MIMIC NLP

By David Bush

1. Prepare NoteEvents/Diagnoses Datasets

Load the Patients, NoteEvents, and Diagnoses_ICD tables from Google Drive and store each in a Pandas dataframe

```
# Set folder path for MIMIC dataset
MIMIC_path = '/content/drive/My Drive/MIMIC Dataset'
os.chdir(MIMIC_path)
```

```
# Load the Patients table
patients_df = pd.read_csv('PATIENTS.csv')

patients_df.head()
```

```
# Load the NoteEvents table
noteevents_df = pd.read_csv('NOTEEVENTS.csv.gz', low_memory=False).set_index('ROW_ID')
```

```
# Load Diagnoses_ICD table
diagnoses_icd_df = pd.read_csv('DIAGNOSES_ICD.csv.gz').set_index('ROW_ID')
```

Filter NoteEvents by Discharge summary and Diagnoses_ICD by ICD_9 code

```
# Filter NoteEvents by Discharge summary and extract text
discharge_df = noteevents_df.loc[noteevents_df['CATEGORY'] == 'Discharge summary',
                                 ['SUBJECT_ID', 'HADM_ID', 'TEXT']]
# print(discharge_df.head())

# Filter Diagnoses_ICD by disease codes
disease_codes = ['41401', '42731', '4280', '5849', '51881']
disease_df = diagnoses_icd_df[diagnoses_icd_df['ICD9_CODE'].isin(disease_codes)].copy()

# Map the ICD_9 codes to their related disease
disease_types = {
    '41401': 'coronary atherosclerosis',
    '42731': 'atrial fibrillation',
    '4280': 'congestive heart failure',
    '5849': 'acute kidney failure',
    '51881': 'acute respiratory failure'
}
disease_df['DISEASE_TYPE'] = disease_df['ICD9_CODE'].map(disease_types)
```

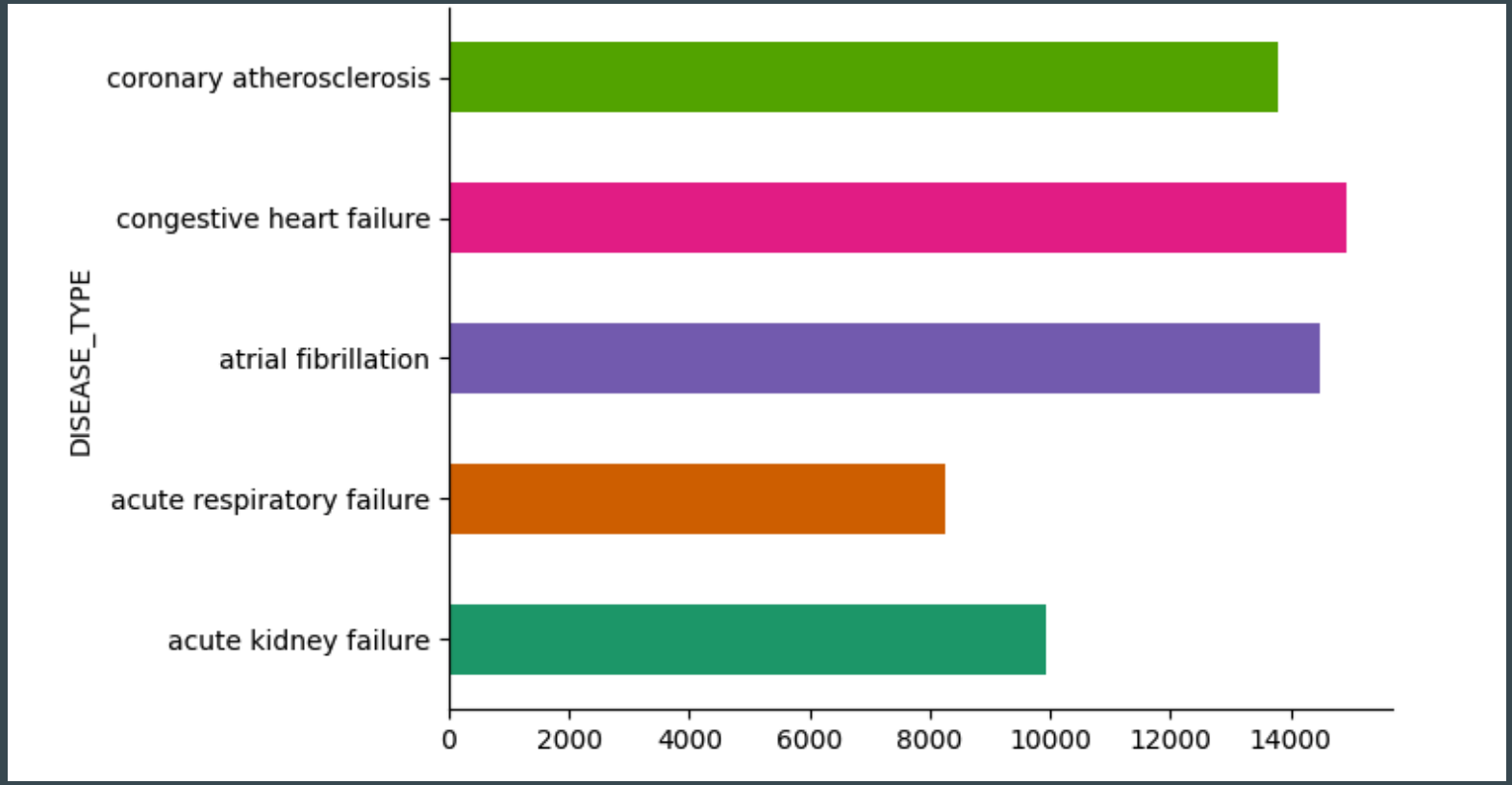
Join the discharge_df and disease_df and save the merged results into a disease_summary.csv file.

```
# Merge discharge_df and disease_df on subject_id and hadm_id
discharge_disease_df = pd.merge(discharge_df, disease_df, on=['SUBJECT_ID', 'HADM_ID'], how='inner')
print(discharge_disease_df.head())

# Save as subset so we don't have to regenerate
discharge_disease_df.to_csv("disease_summary1.csv.gz")
```

