MIMIC NLP

Greg Gipson

Background

This tutorial covers the analysis of medical notes using Spacy, SciSpacy and ClinicalBERT.

The objective of the analysis is to generate extracted entities using Spacy and SciSpacy, word2vec, and tSNE plots.

Additionally this tutorial will use ClinicalBert to generate the same analysis to determine whether there is increased efficacy and information relative to the other two processes.

The first step is to download and extract the MIMIC-III files from https://physionet.org/. The NOTEEVENTS file is then uploaded to Google drive for loading into the Jupyter notebook while other files are stored locally and uploaded as needed.

Data Loading and Prep

These steps are designed to create a reusable file for analysis so as to minimize the processing load and increase analysis speed.

```
import pandas as pd
import numpy as np

#connect to google drive to read in NOTEEVENTS file
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
%cd drive/MyDrive
noteevents_df = pd.read_csv('NOTEEVENTS.csv', low_memory=False)#.set_index('ROW_ID’)

# keep discharge summary text and create subject_admission_id key for merging
dischargetext_df = noteevents_df.loc[noteevents_df['CATEGORY'] == 'Discharge summary',
                                     ['SUBJECT_ID', 'HADM_ID', 'TEXT']]
dischargetext_df['subj_hadm'] = list(zip(dischargetext_df['SUBJECT_ID'].astype(int),
                                         dischargetext_df['HADM_ID'].astype(int)))

#upload the DIAGNOSES_ICD in a dataframe
from google.colab import files
uploaded = files.upload()
diagnoses_icd_df = pd.read_csv('DIAGNOSES_ICD.csv').set_index('ROW_ID’)

#keeping with my SQL assignment, pulling data for Wolf-Parkinson White (WPW) Syndrome
disease_list = ['4267']

disease_df = diagnoses_icd_df[diagnoses_icd_df['ICD9_CODE'].isin(disease_list)].copy()
disease_df['subj_hadm'] = list(zip(disease_df['SUBJECT_ID'].astype(int),
                                   disease_df['HADM_ID'].astype(int)))

# Join discharge summary text WITH disease subset ON (subj_id, hadm_id)
patients_df = dischargetext_df[['TEXT', 'subj_hadm']]\
    .join(disease_df.set_index('subj_hadm')['ICD9_CODE'], on='subj_hadm', how='left')\
    .dropna()\
    .drop(columns=['subj_hadm', 'ICD9_CODE’])

# save the WPW subset so no need to regenerate; saving as a tsv file given the large number of commas in medical notes
patients_df.to_csv("wpw_notes.tsv", sep = "\t")
```

Spacy Tutorial – loading and building notes for processing

This tutorial will cover using Spacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#install spacy
! pip install -U pip setuptools wheel
! pip install -U spacy
! python -m spacy download en_core_web_sm

#load patient notes
import pandas as pd
import numpy as np

#connect to google drive
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
%cd drive/MyDrive
notes_cardiac_df = pd.read_csv('wpw_notes.tsv', low_memory=False, sep='\t’)
notes_cardiac_df.shape

#build notes
import spacy
nlp = spacy.load('en_core_web_sm')

#clean up the notes by removing excess punctuation, characters, and numbers
temp = []
def clean_up(text_series):
    return (text_series
            .str.replace('\[\*\*[^\]]*\*\*\]','')
            .str.replace('<[^>]*>', '')
            .str.replace('[\W]+', ' ')
            .str.replace(' \d+', ' '))
temp = clean_up(notes_cardiac_df['TEXT'])
temp.shape

doc = []
for i in range(len(temp)):
  doc.append(nlp(temp[i]))
  print(doc[-1])
  print('*****************************************************************************************************')
```

Spacy Tutorial – analyzing notes and visualization

This tutorial will cover using Spacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Get tokens without punctuations or white space for all patient notes
token_without_punct = []
for i in range(len(doc)):
  token_without_punct.append([token.orth_ for token in nlp(doc[i]) if not token.is_punct or token.is_space])
  print(token_without_punct[-1])
  print('*****************************************************************************************’)

for i in range(len(doc)):
  for token in doc[i]:
    print(token.text, token.pos_)
  print('*************************************************’)

#Name Entity Recognition
entity_doc = []
for i in range(len(doc)):
  entity_doc.append(nlp(doc[i]))
  for ent in entity_doc[-1].ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
  print("*********************************************************")

# Entity Visualizer
from spacy import displacy
for i in range(len(doc)):
  displacy.render(doc[i], style="ent", jupyter=True)
print('*****************************************************************************************************************************************************************’)

#sentence identifier
for i in range(len(doc)):
  for ix, sent in enumerate(doc[i].sents, 1):
    print("Sentence number {}:{}".format(ix, sent))
  print('*****************************************************************************************’)

# Visualizing dependencies: dependence tree
for i in range(len(doc)):
  sentence_spans = list(doc[i].sents)
  displacy.render(sentence_spans, style="dep", jupyter=True)
```

Spacy Tutorial – Entity Visualizer snapshot

Topic: Week 7 Discussion Quest | Module 8 Learning Objectives: | MIMIC NLP | Ethics in AI – Ed Discussion | MIMIC_NLP_GG.ipynb - Colabo

colab.research.google.com/drive/12-G6cdgTHnsmLXj9IAt9q9OB9D0E6IFV#scrollTo=mx3vba6QjSss

MIMIC_NLP_GG.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

Entity Visualizer

```
# Entity Visualizer
from spacy import displacy
for i in range(len(doc)):
  displacy.render(doc[i], style="ent", jupyter=True)
  print('**********************************************************************************************************************************************************')
```

