

4_medication_analysis

August 1, 2023

```
[1]: import json
import os
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
import numpy as np
from drug_named_entity_recognition import find_drugs
import json5
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]: chief_complaint = "headache"
folder_location = os.path.join(
    DATA_PROCESSED_DOCUMENTS_DIR / chief_complaint / "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]
```

```

        tmp = "".join(tmp)
        content = json5.loads(tmp)
        b_docs.append(content)
    except Exception as e:
        # print(f"{file_location} Error: {e}")
        pass

folder_location = os.path.join(
    DATA_PROCESSED_DOCUMENTS_DIR / chief_complaint / "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]
                    tmp = "".join(tmp)
                    content = json5.loads(tmp)
                    w_docs.append(content)
                except Exception as e:
                    # print(f"{file_location} Error: {e}")
                    pass

```

```

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

```

4964

4953

```

[4]: b_normalized_medications = []
      for doc in b_docs:
          if doc.get("medications") is not None:
              res = []
              res = doc.get("medications").split(" ")
              try:
                  res.remove("other")

```

```

        except ValueError:
            pass
        res = find_drugs(res, is_ignore_case=True)
        b_normalized_medications.append(res)
    len(b_normalized_medications)

```

[4]: 4953

[5]: b_normalized_medications[:2]

[5]: [[], []]

```

[6]: w_normalized_medications = []
for doc in w_docs:
    if doc.get("medications") is not None:
        res = []
        res = doc.get("medications").split(" ")
        try:
            res.remove("other")
        except ValueError:
            pass
        res = find_drugs(res, is_ignore_case=True)
        w_normalized_medications.append(res)
    len(w_normalized_medications)

```

[6]: 4947

```

[7]: # For each patient, parse out the medications and normalize them. De-dup them
    ↳ so each patient has each medication listed only once.
b_just_names = list(
    map(
        lambda n: set(list(map(lambda m: m[0].get("name"), n))),
        b_normalized_medications,
    )
)
b_normalized_medications_names = [
    element for sublist in b_just_names for element in sublist
]
w_just_names = list(
    map(
        lambda n: set(list(map(lambda m: m[0].get("name"), n))),
        w_normalized_medications,
    )
)
w_normalized_medications_names = [
    element for sublist in w_just_names for element in sublist
]

```

```
b_just_names[:5]
# print(len(b_normalized_medications_names))
# print(len(w_normalized_medications_names))
```

```
[7]: [set(),
      set(),
      {'Ibuprofen'},
      {'Acetaminophen', 'Ibuprofen'},
      {'Atenolol', 'Simvastatin'}]
```

```
[8]: b_cv = CountVectorizer(analyzer="word")
b_cv_fit = b_cv.fit_transform(b_normalized_medications_names)
b_word_list = b_cv.get_feature_names_out()
b_count_list = b_cv_fit.toarray().sum(axis=0)

b_word_freq = dict(zip(b_word_list, b_count_list))

w_cv = CountVectorizer(analyzer="word")
w_cv_fit = w_cv.fit_transform(w_normalized_medications_names)
w_word_list = w_cv.get_feature_names_out()
w_count_list = w_cv_fit.toarray().sum(axis=0)

w_word_freq = dict(zip(w_word_list, w_count_list))
```

```
[9]: 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)
```

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

```
[11]: 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 absolute difference
wf_df.sort_values(by="frequency_pct_diff", ascending=False).head(10)
```

```
[11]:
```

	word	w.frequency	b.frequency	w.frequency_pct	\
5	acetaminophen	408	450	0.090186	
2	ibuprofen	507	552	0.112069	
0	lisinopril	1496	1566	0.330681	
9	metformin	72	92	0.015915	

10	hydrochlorothiazide	65	84	0.014368
7	metoprolol	102	115	0.022546
14	loratadine	12	18	0.002653
18	aspirin	8	11	0.001768
19	paracetamol	5	7	0.001105
27	alprazolam	1	1	0.000221

	b.frequency_pct	frequency_pct_diff	frequency_pct_diff_abs
5	0.096133	0.005948	0.005948
2	0.117924	0.005855	0.005855
0	0.334544	0.003863	0.003863
9	0.019654	0.003739	0.003739
10	0.017945	0.003577	0.003577
7	0.024567	0.002021	0.002021
14	0.003845	0.001193	0.001193
18	0.002350	0.000582	0.000582
19	0.001495	0.000390	0.000390
27	0.000214	-0.000007	0.000007

```
[12]: # 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"]),
        ]