	SUBJECT_ID	HADM_ID	TEXT	SEQ_NUM	ICD9_CODE	DISEASE_TYPE
0	22532	167853.0	Admission Date: [**2151-7-16**] Dischar...	3.0	42731	atrial fibrillation
1	22532	167853.0	Admission Date: [**2151-7-16**] Dischar...	3.0	42731	atrial fibrillation
2	13702	107527.0	Admission Date: [**2118-6-2**] Discharg...	3.0	51881	acute respiratory failure
3	13702	196489.0	Admission Date: [**2124-7-21**] ...	2.0	5849	acute kidney failure
4	13702	196489.0	Admission Date: [**2124-7-21**] ...	19.0	41401	coronary atherosclerosis

Visualize the number of Discharge Notes by Disease Type



2. SpaCy Extraction (Kidney Failure)

Load SpaCy and filter summary_df by kidney failure ICD_9 code

```
# Load the SpaCy model for clinical notes
nlp = spacy.load("en_core_web_sm")

# Load disease summary text
summary_df = pd.read_csv('disease_summary1.csv.gz', index_col=0)
summary_df
```

```
# Filter summary_df with ICD9_CODE '5849' for Acute Kidney Failure
kidney_failure_df = summary_df[summary_df['ICD9_CODE'] == 5849]
kidney_failure_df
```

	SUBJECT_ID	HADM_ID	TEXT	SEQ_NUM	ICD9_CODE	DISEASE_TYPE
3	13702	196489.0	Admission Date: [**2124-7-21**] ...	2.0	5849	acute kidney failure
10	28063	121936.0	Admission Date: [**2125-2-9**] D...	5.0	5849	acute kidney failure
15	1136	139574.0	Admission Date: [**2192-4-19**] ...	1.0	5849	acute kidney failure
22	79900	120644.0	Admission Date: [**2194-7-18**] ...	5.0	5849	acute kidney failure
31	9805	177212.0	Admission Date: [**2131-6-28**] ...	5.0	5849	acute kidney failure

Functions to extract tokens and entities from text using SpaCy library

```python
# This function tokenizes and cleans the text using SpaCy
def extract_tokenize_spacy(text):
  doc = nlp(text)

  # Tokenize and remove punctuation
  tokens = [token.text.lower() for token in doc if (token.is_alpha or token.is_digit) and not token.is_punct]
  # print('Tokens without punct: ', tokens)

  # Remove stop words such as 'the, and, is, of, etc'
  tokens = [token for token in tokens if token not in STOP_WORDS]
  # print('Tokens without stop words: ', tokens)

  processed_text = ' '.join(tokens)
  return processed_text

# Function to extract entities from text using SpaCy
def extract_entities_spacy(text):
    doc = nlp(text)
    for ent in doc.ents:
      print(ent.text, ent.start_char, ent.end_char, ent.label_)
    print("**********************************************************")
    return doc
```

Apply tokenization to kidney_df and show resulting dataframe

```
# Apply tokenize function to kidney_subset_df
kidney_subset_df['PROCESSED_TEXT'] = kidney_subset_df['TEXT'].apply(extract_tokenize_spacy)
kidney_subset_df
```

SUBJECT_ID	HADM_ID	TEXT	ICD9_CODE	DISEASE_TYPE	PROCESSED_TEXT
13702	196489.0	Admission Date: [**2124-7-21**] ...	5849	acute kidney failure	admission date 2124 7 21 discharge date 2124 8...
28063	121936.0	Admission Date: [**2125-2-9**] D...	5849	acute kidney failure	admission date 2125 2 9 discharge date 2125 2 ...
1136	139574.0	Admission Date: [**2192-4-19**] ...	5849	acute kidney failure	admission date 2192 4 19 discharge date 2192 5...
79900	120644.0	Admission Date: [**2194-7-18**] ...	5849	acute kidney failure	admission date 2194 7 18 discharge date 2194 7...
9805	177212.0	Admission Date: [**2131-6-28**] ...	5849	acute kidney failure	admission date 2131 6 28 discharge date 2131 7...

Extract entities from kidney_df and show the results

```
extract_entities_spacy(kidney_subset_df.iloc[0]['PROCESSED_TEXT'])
```

```
admission 0 9 ENTITY
discharge 25 34 ENTITY
service 50 57 ENTITY
medicine allergies 58 76 ENTITY
amlodipine lf 77 90 ENTITY
chief complaint 95 110 ENTITY
copd 111 115 ENTITY
breath 116 122 ENTITY
surgical invasive procedure 129 156 ENTITY
arterial line placement picc line 168 201
history 239 246 ENTITY
illness 255 262 ENTITY
yo f chf copd 266 279 ENTITY
oxygen baseline 284 299 ENTITY
tracheobronchomalacia stent 300 327 ENTITY
acute dyspnea 337 350 ENTITY
```

admission ENTITY date 2124 7 21 discharge ENTITY date 2124 8 18

ENTITY intubation arterial line placement picc line ENTITY placement

presents acute dyspnea ENTITY days ENTITY lethargy morning

tripoding ENTITY given nebulizer ENTITY brought er according pat

orthopnea pnd dysuria ENTITY diarrhea ENTITY confusion ENTITY

consolidation report ekg ENTITY unremarkable ENTITY laboratory

sulfate 2 g ENTITY iv azithromycin ENTITY 500 mg levofloxacin

nrb nebulizers 2 hours ENTITY time patient ENTITY agitated hypox

```
# Use displacy to visualize entities
displacy.render(nlp(kidney_subset_df.iloc[0]['PROCESSED_TEXT']), style="ent", jupyter=True)
```

