}{bcolors.ENDC}")
        print(f"    W   ^W")
        print(f"AA: {x1}")
        print(f"CA: {y1}")
        print(
            f'There {bcolors.OKGREEN}is a significant difference{bcolors.ENDC}␣
 ↪in the prevalence of the condition "{word}" between the groups with a␣
 ↪p-value of {bcolors.OKGREEN +"{:0.3f}".format(test.pvalue) + bcolors.ENDC}'
        )
        print(f"")
    else:
        print(f"{bcolors.BOLD}Condition: {k}{bcolors.ENDC}")
        print(f"    W   ^W")
        print(f"AA: {x1}")
        print(f"CA: {y1}")
        print(
            f'There was no significant difference in the prevalence of the␣
 ↪condition "{word}" between the groups with a p-value of {"{:0.3f}".
 ↪format(test.pvalue)}'
        )
if len(sig_results) == 0:
    print(f'{bcolors.BOLD}{bcolors.FAIL}No significant differences in any␣
 ↪conditions between groups found{bcolors.ENDC}')
```

```
Condition: osteoporosis
    W   ^W
```

```
AA: [11, 6368]
CA: [8, 6305]
There was no significant difference in the prevalence of the condition
"osteoporosis" between the groups with a p-value of 0.662
Condition: hypercholesterolemia
    W   ~W
AA: [129, 6250]
CA: [112, 6201]
There was no significant difference in the prevalence of the condition
"hypercholesterolemia" between the groups with a p-value of 0.338
Condition: asthma
    W   ~W
AA: [72, 6307]
CA: [70, 6243]
There was no significant difference in the prevalence of the condition "asthma"
between the groups with a p-value of 0.982
Condition: hyperlipidemia
    W   ~W
AA: [2334, 4045]
CA: [2347, 3966]
There was no significant difference in the prevalence of the condition
"hyperlipidemia" between the groups with a p-value of 0.504
Condition: myocardial infarction
    W   ~W
AA: [48, 6331]
CA: [34, 6279]
There was no significant difference in the prevalence of the condition
"myocardial infarction" between the groups with a p-value of 0.164
Condition: gastroesophageal reflux disease
    W   ~W
AA: [9, 6370]
CA: [12, 6301]
There was no significant difference in the prevalence of the condition
"gastroesophageal reflux disease " between the groups with a p-value of 0.645
Condition: osteoarthritis
    W   ~W
AA: [265, 6114]
CA: [286, 6027]
There was no significant difference in the prevalence of the condition
"osteoarthritis" between the groups with a p-value of 0.319
Condition: coronary artery disease
    W   ~W
AA: [73, 6306]
CA: [69, 6244]
There was no significant difference in the prevalence of the condition "coronary
artery disease" between the groups with a p-value of 0.849
Condition: atrial fibrillation
    W   ~W
```

```
AA: [5, 6374]
CA: [7, 6306]
There was no significant difference in the prevalence of the condition "atrial
fibrillation" between the groups with a p-value of 0.759
Condition: type ii diabetes mellitus
    W   ~W
AA: [188, 6191]
CA: [167, 6146]
There was no significant difference in the prevalence of the condition "type ii
diabetes mellitus" between the groups with a p-value of 0.328
Condition: hypertension
    W   ~W
AA: [3245, 3134]
CA: [3201, 3112]
There was no significant difference in the prevalence of the condition
"hypertension" between the groups with a p-value of 0.866
No significant differences in any conditions between groups found
```

```
[ ]:
```