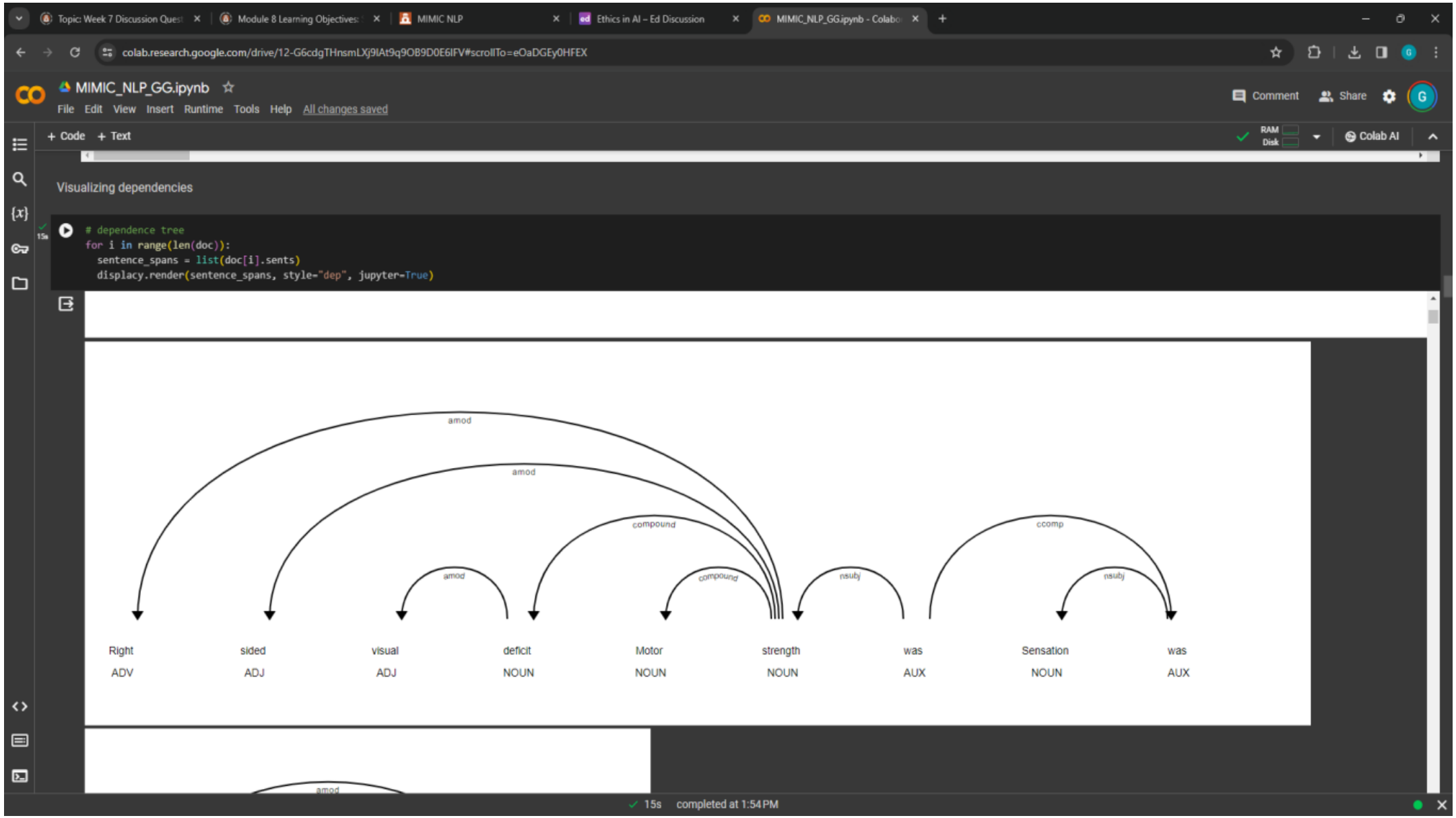
Admission Date Discharge Date Date of Birth Sex M Service HISTORY OF PRESENT ILLNESS The patient is a year old DATE gentleman who presented with right sided visual loss He woke up with normal vision and noted to his wife at a m that he was not able to see out of the right side and complained of a slight headache His son reports that he had normal speech on the telephone but primary care physician called and reported the patient had slurred speech No nausea or vomiting No chest pain or shortness of breath No weakness PAST MEDICAL HISTORY Hypertension Hypercholesterolemia Temporal lobe epilepsy MEDICATIONS ON ADMISSION Tegretol ORG and Lipitor ALLERGIES PHYSICAL PERSON EXAMINATION ON PRESENTATION On physical examination he was in no acute distress He had some difficulty following commands His pupils were equal round and reactive to light The extraocular movements were full There was no nystagmus The neck was supple The chest was clear to auscultation Cardiovascular ORG examination revealed a regular rate and rhythm The abdomen was soft nontender and nondistended Extremity ORG examination revealed edema The skin was normal and dry Cranial NORP nerves II CARDINAL through XII ORG were intact Right sided visual deficit Motor strength was Sensation PERSON was intact PERTINENT RADIOLOGY IMAGING The patient had a magnetic resonance imaging that showed a left temporal lobe mass extending to the parietal area with a large hemorrhage BRIEF SUMMARY OF HOSPITAL COURSE The patient was therefore to the operating room and had a left occipital lobe hematoma evacuated without intraoperative complications The patient was monitored in the Intensive Care Unit ORG postoperatively He was awake and alert Pupils were equal round and reactive to light His speech continued to be garbled He was following commands times four CARDINAL with no motor deficits His vital signs were stable On postoperative day DATE one the patient was alert awake and oriented to name He was following simple commands He was moving all extremities with some right sided neglect The patient was transferred to the regular floor He continued to remain neurologically stable The incision was clean dry and intact The patient was seen by the Physical Therapy Service ORG and Occupational Therapy PERSON and found to be safe for discharge to home A repeat head computed tomography postoperatively showed good evacuation of the hematoma The patient continued to have a right dense homonymous visual field cut on the right side His vital signs remained stable DISCHARGE DISPOSITION He was assessed by Physical Therapy PERSON and Occupational Therapy PERSON and felt to be safe for discharge to home DISCHARGE INSTRUCTIONS ORG FOLLOWUP The patient was to follow up with Dr GPE in days for staple removal and thereafter in three weeks DATE for a repeat head computed tomography MEDICATIONS ON DISCHARGE ORG Percocet one to two CARDINAL tablets by mouth q h as needed Pravastatin PERSON mg by mouth once per day Colace mg by mouth twice per day Lansoprazole PERSON mg by mouth q h Metoprolol GPE mg by mouth twice per day Carbamazepine NORP mg by mouth twice per day for seizures CONDITION PERSON AT DISCHARGE Stable M D Dictated ORG By MEDQUIST36 D T JOB

****

Admission Date Discharge Date Date of Birth Sex M Service MEDICINE Allergies Patient recorded as having No Known Allergies ORG to Drugs Attending Chief Complaint SVT and chest pressure Major Surgical or Invasive Procedure Cardiac Catherization History of Present Illness yo male with h o DM HTN ORG WPW ETOH PERSON and tob abuse was transferred from OSH for ACS ORG and SVT ORG He presented the morning TIME of after having been awoken from sleep by palpitations He has had palpitations before which were converted with a medication the patient does not know at in the ED in the past He has never had chest pressure He went to Hospital where he was found to have SVT ORG and chest pressure He was given a procainamide IV GPE which converted him to NSR ORG rate Twave changes and borderline elevation inf He was given ASA Aggrastat heparin lopressor and NTG ORG The patient did not become CP free and was transferred to and admitted to the CCU ORG team At this time he still had pressure in his chest non radiating not associated with other symptoms A stat cardiac echo was preformed which showed inf hypokinesis Cardiac ORG enzymes revealed a CK ORG of troponin Past Medical History WPW DM Type II on glyburide no complications alcoholism tobacco abuse HTN Social History Drinks ORG

Connected to Python 3 Google Compute Engine backend

Spacy Tutorial – Dependence Tree snapshot



Spacy Tutorial – Word2Vec

This tutorial will cover using Spacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
# Build corpus of all the entities extracted from the notes using spaCy model.
# The corpus is an array of arrays or list of lists where each of the nested lists corresponds to a note.
corpus=[]
for row in range(0, len(doc)):
  str_tokens=[]
  tokens= nlp(doc[row]).ents
  for i in range(0, len(tokens)):
    str_tokens.append(tokens[i].text)
  corpus.append(list(str_tokens))
print(corpus)

#Get Word2Vec
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
import re

import gensim
from gensim.models import Word2Vec

from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
%matplotlib inline

model_spacy = Word2Vec(corpus, min_count=1)

#Check model vocabulary
model_spacy.wv.key_to_index

#Use key word Xanax which is used to calm patients experiencing cardiac events along with its generic form Alprazolam
model_spacy.wv['Xanax’]

#Check for similar words to ‘Xanax’
model_spacy.wv.similar_by_word('Xanax’)

#The output shows a number of words that have no similarity to "Xanax", likely due to its high degree of specificity in drug name.
```

Spacy Tutorial – Word2Vec snapshot

Topic: Week 7 Discussion Quest | Module 8 Learning Objectives: | MIMIC NLP | Ethics in AI – Ed Discussion | MIMIC_NLP_GG.ipynb - Colabo

colab.research.google.com/drive/12-G6cdgTHnsmLXj9IAt9q9OB9D0E6IFV#scrollTo=txXufqVuINxt

MIMIC_NLP_GG.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
model_spacy.wv['Xanax']
```

```
array([-2.5371241e-03, -3.5198121e-03,  9.1029759e-03, -2.4148358e-03,
       -3.8045535e-03,  5.9581017e-03,  6.0487837e-03,  2.5027525e-03,
       -8.6405117e-04, -5.9383800e-03,  5.7881367e-03, -8.3531318e-03,
        8.1412354e-03,  5.3541721e-03,  9.9227820e-03,  1.2321677e-03,
       -9.6106445e-03, -2.5880027e-03,  5.2542915e-03,  5.5384971e-03,
       -2.4439197e-03,  5.5143065e-03,  5.0295866e-03, -4.1510612e-03,
       -5.0427695e-03,  9.6667204e-03,  2.8862953e-03, -6.8242010e-03,
        8.9629600e-03, -3.3323569e-03,  9.5986882e-03, -8.4142732e-03,
        7.7820295e-03,  1.6660260e-03,  1.1572118e-03, -6.7948415e-03,
        5.5852556e-03,  8.8149533e-03,  3.5690686e-03, -7.4194111e-03,
        3.3811054e-03, -8.7301625e-04, -1.4906197e-03, -5.3079091e-03,
       -5.8557922e-03,  6.3645840e-03, -1.2479887e-03,  5.3971936e-03,
        8.9310398e-03, -9.8116565e-03, -7.5463005e-03,  4.1715289e-03,
       -7.0269718e-03,  7.3541566e-03,  3.7977828e-05, -5.5916519e-03,
       -7.7364207e-03, -7.4853860e-03, -2.5122797e-03, -6.3355421e-03,
        9.5297499e-03,  1.6900389e-03, -7.9560820e-03, -6.9486862e-04,
       -3.1241640e-03,  9.5729968e-03, -6.5530515e-03, -8.1773772e-04,
       -3.7331707e-03, -2.6989453e-03, -9.8481979e-03, -5.5521438e-03,
        6.6087880e-05,  8.0866907e-03,  1.9019684e-04,  1.2703157e-03,
       -5.6999372e-03, -7.0234481e-03, -5.6367973e-03, -2.3882450e-03,
        4.4699703e-04, -8.1341956e-03, -2.3789888e-03,  5.3017954e-03,
       -5.1543750e-03, -7.2316844e-03, -5.2084224e-03,  1.9894247e-03,
        5.0234487e-03, -1.9179692e-03,  3.5758705e-03, -7.5830133e-03,
        5.5045220e-03,  4.9451659e-03,  7.5306376e-03, -5.7476875e-03,
       -5.4272637e-03, -3.3051362e-03,  1.1353735e-03,  8.2455054e-03],
      dtype=float32)
```