3. SciSpacy Extraction (Respiratory Failure)

Load SciSpacy NLP and filter by respiratory failure ICD_9 code

```
import en_core_sci_md
# Load SciSpacy en_core_sci_md nlp
nlp = en_core_sci_md.load()
```

```
# Filter summary_df with ICD9_CODE '51881' for Acute Respiratory Failure
respiratory_failure_df = summary_df[summary_df['ICD9_CODE'] == 51881]
respiratory_failure_df
```

SUBJECT_ID	HADM_ID	TEXT	ICD9_CODE	DISEASE_TYPE
13702	107527.0	Admission Date: [**2118-6-2**] Discharg...	51881	acute respiratory failure
20646	134727.0	Admission Date: [**2112-12-8**] ...	51881	acute respiratory failure
26601	155131.0	Admission Date: [**2131-12-23**] ...	51881	acute respiratory failure
19183	145164.0	Admission Date: [**2141-7-9**] D...	51881	acute respiratory failure
26175	156154.0	Admission Date: [**2112-4-22**] ...	51881	acute respiratory failure

Functions to extract tokens and entities using SciSpacy library

```python
# This function tokenizes and cleans the text using SciSpacy
def extract_tokenize_sci(text):
  doc = nlp(text)

  # Tokenize and remove punctuation
  tokens = [token.text.lower() for token in doc if (token.is_alpha or token.is_digit) and not token.is_punct]
  # print('Tokens without punct: ', tokens)

  # Remove stop words such as 'the, and, is, of, etc'
  tokens = [token for token in tokens if token not in STOP_WORDS]
  # print('Tokens without stop words: ', tokens)

  processed_text = ' '.join(tokens)
  return processed_text

# Function to extract entities from text using SciSpacy
def extract_entities_sci(text):
    doc = nlp(text)
    for ent in doc.ents:
      print(ent.text, ent.start_char, ent.end_char, ent.label_)
    print("**********************************************************")
    return doc
```

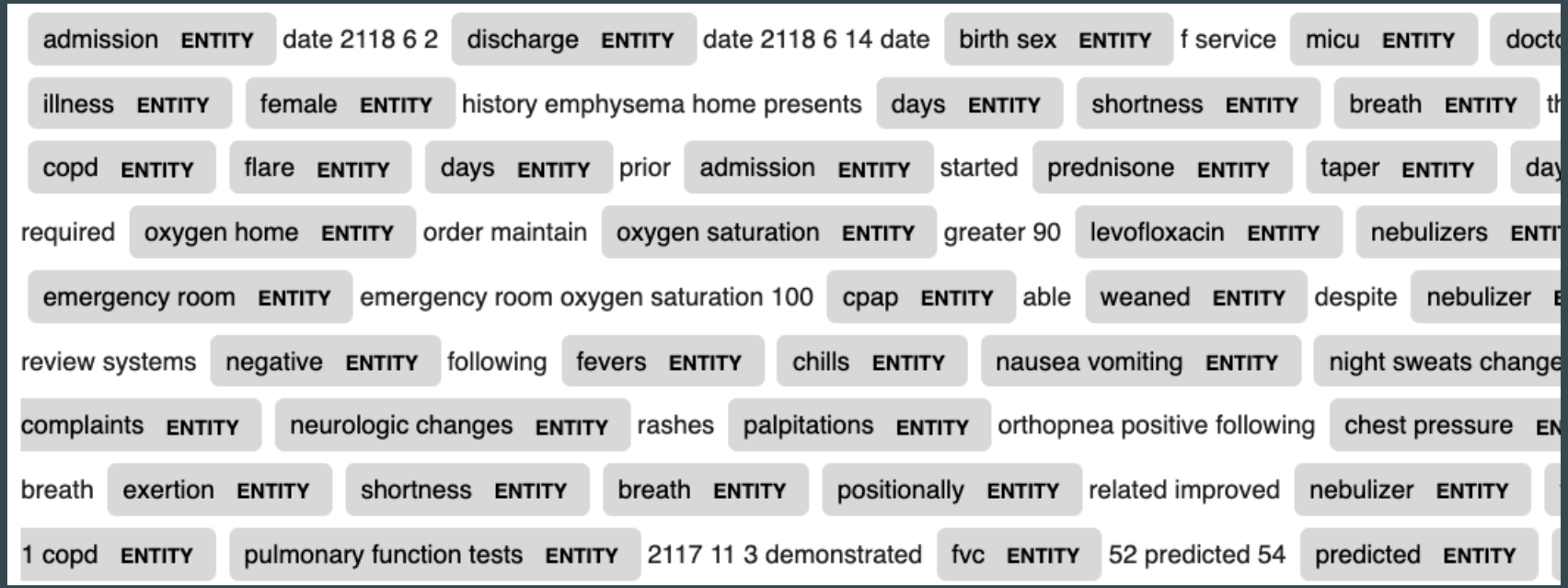
Apply tokenization to respiratory_df and show results

```
# Apply tokenize function to respiratory_df
respiratory_failure_df['PROCESSED_TEXT'] = respiratory_failure_df['TEXT'].apply(extract_tokenize_sci)
respiratory_failure_df
```