```

```

[13]: # 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 agree'
    category_names : list of str
        The category labels.
    """

    labels = list(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
    )

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

# X Axis
# 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)

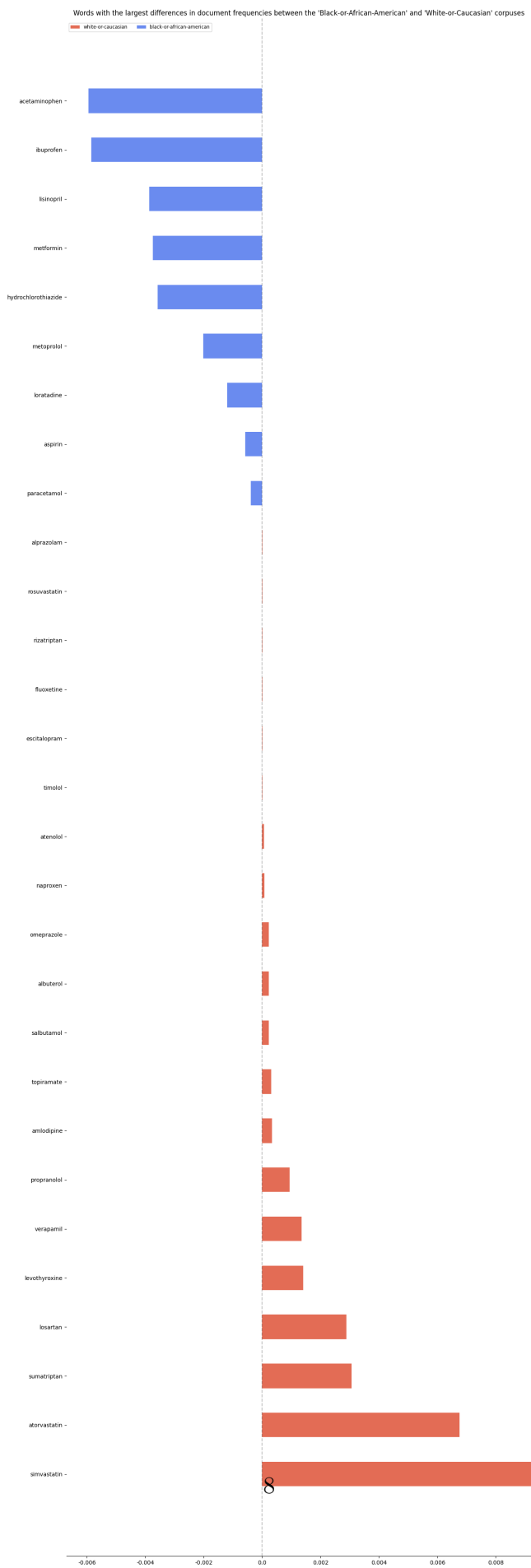
# Legend
ax.legend(
    ncol=len(category_names),
    bbox_to_anchor=(0, 0.99),
    loc="lower left",
    fontsize="small",
)

# 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()