Check for similar words to 'Xanax'

```
[13] model_spacy.wv.similar_by_word('Xanax')
```

```
[('Ceftriaxone', 0.30362313985824585),
 ('C3 C4', 0.29459142684936523),
 ('simvastatin', 0.2901636064052582),
 ('Potassium', 0.27986130118370056),
 ('Physical Exam', 0.27782225608825684),
 ('NPO', 0.2723531126976013),
 ('Include Gatorade', 0.27068403363227844),
 ('Seasonal', 0.268297404050827),
 ('weeks', 0.2647120654582977 3),
```

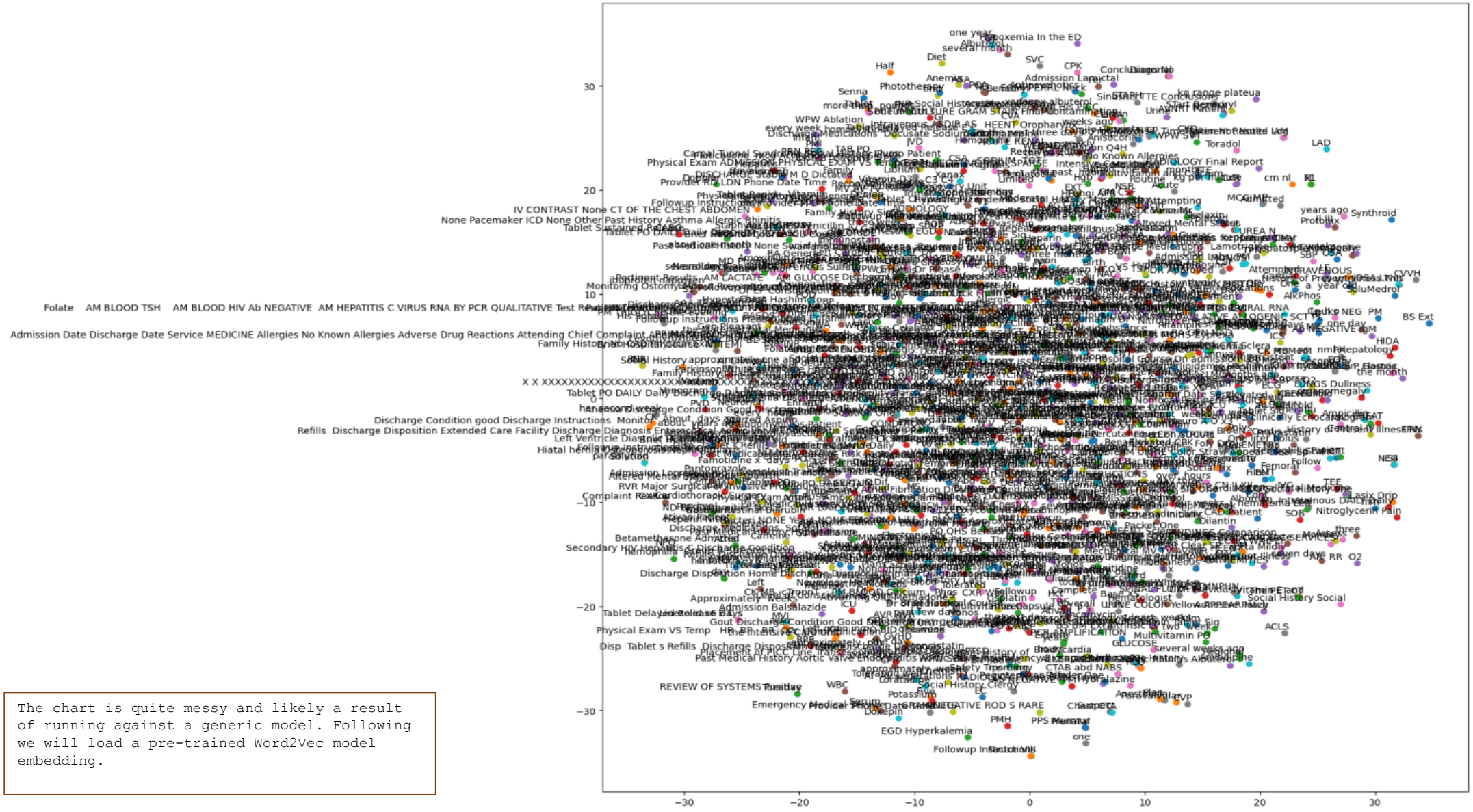
0s completed at 1:57 PM

Spacy Tutorial – tSNE function and plotting

This tutorial will cover using Spacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Define the tSNE plot
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)

    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()

#Plot the corpus in a tSNE plot
vocabs = model_spacy.wv.key_to_index.keys()
new_v = np.array(list(vocabs))
tsne_plot(model_spacy,new_v)
```

Spacy Tutorial – tSNE plot



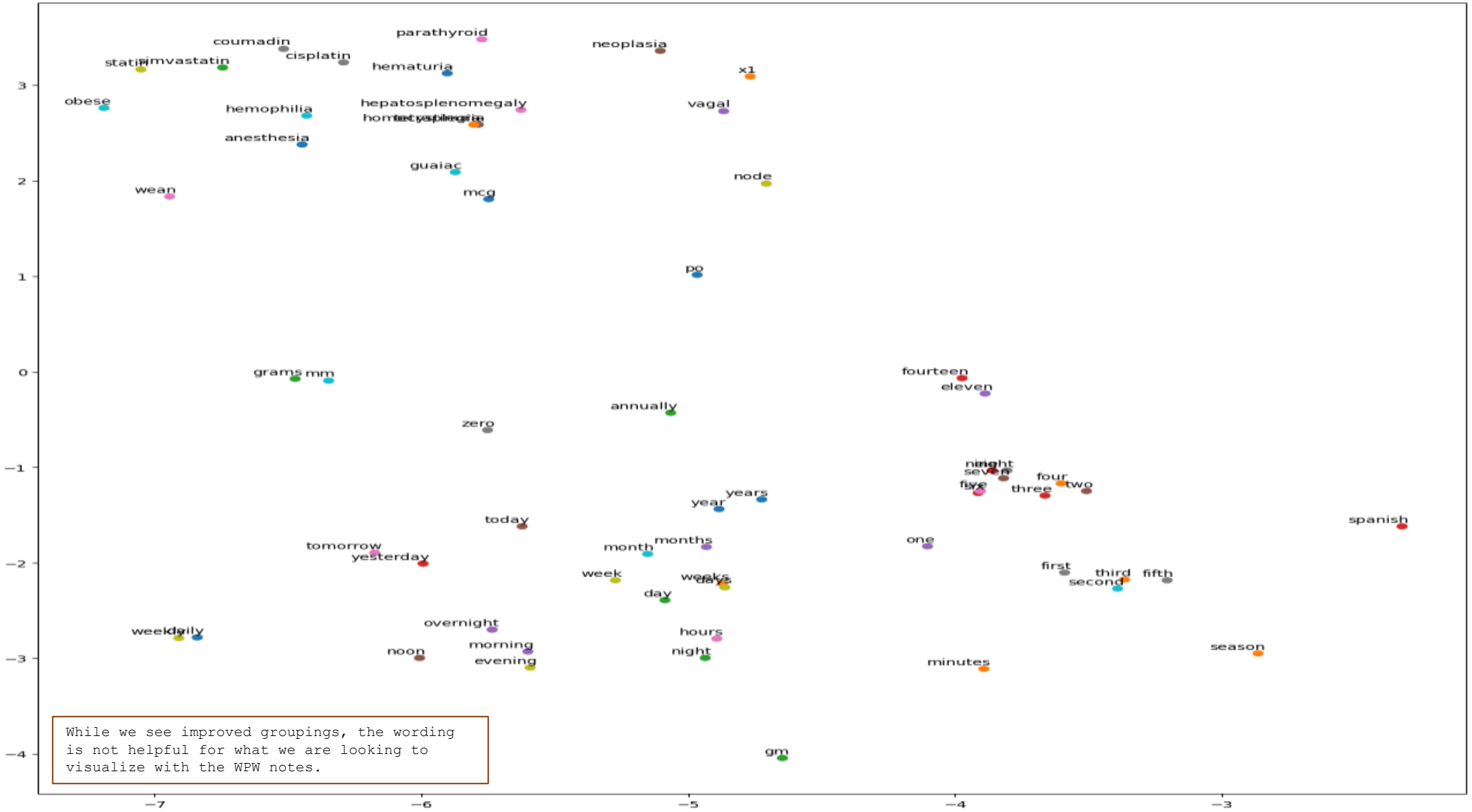
The chart is quite messy and likely a result of running against a generic model. Following we will load a pre-trained Word2Vec model embedding.

Spacy Tutorial – loading pre-trained model

This tutorial will cover using Spacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
# load pre-trained word2vec embeddings
import gensim
import gensim.downloader as api

info = api.info()  # show info about available models/datasets
pretrained_model= api.load("glove-wiki-gigaword-50")  # download the model and return as object ready for use

#Check for words in the pre-trained model that are similar to ‘cardiac’ given WPW is a heart-related medical condition
pretrained_model.most_similar("cardiac")

#Generate a new corpus based on the pre-trained model
corpus_in_pretrained_model = []
for word in vocabs:
  if word in pretrained_model:
    corpus_in_pretrained_model.append(word)
  else:
    print(word) #

#Generate a new tSNE plot using the pre-trained model
tsne_plot(pretrained_model,corpus_in_pretrained_model,True)
```