SUBJECT_ID	HADM_ID	TEXT	ICD9_CODE	DISEASE_TYPE	PROCESSED_TEXT
13702	107527.0	Admission Date: [**2118-6-2**] Discharg...	51881	acute respiratory failure	admission date 2118 6 2 discharge date 2118 6 ...
20646	134727.0	Admission Date: [**2112-12-8**] ...	51881	acute respiratory failure	admission date 2112 12 8 discharge date 2112 1...
26601	155131.0	Admission Date: [**2131-12-23**] ...	51881	acute respiratory failure	admission date 2131 12 23 discharge date 2131 ...
19183	145164.0	Admission Date: [**2141-7-9**] D...	51881	acute respiratory failure	admission date 2141 7 9 discharge date 2141 7 ...
26175	156154.0	Admission Date: [**2112-4-22**] ...	51881	acute respiratory failure	admission date 2112 4 22 discharge date 2112 5...

Use SciSpacy NLP to extract entities and show results with displacy

```
# Use displacy to visualize entities
displacy.render(nlp(respiratory_subset_df.iloc[0]['PROCESSED_TEXT']), style="ent", jupyter=True)
```



Use medically trained NLP to extract specific entities

```
import en_ner_bc5cdr_md
# Use medical trained nlp to extract entities
nlp = en_ner_bc5cdr_md.load()
doc = nlp(respiratory_subset_df.iloc[0]['PROCESSED_TEXT'])
displacy.render(doc, style="ent", jupyter=True)
```

admission date 2118 6 2 discharge date 2118 6 14 date birth sex f service micu doctor medicine history present illness female history

days shortness breath DISEASE thought primary care doctor copd flare days prior admission started prednisone CHEMICAL tape

CHEMICAL home order maintain oxygen CHEMICAL saturation greater 90 levofloxacin CHEMICAL nebulizers getting better pre

room oxygen CHEMICAL saturation 100 cpap able weaned despite nebulizer treatment 125 mg iv review systems negative following

night sweats DISEASE change weight gastrointestinal complaints neurologic changes DISEASE rashes palpitations orthopnea

pressure occasionally shortness breath DISEASE exertion shortness breath DISEASE positionally related improved nebulizer tre

pulmonary function tests 2117 11 3 demonstrated fvc 52 predicted 54 predicted mmf 23 predicted ratio 67 predicted improve bronchodil

bronchodilator treatment consistent known reversible air flow obstruction addition underlying restrictive ventilatory defect patient home

steroid CHEMICAL taper intubated past 2 lacunar cva mri head 2114 11 4 demonstrates mild degree multiple small foci high signal

4. Word2Vec Respiratory Failure

First create corpus of entities for Word2Vec to use as input

```
# Create full corpus with all respiratory dataset
respiratory_corpus = []
for row in range(0, len(respiratory_failure_df)):
  ent_str = []
  ents = nlp(respiratory_failure_df.iloc[row]['PROCESSED_TEXT']).ents
  for i in range(0, len(ents)):
    ent_str.append(ents[i].text)
  # print(ent_str)
  respiratory_corpus.append(list(ent_str))
```

```
print(len(respiratory_corpus))
print(respiratory_corpus[:10])
```

```
8271
[['emphysema', 'shortness breath', 'prednisone', 'oxygen', 'oxygen',
```

Find the max, min, and median word count of the corpus

```
# Find max, min, median word counts in the respiratory_corpus

# Flatten the corpus
flat_corpus = [word for sublist in respiratory_corpus for word in sublist]

# Calculate word counts
word_counts = [len(sublist) for sublist in respiratory_corpus]

# Calculate max, min, median word counts
max_count = max(word_counts)
min_count = min(word_counts)
median_count = sorted(word_counts)[len(word_counts) // 2]

# Display the results
print(f"Max Word Count: {max_count}")
print(f"Min Word Count: {min_count}")
print(f"Median Word Count: {median_count}")
```

```
Max Word Count: 370
Min Word Count: 0
Median Word Count: 84
```

Create the Word2Vec model using respiratory corpus

```
from gensim.models import Word2Vec
# Create Word2Vec model using respiratory entities
# Adjust the min_count, window, vector_size as needed
model_respiratory = Word2Vec(respiratory_corpus, min_count=40, window=10, vector_size=200)
```

```
model_respiratory.wv.key_to_index.keys()
```

```
dict_keys(['pain', 'pneumonia', 'allergies', 'heparin', 'lasix', 'oxygen', 'aspirin', 'vancomycin',
'hypertension', 'respiratory failure', 'creatinine', 'hypotension', 'prednisone', 'albuterol',
'infection', 'atrial fibrillation', 'metoprolol', 'bleeding', 'respiratory distress', 'sepsis',
'anemia', 'stitle', 'copd', 'alcohol', 'edema', 'lisinopril', 'htn', 'levofloxacin', 'pulmonary
edema', 'fever', 'pleural effusion', 'shortness breath', 'chest pain', 'constipation', 'steroids',
'seizure', 'pcp', 'acetaminophen', 'mitral regurgitation', 'pleural effusions', 'hypotensive',
'cough', 'qid', 'amiodarone', 'morphine', 'hypoxia', 'coumadin', 'atelectasis', 'hematoma', 'qhs',
'docusate sodium', 'urea', 'sodium', 'furosemide', 'abdominal pain', 'warfarin', 'metoprolol
tartrate', 'dyspnea', 'ativan', 'lactate', 'flagyl', 'iron', 'ceftriaxone', 'diarrhea',
'pantoprazole', 'seizures', 'shock', 'tylenol', 'tte', 'stenosis', 'ipratropium bromide',
'diltiazem', 'oxycodone', 'ischemia', 'levophed', 'levothyroxine', 'hemorrhage', 'stroke',
'depression', 'fentanyl', 'anxiety', 'atorvastatin', 'acute renal failure', 'na', 'tachycardia',
'glucose', 'dopamine', 'fracture', 'senna', 'simvastatin', 'encephalopathy', 'diabetes',
'bisacodyl', 'pancreatitis', 'rash', 'infectious disease', 'lactulose', 'ascites', 'propofol',
'pneumothorax', 'firstname', 'omeprazole', 'trauma', 'dvt', 'bactrim', 'lorazepam', 'multivitamin',
```