```




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

```
[15]: b_just_names_lower = [list(map(lambda x: x.lower(), 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: x.lower(), arr)) for arr in
      ↪ w_just_names]
      w_list_of_doc_counter = list(map(Counter, w_just_names_lower))
      b_just_names_lower
      b_medications_names_counter = Counter(
          [element for sublist in b_just_names_lower for element in sublist]
      )
      w_medications_names_counter = Counter(
          [element for sublist in w_just_names_lower for element in sublist]
      )
```

```
[16]: b_medications_names_counter
```

```
[16]: Counter({'lisinopril': 1566,
              'atorvastatin': 674,
              'ibuprofen': 552,
              'amlodipine': 463,
              'acetaminophen': 450,
              'sumatriptan': 414,
              'metoprolol': 115,
              'simvastatin': 110,
              'metformin': 92,
              'hydrochlorothiazide': 84,
              'losartan': 62,
              'loratadine': 18,
              'topiramate': 12,
              'aspirin': 11,
              'naproxen': 11,
              'levothyroxine': 10,
              'atenolol': 9,
              'propranolol': 8,
              'paracetamol': 7,
              'tamsulosin': 5,
              'verapamil': 4,
              'enalapril': 2,
              'latanoprost': 2,
```

```
'cetirizine': 1,  
'memantine': 1,  
'omeprazole': 1,  
'escitalopram': 1,  
'travoprost': 1,  
'celecoxib': 1,  
'albuterol': 1,  
'salbutamol': 1,  
'fluoxetine': 1,  
'alprazolam': 1,  
'sertraline': 1,  
'meloxicam': 1,  
'timolol': 1,  
'rizatriptan': 1,  
'rosuvastatin': 1})
```

```
[17]: w_medications_names_counter
```

```
[17]: Counter({'lisinopril': 1496,  
              'atorvastatin': 682,  
              'ibuprofen': 507,  
              'amlodipine': 449,  
              'sumatriptan': 414,  
              'acetaminophen': 408,  
              'simvastatin': 148,  
              'metoprolol': 102,  
              'losartan': 73,  
              'metformin': 72,  
              'hydrochlorothiazide': 65,  
              'levothyroxine': 16,  
              'topiramate': 13,  
              'propranolol': 12,  
              'loratadine': 12,  
              'naproxen': 11,  
              'verapamil': 10,  
              'atenolol': 9,  
              'aspirin': 8,  
              'paracetamol': 5,  
              'alendronate': 2,  
              'albuterol': 2,  
              'salbutamol': 2,  
              'omeprazole': 2,  
              'furosemide': 1,  
              'rizatriptan': 1,  
              'fluoxetine': 1,  
              'epinephrine': 1,  
              'escitalopram': 1,
```

```

'alprazolam': 1,
'rosuvastatin': 1,
'timolol': 1})

```

```

[18]: total_keys = list(
    set(
        list(w_medications_names_counter.keys())
        + list(b_medications_names_counter.keys())
    )
)
new_counts = {}
aa = []
ca = []
for k in total_keys:
    # [aa,ca]
    new_counts[k] = [
        b_medications_names_counter.get(k, 0),
        w_medications_names_counter.get(k, 0),
    ]
    aa.append(b_medications_names_counter.get(k, 0))
    ca.append(w_medications_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

```

```

[18]:      levothyroxine  memantine  metformin  amlodipine  acetaminophen  \
b.freq           10           1           92           463           450
w.freq           16           0           72           449           408

      fluoxetine  sumatriptan  aspirin  propranolol  celecoxib  ...  \
b.freq           1           414          11           8           1  ...
w.freq           1           414           8           12           0  ...

      sertraline  latanoprost  tamsulosin  rosuvastatin  furosemide  \
b.freq           1           2           5           1           0
w.freq           0           0           0           1           1

      losartan  rizatriptan  enalapril  escitalopram  alendronate
b.freq         62           1           2           1           0
w.freq         73           1           0           1           2

[2 rows x 41 columns]

```

```

[19]: 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"

```

```

[20]: 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.OKGREEN}{bcolors.BOLD}Medication: {k}{bcolors.ENDC}")
        print(f"AA: {x1}")
        print(f"CA: {y1}")
        print(
            f'There {bcolors.OKGREEN}is a significant difference{bcolors.ENDC}
↳in the use of medication "{word}" between the groups with a p-value of
↳{bcolors.OKGREEN + "{:0.3f}".format(test.pvalue) + bcolors.ENDC}'
        )
        print(f"")
    # else:
    #     print(f"{bcolors.FAIL}{bcolors.BOLD}Medication: {k}{bcolors.ENDC}")
    #     print(f"AA: {x1}")
    #     print(f"CA: {y1}")
    #     print(
    #         f'There {bcolors.FAIL}is no significant difference{bcolors.ENDC}
↳in the use of medication "{word}" between the groups with a p-value of
↳{bcolors.FAIL + "{:0.3f}".format(test.pvalue) + bcolors.ENDC}'
    #     )
    #     print(f"")
if len(sig_results) == 0:
    print(
        f"{bcolors.BOLD}{bcolors.FAIL}No significant differences in any
↳conditions between groups found{bcolors.ENDC}"
    )

```

)

Medication: simvastatin

AA: [110, 4586]

CA: [148, 4380]

There is a significant difference in the use of medication
"simvastatin" between the groups with a p-value of 0.008