Spacy Tutorial – tSNE plot on pre-trained model



While we see improved groupings, the wording is not helpful for what we are looking to visualize with the WPW notes.

SciSpacy Tutorial – loading and building notes for processing

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#install SciSpacy
!pip install -U spacy
!pip install scispacy

#Load SciSpacy pre-trained models: using the spaCy NER model trained on BC5CDR corpus given it has the highest efficacy of the imported models
!pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.4.0/en_ner_bc5cdr_md-0.4.0.tar.gz

import scispacy
import spacy

import en_ner_bc5cdr_md
nlp = en_ner_bc5cdr_md.load()

#Import condensed NOTEEVENTS file from the Spacy tutorial
# load condensed NOTEEVENTS file
import pandas as pd
import numpy as np

#connect to google drive
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
%cd drive/MyDrive
notes_cardiac_df = pd.read_csv('wpw_notes.tsv', low_memory=False, sep='\t')#.set_index('ROW_ID’)
```

SciSpacy Tutorial – loading and building notes for processing

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

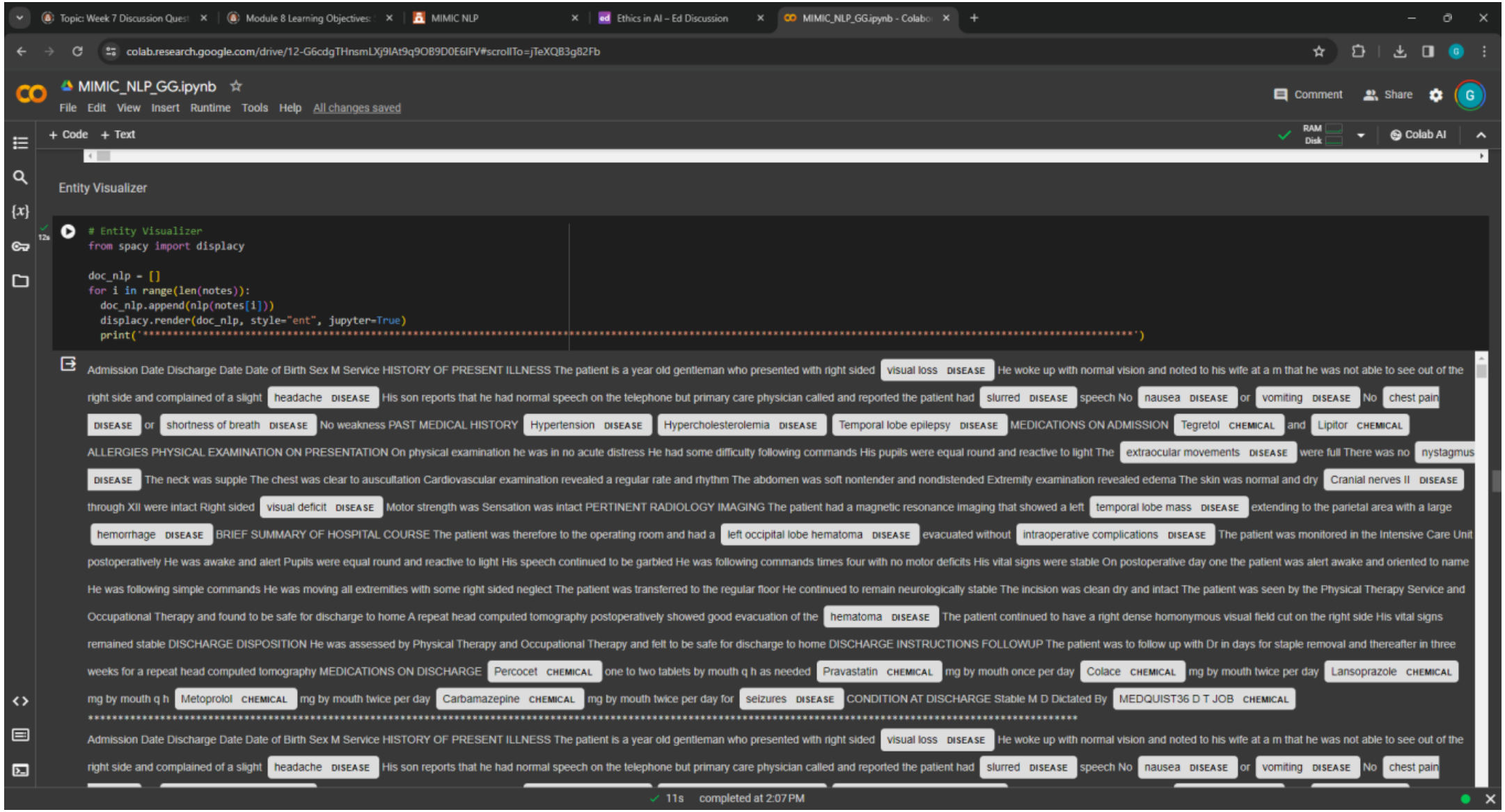
```
#clean up the notes by removing excess punctuation, characters, and numbers
temp = []
def clean_up(text_series):
    return (text_series
            .str.replace('\[\*\*[^\]]*\*\*\]','')
            .str.replace('<[^>]*>', '')
            .str.replace('[\W]+', ' ')
            .str.replace(' \d+', ' '))
temp = clean_up(notes_cardiac_df['TEXT'])
temp.shape

doc_scispacy = []
for i in range(len(temp)):
  doc_scispacy.append(nlp(temp[i]))
  print(doc_scispacy[-1])
  print('*****************************************************************************************************')

#save down cleaned-up version of the WPW notes
text_file = open("clean_wpw_notes.txt", "w")
with open("clean_wpw_notes.txt", "w") as text_file:
    for i in range(len(temp)):
      doc_scispacy.append(nlp(temp[i]))
      print(doc_scispacy[-1], file=text_file)
text_file.close

#Load cleaned-up WPW notes file: clean_wpw_notes.txt
from google.colab import files
uploaded = files.upload()