Use the model to look at similar words to “shortness of breath”

```
# Find words that are similar to 'shortness breath' using our model
model_respiratory.wv.similar_by_word('shortness breath')
```

```
[('dyspnea', 0.807198047637939),
 ('shortness breath cough', 0.796778738498687),
 ('cough', 0.76764297485351),
 ('orthopnea', 0.760616362094879),
 ('exacerbations', 0.75909322500228),
 ('acute shortness breath', 0.75304168462753),
 ('weight gain', 0.7474365234375),
 ('shortness breath chest pain', 0.696149408817291),
 ('dry cough', 0.690732359886169),
 ('respiratory infection', 0.683655083179473)]
```

Use the model to look at similar words to “cough”

```
# Find words that are similar to 'cough' using our model
model_respiratory.wv.similar_by_word('cough')
```

```
[('shortness breath cough', 0.7986708283424377),
 ('shortness breath', 0.7676429748535156),
 ('chills', 0.7535938024520874),
 ('dry cough', 0.748060882091522),
 ('sweats', 0.7080130577087402),
 ('respiratory infection', 0.703165829181671),
 ('fever chills', 0.6948913931846619),
 ('bronchitis', 0.6920480132102966),
 ('palpitations', 0.6915125846862793),
 ('ceftriaxone azithromycin', 0.6875654458999634)]
```

Find similarity between two words using our respiratory model

```
# Shows the similarity between shortness breath and pneumonia
model_respiratory.wv.similarity('shortness breath', 'pneumonia')
```

```
0.20784101
```

```
# Shows the similarity between cough and shortness breath cough
model_respiratory.wv.similarity('cough', 'shortness breath cough')
```

```
0.79867095
```

```
# Shows the similarity between bleeding and blood loss
model_respiratory.wv.similarity('bleeding', 'blood loss')
```

```
0.67575264
```

5. Plot Respiratory Entities t-SNE

Create the t-SNE plots (1/2)

```
# Create and plot words using TSNE
def tsne_plot(model,words, preTrained=False):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    for word in words:
      if preTrained:
          tokens.append(model[word])
      else:
          tokens.append(model.wv[word])
      labels.append(word)

    tokens = np.array(tokens)
    tsne_model = TSNE(perplexity=30, early_exaggeration=12, n_components=2,
                      init='pca', n_iter=1000, random_state=23)
    new_values = tsne_model.fit_transform(tokens)
```

Create the t-SNE plots (2/2)

```
x = []
y = []
for value in new_values:
    x.append(value[0])
    y.append(value[1])

plt.figure(figsize=(16, 16))
for i in range(len(x)):
    plt.scatter(x[i],y[i])
    plt.annotate(labels[i],
                 xy=(x[i], y[i]),
                 xytext=(5, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom')
plt.show()
```

Generate random sample of words from our model's vocab

```
import random
# Function to retrieve random sample of words used
def random_word_sample(word_list, sample_size):
  random_sample = random.sample(word_list, sample_size)
  return random_sample
```

```
# Create vocab list from model's keys
vocabs = model_respiratory.wv.key_to_index.keys()
new_v = np.array(list(vocabs))
len(new_v)
```

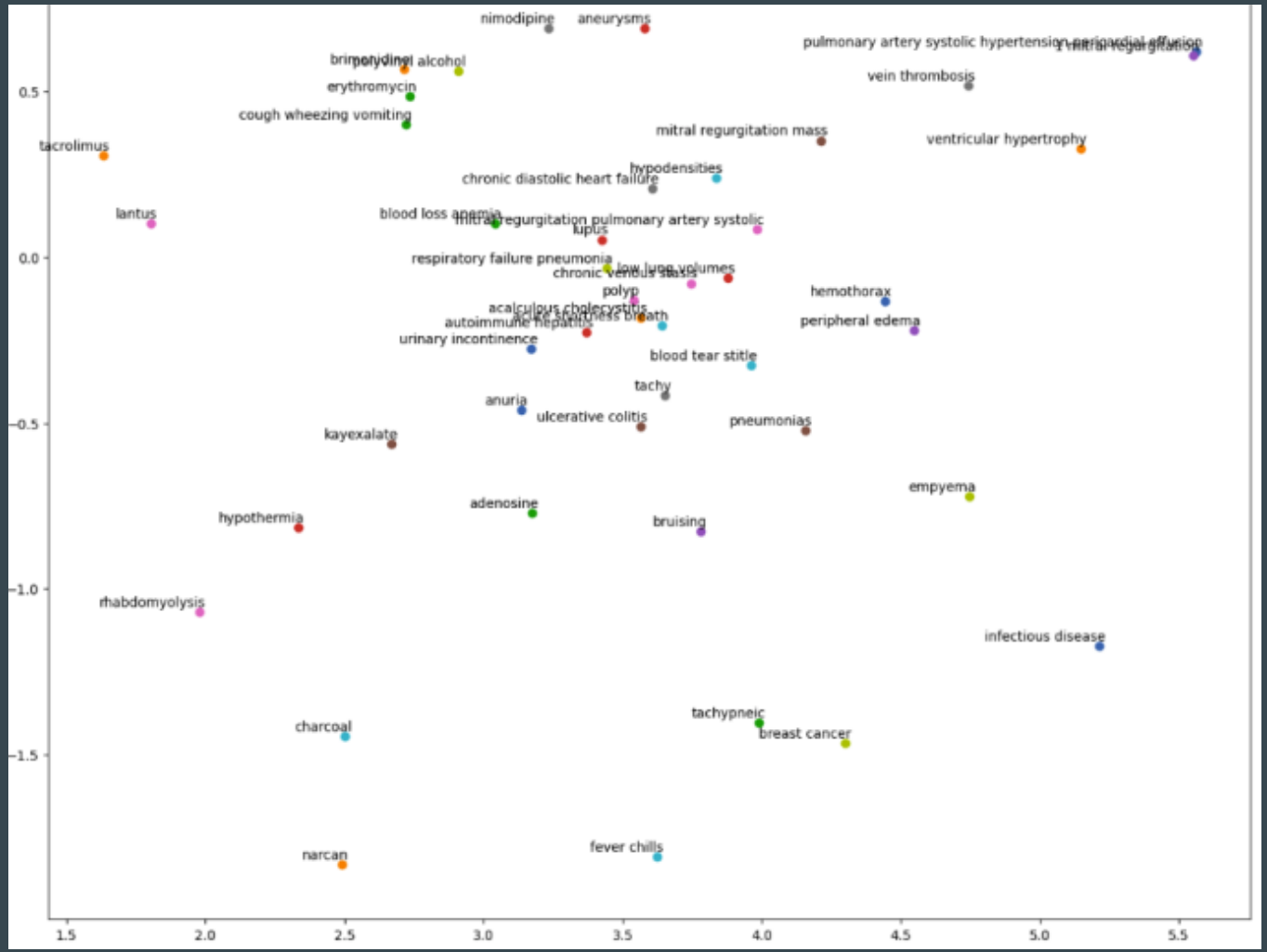
```
1478
```

```
print(random_word_sample(list(vocabs), 50))
```

```
['epinephrine', 'folic acid', 'difficile toxin', 'vancomycin', 'levetiracetam', 'cellulitis', 'rifampin', 'anasarca',
```

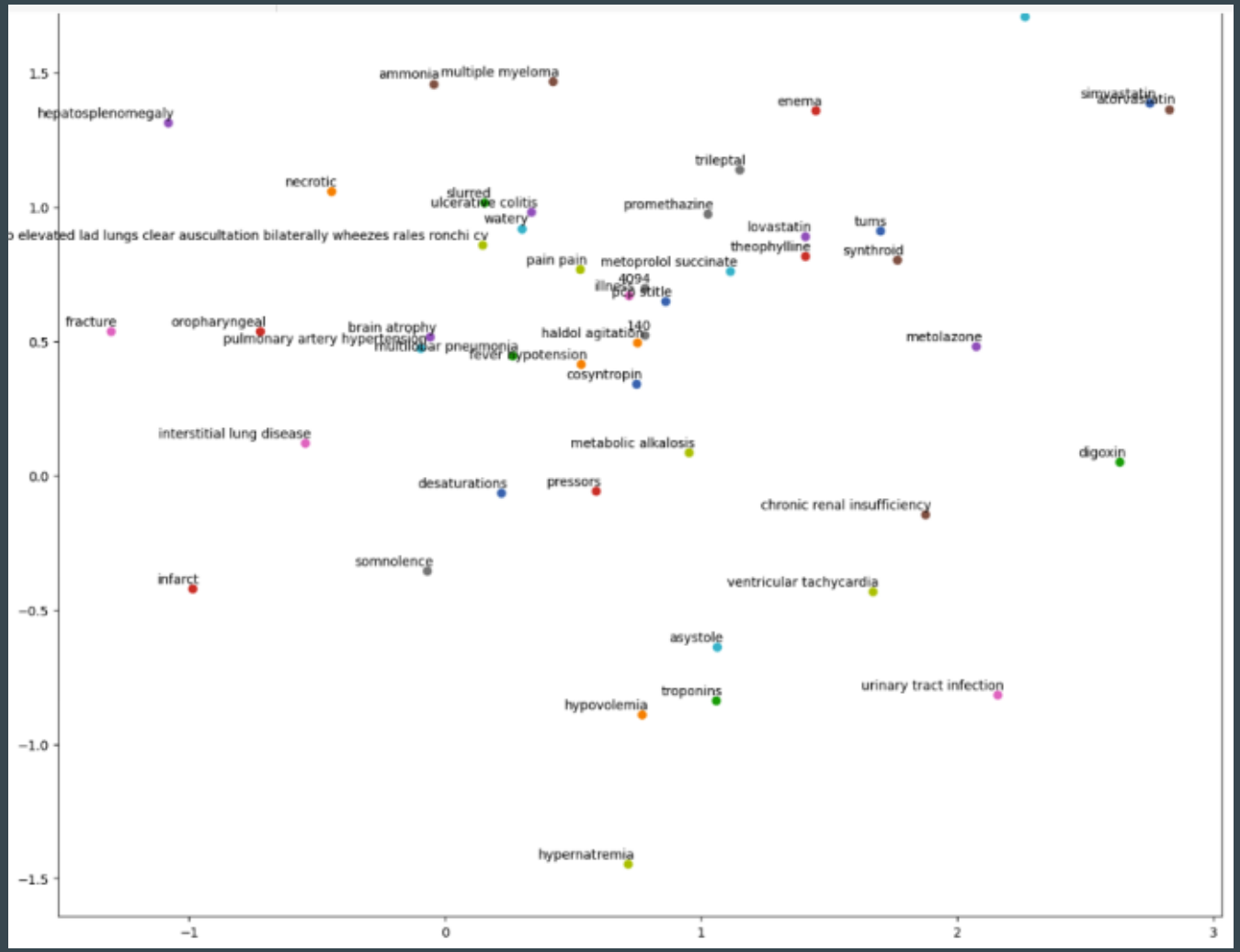
Take a random sample of 50 words from our model's vocabulary and plot the results in t-SNE graphs (Next slide)

```
# Plot random sample of 50 words from model
tsne_plot(model_respiratory, np.array(random_word_sample(list(vocabs), 50)))
```