notes = []
with open('clean_wpw_notes.txt', 'r') as fin:
  lines = fin.readlines()
  for line in lines:
    notes.append(line)
print(notes)
print(len(notes))
```

SciSpacy Tutorial – analyzing notes and visualization

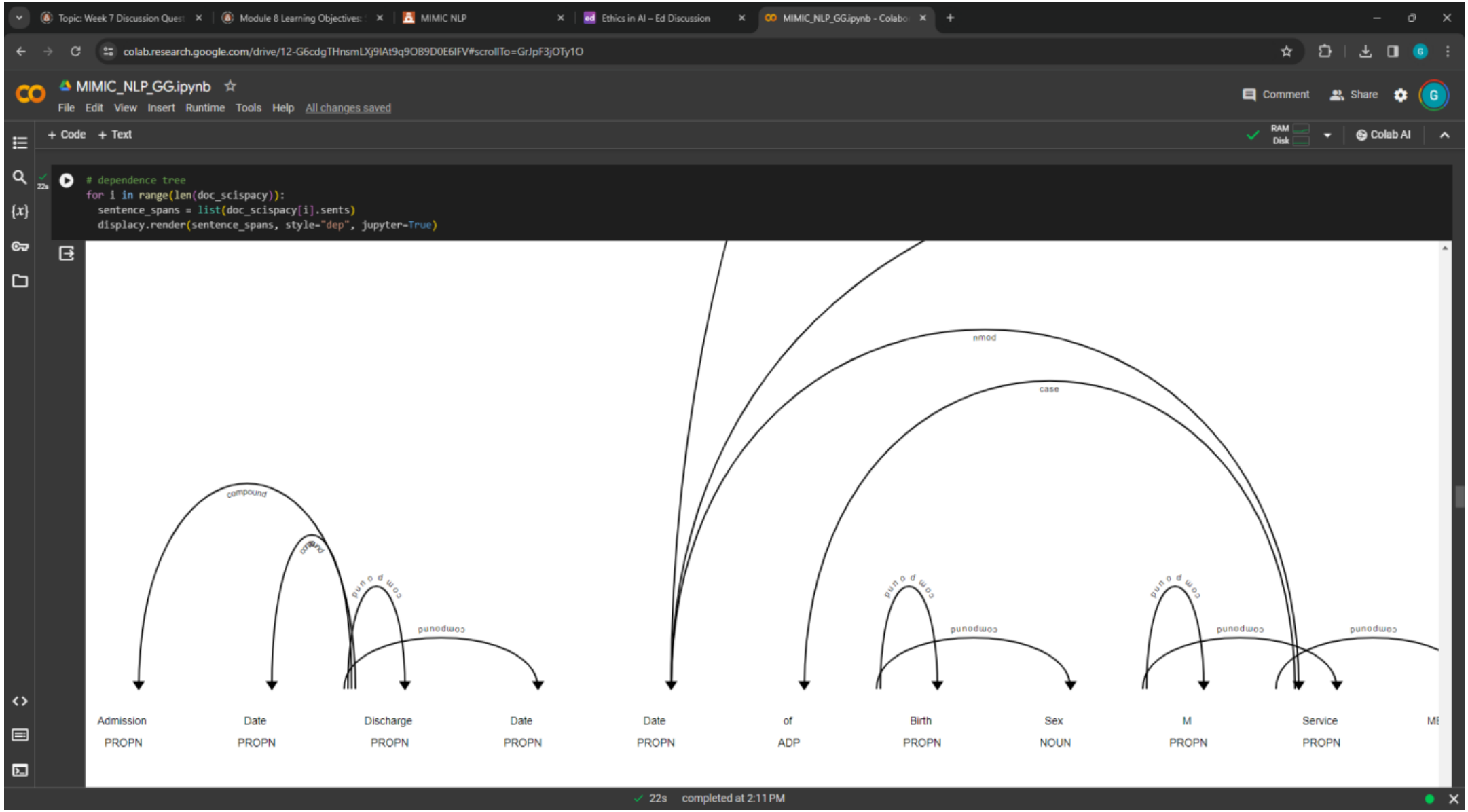
This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#view notes
for i in range(len(notes)):
  print(notes[i])
  print("*********************************************************************************")

#Get tokens for all patient notes
# Token
token_without_punct = []
for i in range(len(notes)):
  token_without_punct.append([token.orth_ for token in nlp(notes[i]) if not token.is_punct or token.is_space])
  print(token_without_punct[-1])
  print('**********************************************************************************’)

# Entity Visualizer
from spacy import displacy

doc_nlp = []
for i in range(len(notes)):
  doc_nlp.append(nlp(notes[i]))
  displacy.render(doc_nlp, style="ent", jupyter=True)
print('*****************************************************************************************************************************************************************’)

#Visualizing dependencies
# dependence tree
for i in range(len(doc_scispacy)):
  sentence_spans = list(doc_scispacy[i].sents)
  displacy.render(sentence_spans, style="dep", jupyter=True)
```

SciSpacy Tutorial – Entity Visualizer snapshot



SciSpacy Tutorial – Dependence Tree snapshot



SciSpacy Tutorial – loading notes and model, Word2Vec

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Build corpus of notes
# Build corpus of all the entities extracted from the notes using spaCy model.
# The corpus is an array of arrays or list of lists where each of the nested lists corresponds to a note.

corpus=[]
for row in range(0, len(notes)):
  str_tokens=[]
  tokens= nlp(notes[row]).ents
  for i in range(0, len(tokens)):
    str_tokens.append(tokens[i].text)
  corpus.append(list(str_tokens))
print(corpus)

#import Word2Vec model
import gensim
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
import re

from gensim.models import Word2Vec

from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
%matplotlib inline

#Using SciSpacy and BC5CDR pre-trained model
import en_ner_bc5cdr_md
nlp = en_ner_bc5cdr_md.load()

#Define model
model_scispacy = Word2Vec(corpus, min_count=1)

#Use key word Xanax which is used to calm patients experiencing cardiac events along with its generic form Alprazolam
model_scispacy.wv['Xanax’]

#find similar words to ‘Xanax in the pre-trained model
model_scispacy.wv.similar_by_word('Xanax’)
```

SciSpacy Tutorial – Word2Vec snapshot

Topic: Week 7 Discussion Quest | Module 8 Learning Objectives | MIMIC NLP | Ethics in AI – Ed Discussion | MIMIC_NLP_GG.ipynb - Colabo

colab.research.google.com/drive/12-G6cdgTHnsmLXj9IAt9q9OB9D0E6IFV#scrollTo=9YjLMvy13mRI

MIMIC_NLP_GG.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Use key word Xanax which is used to calm patients experiencing cardiac events along with its generic form Alprazolam

```
model_scispacy.wv['Xanax']
```

```
array([-1.3359583e-03,  9.6574677e-03,  9.4651076e-04,  1.2216587e-03,
       -4.4621527e-03,  9.1709383e-03, -6.7667672e-03,  4.8836204e-03,
       -6.7308787e-03,  2.4917405e-04, -1.7706396e-03, -1.0171688e-02,
        4.7755996e-03,  6.3900966e-03,  3.1965333e-03,  4.2991331e-03,
        1.2975784e-03, -5.9295655e-03,  9.6756928e-03, -8.8837128e-03,
        9.2947278e-03, -9.2492150e-03, -7.3472662e-03,  8.1484048e-03,
        8.2418034e-03, -3.9818301e-03,  9.5116496e-03,  5.3492975e-03,
       -9.8777646e-03, -9.7668208e-03,  3.1096032e-03,  1.6247818e-03,
        3.6126152e-03, -4.8036119e-03,  7.3745428e-03,  6.2286560e-03,
        7.9076886e-03, -8.4149092e-03,  8.4716995e-04,  7.8514218e-03,
        7.8208126e-05, -1.7137483e-03,  3.3198157e-03,  5.5895397e-03,
        5.7488622e-04, -3.4570375e-03, -5.9004994e-03,  4.5659179e-03,
       -6.4415792e-03, -2.4161923e-03,  9.2425430e-03, -5.9305043e-03,
        3.8118432e-03, -6.4891698e-03, -2.4223322e-04,  2.2607793e-03,
       -9.2026843e-03, -3.1664257e-03, -5.4784492e-04, -3.3065353e-03,
       -5.7100556e-03,  8.1186794e-04, -5.3866222e-03,  9.7892862e-03,
       -5.5098077e-03, -6.4972765e-03,  3.5485055e-03, -1.9269312e-03,
       -4.1023283e-03,  9.6506299e-03,  3.8381414e-03,  5.5343485e-03,
        2.1700948e-03, -3.8751089e-03,  1.2874635e-03,  8.4847519e-03,
       -3.5930134e-04, -9.8931910e-03,  8.8089444e-03, -9.4520766e-03,
        9.6237725e-03, -5.8888537e-03,  1.7093368e-03, -1.1007066e-03,
       -3.5002772e-03, -5.4530725e-03,  4.0272730e-03, -8.3335927e-03,
        5.4139262e-03,  7.7904640e-03,  2.8857575e-03, -9.4772363e-03,
        1.8633602e-04, -2.5334179e-03, -3.2881242e-03,  9.9083195e-03,
        7.1686530e-03, -4.4592498e-03,  5.4475209e-03, -8.7778419e-03],
      dtype=float32)
```

```
[17] #find similar words to 'Xanax' in the pre-trained model
     model_scispacy.wv.similar_by_word('Xanax')
```