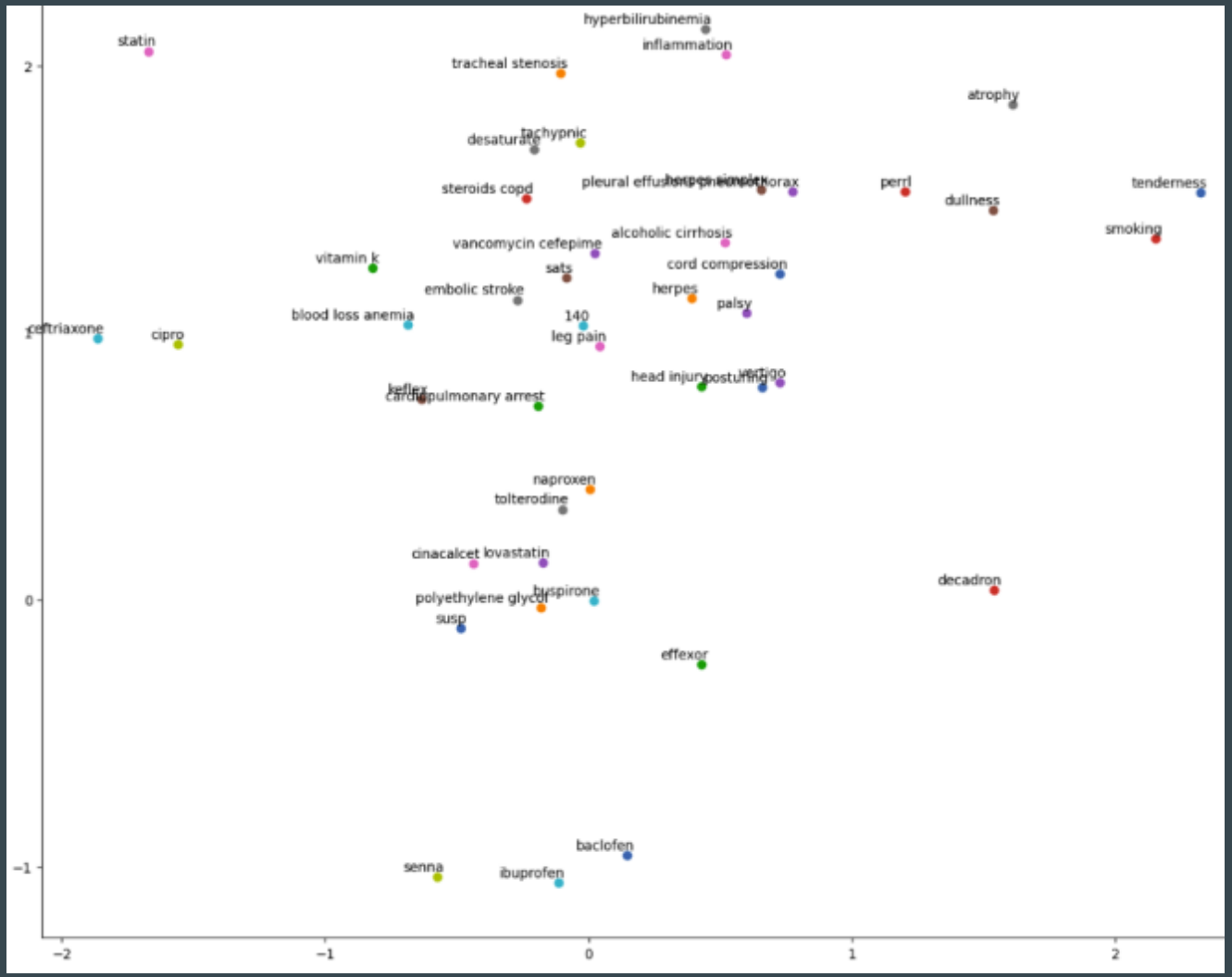

nimodipine
aneurysms
pulmonary artery systolic hypertension
pericardial effusion
mitral regurgitation
vein thrombosis
bronchodilator
polydipsia alcohol
erythromycin
cough wheezing vomiting
mitral regurgitation mass
ventricular hypertrophy
tacrolimus
hypodensities
chronic diastolic heart failure
lantus
blood loss anemia
mitral regurgitation pulmonary artery systolic
lupus
respiratory failure pneumonia
low lung volumes
polyp
hemothorax
acalculous cholecystitis
peripheral edema
urinary incontinence
blood tear stitle
tachy
anuria
ulcerative colitis
pneumonias
kayexalate
empyema
adenosine
hypothermia
bruising
rhabdomyolysis
infectious disease
tachypneic
charcoal
breast cancer
narcan
fever chills
0.5
0.0
−0.5
−1.0
−1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

Result (1/4)

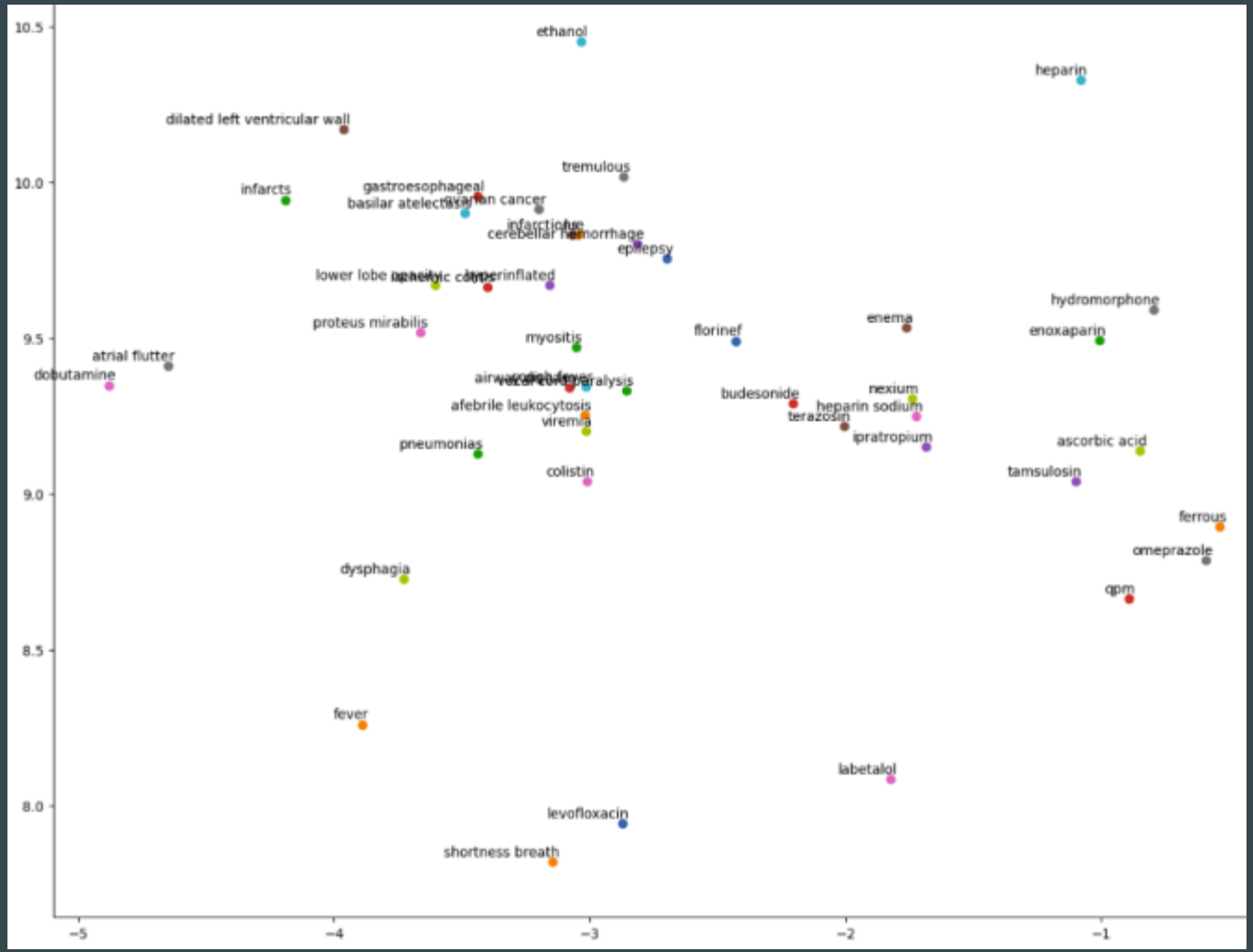
Result (2/4)


ammonia
multiple myeloma
hepatosplenomegaly
enema
simvastatin
atorvastatin
trileptal
necrotic
slurred
ulcerative colitis
watery
promethazine
lovastatin
tums
elevated lad lungs clear auscultation bilaterally wheezes rales ronchi cv
theophylline
synthroid
pain pain
metoprolol succinate
4094
fracture
oropharyngeal
brain atrophy
pulmonary artery hypertension
haldol agitation
140
metolazone
fever
hypotension
cosyntropin
interstitial lung disease
metabolic alkalosis
digoxin
desaturations
pressors
chronic renal insufficiency
somnolence
infarct
ventricular tachycardia
asystole
urinary tract infection
troponins
hypovolemia
hypernatremia
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1
0
1
2
3

Result (3/4)


hyperbilirubinemia
statin
inflammation
tracheal stenosis
atrophy
tachypnic
desaturats
steroids copd
pleural effusion
perrl
tenderness
dullness
smoking
alcoholic cirrhosis
vancomycin cefepime
vitamin k
sats
cord compression
embolic stroke
herpes
palsy
blood loss anemia
140
ceftriaxone
cipro
leg pain
head injury
vertigo
keflex
cardiopulmonary arrest
naproxen
tolterodine
cinacalcet
lovastatin
decadron
buspirone
polyethylene glycol
susp
effexor
baclofen
senna
ibuprofen
2
1
0
−1
−2
−1
0
1
2

Result (4/4)


ethanol
heparin
dilated left ventricular wall
tremulous
infarcts
gastroesophageal
basilar atelectasis
cerebellar hemorrhage
epilepsy
lower lobe
hyperinflated
hydromorphone
proteus mirabilis
myositis
florinef
enema
enoxaparin
atrial flutter
dobutamine
budesonide
nexium
afebrile leukocytosis
heparin sodium
terazosin
viremia
ipratropium
ascorbic acid
pneumonias
colistin
tamsulosin
ferrous
omeprazole
dysphagia
qpm
fever
labetalol
levofloxacin
shortness breath
10.5
10.0
9.5
9.0
8.5
8.0
−5
−4
−3
−2
−1