```
[('magnesium hydroxide', 0.3892369270324707),
 ('Palpitations', 0.35128992795944214),
 ('vomit', 0.3032970428466797),
 ('NAD', 0.2882729768753052),
 ('hematoma', 0.27626070380210876),
 ('allergic symptoms', 0.2641116678714752),
 ('Dissolve PO Q8H', 0.2602812349796295),
 ('Sinus tachycardia', 0.2598678767681122),
 ('EP', 0.2587347626686096),
 ('Fever', 0.25750356912612915)]
```

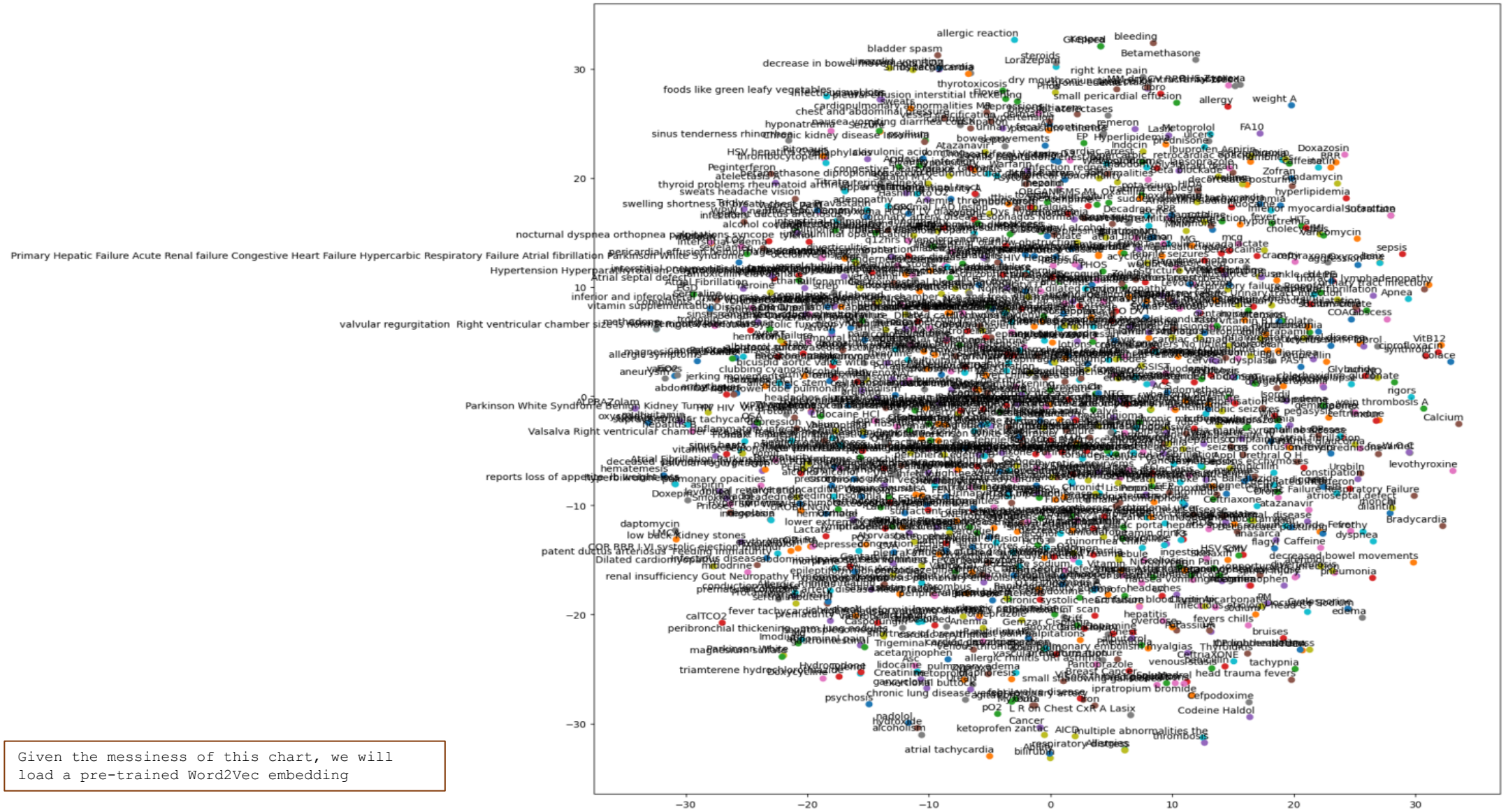
0s completed at 2:14 PM

SciSpacy Tutorial – tSNE function and plotting

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Define the plot function
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)

    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()

#Plot the corpus in a TSNE plot
vocabs = model_scispacy.wv.index_to_key  # Access vocabulary using index_to_key
new_v = np.array(list(vocabs))
tsne_plot(model_scispacy, new_v)
```

SciSpacy Tutorial –tSNE plot



Given the messiness of this chart, we will load a pre-trained Word2Vec embedding

SciSpacy Tutorial – Loading pre-trained model

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

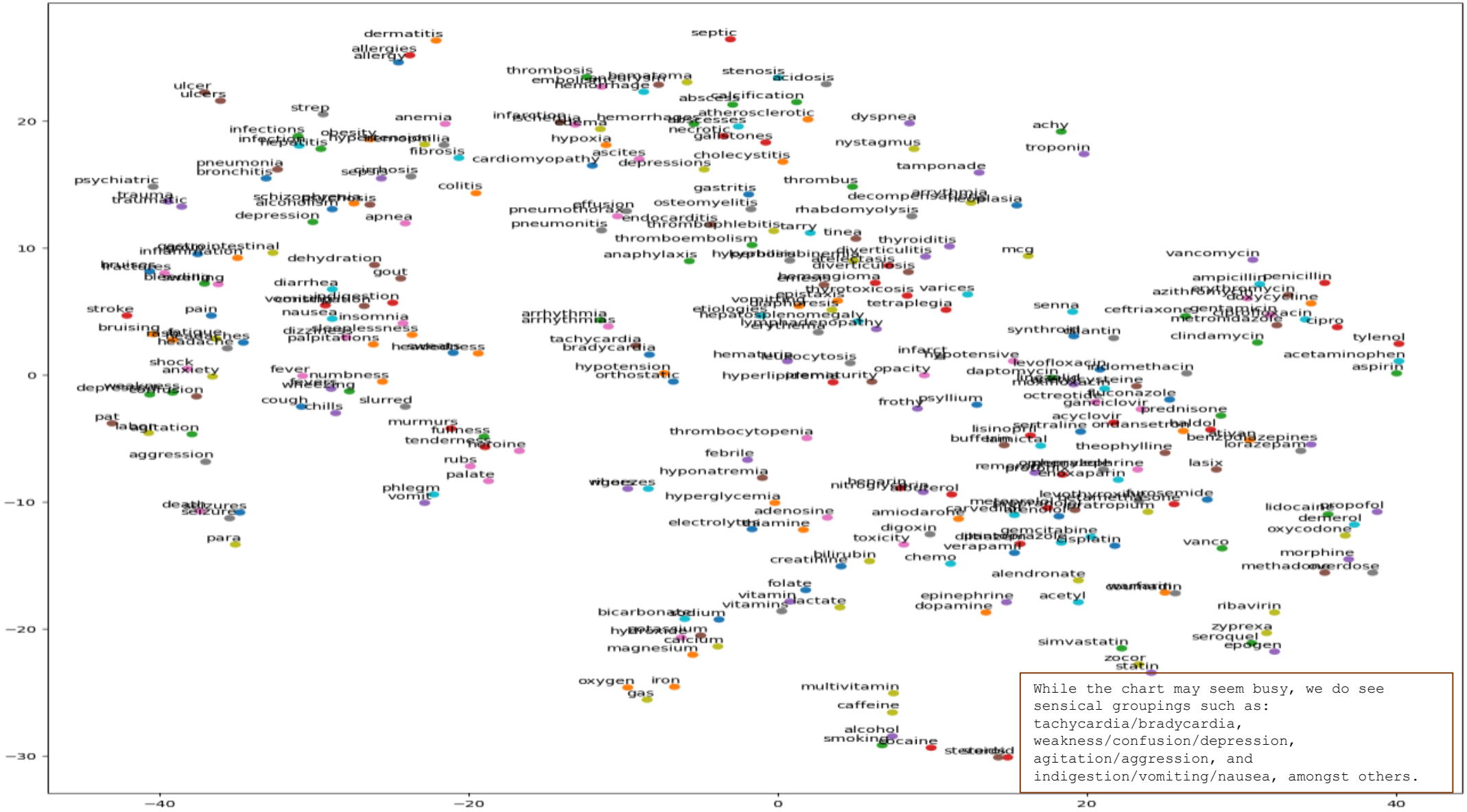
```
#Given the messiness of this chart, we will load a pre-trained Word2Vec embedding

# load pre-trained word2vec embeddings
import gensim
import gensim.downloader as api

info = api.info()  # show info about available models/datasets
pretrained_model= api.load("glove-wiki-gigaword-50")  # download the model and return as object ready for use

#Since we are viewing patient notes regarding those diagnosed with WPW, we will search for words similar to tachycardia
pretrained_model.most_similar("tachycardia")

#Create a new corpus from the pre-trained model
new_corpus_in_pretrained_model = []
for word in new_v:
    if word in pretrained_model.key_to_index:
        new_corpus_in_pretrained_model.append(word)
    else:
        print(word)  # Print out-of-vocabulary words
```

SciSpacy Tutorial – tSNE function and plotting

This tutorial will cover using SciSpacy to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Define the tSNE plot
import numpy as np
def tsne_plot(model, words):
    "Creates a t-SNE model and plots it"
    labels = []
    tokens = []

    for word in words:
        if word in model:
            tokens.append(model[word])
            labels.append(word)
        else:
            print(f"Skipping '{word}' as it is not present in the model's vocabulary.")

    tsne_model = TSNE(perplexity=11, early_exaggeration=12, n_components=2, init='pca', n_iter=1000, random_state=23)
    new_values = tsne_model.fit_transform(np.array(tokens))  # Convert tokens to a NumPy array

    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 the tSNE plot from the pre-trained model
tsne_plot(pretrained_model,new_corpus_in_pretrained_model)
```

SciSpacy Tutorial –tSNE plot on pre-trained model



While the chart may seem busy, we do see sensical groupings such as: tachycardia/bradycardia, weakness/confusion/depression, agitation/aggression, and indigestion/vomiting/nausea, amongst others.

ClinicalBERT Tutorial – loading the model, defining vocabulary

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
!pip install transformers

from transformers import AutoTokenizer, AutoModel,BertTokenizer, BertModel
import torch

bert_model = BertModel.from_pretrained('bert-base-uncased')

print('   Bert_model is type:', type(bert_model))
clinical_model = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")

print('   clinical_model is type:', type(clinical_model))

#ClinicalBERT vocabulary
bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
clinical_tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
blue_tokenizer = AutoTokenizer.from_pretrained('bionlp/bluebert_pubmed_mimic_uncased_L-12_H-768_A-12')
biobert_tokenizer = AutoTokenizer.from_pretrained('dmis-lab/biobert-base-cased-v1.2')
scibert_tokenizer = AutoTokenizer.from_pretrained('allenai/scibert_scivocab_uncased')

# blue_tokenizer = AutoTokenizer.from_pretrained('bionlp/bluebert_pubmed_mimic_uncased_L-24_H-1024_A-16')
print('clinical_tokenizer is type:', type(clinical_tokenizer))
```

ClinicalBERT Tutorial – loading and analyzing tokens from notes

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#using the notes from the Spacy and SciSpacy tutorials

#tokenization
for i in range(len(notes)):
  text = notes[i]
  bert_tokens = bert_tokenizer.tokenize(text)
  clinical_tokens = clinical_tokenizer.tokenize(text)
  bluebert_tokens = blue_tokenizer.tokenize(text)
  biobert_tokens = biobert_tokenizer.tokenize(text)
  # Pad out the clinical bert, bluebert list to be the same length.
  while len(clinical_tokens) < len(bert_tokens):
      clinical_tokens.append("")

  while len(bluebert_tokens) < len(bert_tokens):
      bluebert_tokens.append("")
  while len(biobert_tokens) < len(bert_tokens):
      biobert_tokens.append("")
  # Label the columns.
  print('{:<12} {:<12} {:<12} {:<12}'.format("BERT", "ClinicalBERT","bluebert","biobert"))
  print('{:<12} {:<12} {:<12} {:<12}'.format("----", "-------", "-------", "-------"))

  # Display the tokens.
  for tup in zip(bert_tokens, clinical_tokens, bluebert_tokens, biobert_tokens):
      print('{:<12} {:<12} {:<12} {:<12}'.format(tup[0], tup[1], tup[2], tup[3]))
```

ClinicalBERT Tutorial – loading and analyzing tokens from notes

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Comparing BERT to ClinicalBERT
# Use pandas just for table formatting.
import pandas as pd

# Some strange terms from the paper.
words = ['tachycardia',
         'cardiothoracic',
         'history of present illness'
         ]

# For each term...
for word in words:

    # Print it out
    print('\n\n', word, '\n')

    # Start a list of tokens for each model, with the first one being the model name.
    list_a = ["BERT:"]
    list_b = ["ClinicalBERT:"]

    # Run both tokenizers.
    list_a.extend(bert_tokenizer.tokenize(word))
    list_b.extend( clinical_tokenizer.tokenize(word))

    # Pad the lists to the same length.
    while len(list_a) < len(list_b):
        list_a.append("")
    while len(list_b) < len(list_a):
        list_b.append("")

    # Wrap them in a DataFrame to display a pretty table.
    df = pd.DataFrame([list_a, list_b])

    display(df)

#We see that in general, the two models have similar tokenizations on the selected words
```

ClinicalBERT Tutorial – loading and analyzing tokens from notes

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#The following is a dump of the vocabulary in ClinicalBERT and SciBERT
with open("vocabulary_clinicalbert.txt", 'w') as f:

    # For each token in ClinicalBERT's vocabulary...
    for token in clinical_tokenizer.vocab.keys():

        # Write it out, one per line.
        f.write(token + '\n’)

with open("vocabulary_scibert.txt", 'w') as f:

    # For each token in ClinicalBERT's vocabulary...
    for token in scibert_tokenizer.vocab.keys():

        # Write it out, one per line.
        f.write(token + '\n’)
```

ClinicalBERT Tutorial – Embeddings

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
import numpy as np

def get_word_indeces(tokenizer, text, word):
    '''
    Determines the index or indeces of the tokens corresponding to `word`
    within `text`. `word` can consist of multiple words, e.g., "cell biology".

    Determining the indeces is tricky because words can be broken into multiple
    tokens. I've solved this with a rather roundabout approach--I replace `word`
    with the correct number of `[MASK]` tokens, and then find these in the
    tokenized result.
    '''
    # Tokenize the 'word'--it may be broken into multiple tokens or subwords.
    word_tokens = tokenizer.tokenize(word)

    # Create a sequence of `[MASK]` tokens to put in place of `word`.
    masks_str = ' '.join(['[MASK]']*len(word_tokens))

    # Replace the word with mask tokens.
    text_masked = text.replace(word, masks_str)
    print(text_masked)
    # `encode` performs multiple functions:
    #   1. Tokenizes the text
    #   2. Maps the tokens to their IDs
    #   3. Adds the special [CLS] and [SEP] tokens.
    input_ids = tokenizer.encode(text_masked)
    print(input_ids)
    # Use numpy's `where` function to find all indeces of the [MASK] token.
    mask_token_indeces = np.where(np.array(input_ids) == tokenizer.mask_token_id)[0]

    return mask_token_indeces
```

ClinicalBERT Tutorial – Embeddings

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Select words to mask

words = ['abdomen', 'ablation', 'Acetaminophen', 'Alprazolam', 'aortic', 'artery', 'aspirin', 'Atrial', 'Cardiac', 'cardiothoracic', 'coronary’,
'Coumadin', 'echocardiogram', 'edema', 'fibrillation', 'heart', 'heparin', 'Hypertension', 'Hypotension', ' 'palpitations', 'pericardial', 'pulmonary’,
 'tachycardia', 'ventricular', 'Warfarin', 'WPW', 'Xanax’]

for i in range(len(notes)):
  text = notes[i]
  # [CLS]: 101; [SEP]: 102; [MASK]: 103; [PADDING]: 0
  print(clinical_tokenizer.cls_token_id)
  print(clinical_tokenizer.sep_token_id)
  print(clinical_tokenizer.mask_token_id)
  print(clinical_tokenizer.pad_token_id)
  print(get_word_indeces(clinical_tokenizer, text, words[i]))
```

ClinicalBERT Tutorial – Embeddings

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Note that for visualization in this document, the function is split into two text
Boxes but is a single command in the Jupyter notebook.

#Get embeddings
def get_embedding(b_model, b_tokenizer, text, word=''):
    '''
    Uses the provided model and tokenizer to produce an embedding for the
    provided `text`, and a "contextualized" embedding for `word`, if provided.
    '''

    # If a word is provided, figure out which tokens correspond to it.
    if not word == '':
        word_indeces = get_word_indeces(b_tokenizer, text, word)

    # Encode the text, adding the (required!) special tokens, and converting to
    # PyTorch tensors.
    encoded_dict = b_tokenizer.encode_plus(
                        text,                      # Sentence to encode.
                        add_special_tokens = True, # Add '[CLS]' and '[SEP]'
                        return_tensors = 'pt',     # Return pytorch tensors.
                )

    input_ids = encoded_dict['input_ids']

    b_model.eval()

    # Run the text through the model and get the hidden states.
    bert_outputs = b_model(input_ids)
```

```
    # Run the text through BERT, and collect all of the hidden states produced
    # from all 12 layers.
    with torch.no_grad():

        outputs = b_model(input_ids,output_hidden_states=True )
        # Evaluating the model will return a different number of objects based on
        # how it's  configured in the `from_pretrained` call earlier. In this case,
        # because we set `output_hidden_states = True`, the third item will be the
        # hidden states from all layers. See the documentation for more details:
        # https://huggingface.co/transformers/model_doc/bert.html#bertmodel
        hidden_states = outputs[2]

    # `hidden_states` has shape [13 x 1 x <sentence length> x 768]

    # Select the embeddings from the second to last layer.
    # `token_vecs` is a tensor with shape [<sent length> x 768]
    token_vecs = hidden_states[-2][0]

    # Calculate the average of all token vectors.
    sentence_embedding = torch.mean(token_vecs, dim=0)

    # Convert to numpy array.
    sentence_embedding = sentence_embedding.detach().numpy()

    # If `word` was provided, compute an embedding for those tokens.
    if not word == '':
        # Take the average of the embeddings for the tokens in `word`.
        word_embedding = torch.mean(token_vecs[word_indeces], dim=0)

        # Convert to numpy array.
        word_embedding = word_embedding.detach().numpy()

        return (sentence_embedding, word_embedding)
    else:
        return sentence_embedding
```

ClinicalBERT Tutorial – Embeddings

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#Test out the  get_embeddings function
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re

def clean_text(text):
    # Tokenize the text into words
    words = text.split()

    # Remove special characters and convert to lowercase
    clean_words = [word.lower() for word in words if word.isalnum()]

    # Remove stopwords
    stop_words = set(stopwords.words("english"))
    filtered_words = [word for word in clean_words if word not in stop_words]

    # Remove words with less than 4 characters and numbers. This is done in order to reduce noisy data and numbers dont contribute much in any NLP applications
    filtered_words = [word for word in filtered_words if len(word) >= 4 and not word.isdigit()]

    # Remove duplicate words
    cleaned_text = " ".join(dict.fromkeys(filtered_words))  # This is useful while plotting t-SNE plots

    return cleaned_text
```

ClinicalBERT Tutorial – building notes for analysis

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
#in the first run, there was an issue with the tensor size so truncating notes to run
max_length = 512
notes = [note[:max_length] for note in notes]
words = [word[:max_length] for word in words]

for i in range(len(notes)):
  text = notes[i]
  word = words[i] # words not recognized by the model will return nan, rest of the words will get embeddings
  text = clean_text(text)
  clinical_model.eval()
  # Get the embedding for the sentence, as well as an embedding for 'word'..
  (sen_emb, word_emb) = get_embedding(clinical_model, clinical_tokenizer, text, word)
  print('Embedding sizes:')
  print(sen_emb.shape)
  print(word_emb.shape)
  print(sen_emb)
  print(f'word embeddings for {word}')
  print(word_emb)
```

ClinicalBERT Tutorial – building notes for analysis

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

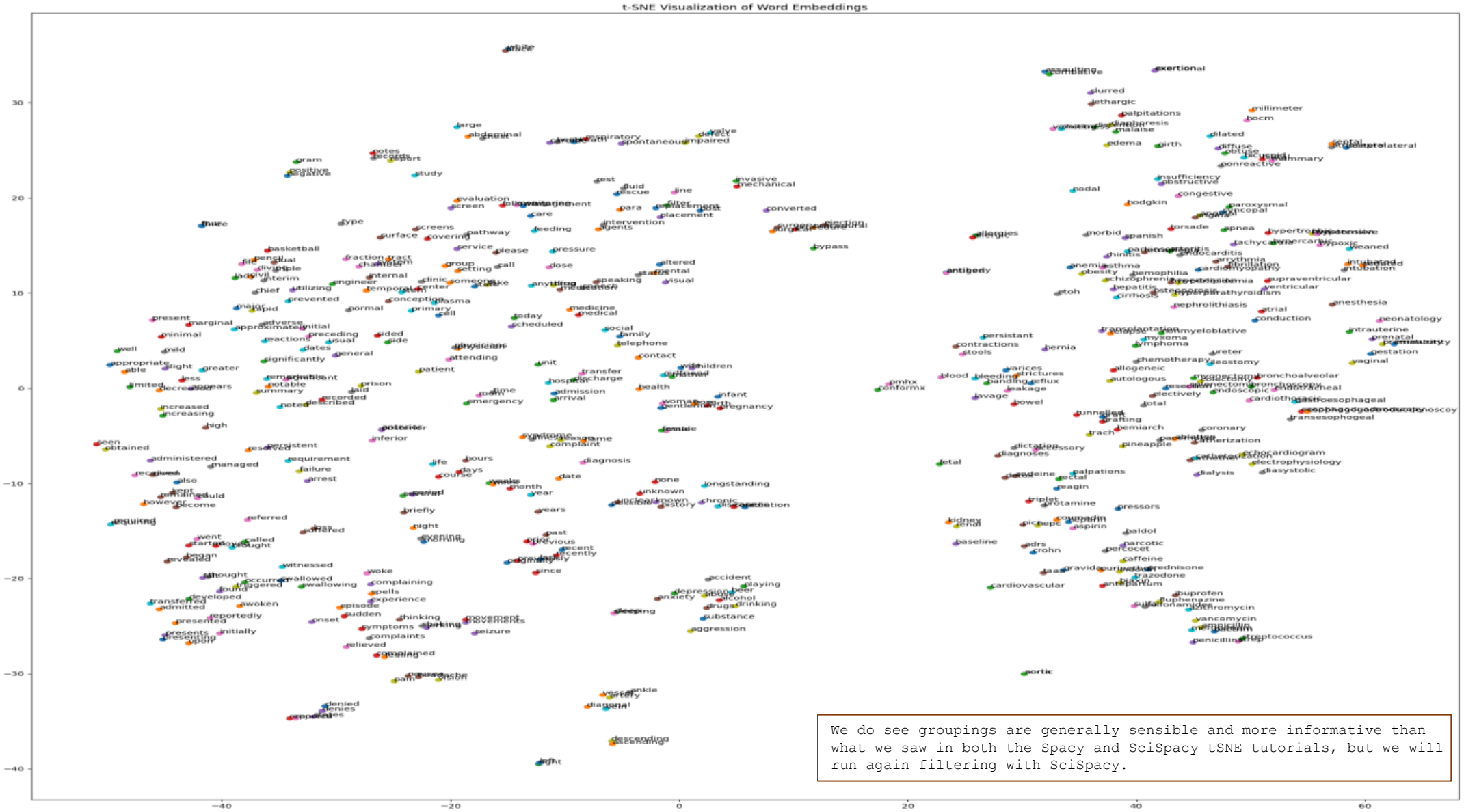
```
import scispacy
import spacy

nlp = spacy.load("en_ner_bc5cdr_md")

# Build corpus of all the entities extracted from the notes using Spacy model.
# The corpus is an array of arrays or list of lists where each of the nested lists corresponds to a note.
corpus=[]
for row in range(0, len(notes)):
  str_tokens=[]
  tokens= nlp(notes[row]).ents
  for i in range(0, len(tokens)):
    str_tokens.append(tokens[i].text)
  corpus.append(list(str_tokens))

print(corpus)

#combining all notes to form a single note which will be easier to visualize in one graph
# Initialize an empty list to store the combined words
notes_combined = []

# Iterate through the sublists and combine the words
for sublist in corpus:
    notes_combined.extend(sublist)

print(notes_combined)

# these notes contains all tokens that are extracted from the original notes using SciSpacy
notes_combined = ' '.join(notes_combined)
notes_combined

# these are the original notes. No SciSpacy is used to filter tokens
all_notes_combined = '. '.join(notes)
all_notes_combined
```

ClinicalBERT Tutorial – visualization of all notes

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
# Visualization of all notes using ClinicalBert
import numpy as np
from sklearn.manifold import TSNE
import string
import matplotlib.pyplot as plt
from transformers import AutoModel, AutoTokenizer

# Load the BERT model and tokenizer
clinical_model = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
clinical_tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
clinical_model.eval()
# Example input text
input_text = clean_text(all_notes_combined)
# Tokenize the input text using the BERT tokenizer
#input_tokens = clinical_tokenizer.tokenize(input_text)
input_tokens = input_text.split()
# Initialize an empty list to store word embeddings
word_embs = []
for token in input_tokens:
    # Check if the token is a valid word
    if token not in string.punctuation:
        # Encode the token using the BERT model
        inputs = clinical_tokenizer(token, return_tensors="pt")
        with torch.no_grad():
            outputs = clinical_model(**inputs)
        token_emb = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
        word_embs.append(token_emb)

# Perform t-SNE dimensionality reduction
tsne_model = TSNE(n_components=2, perplexity=10, random_state=42)
word_embs_2d = tsne_model.fit_transform(np.array(word_embs))
# Create a scatter plot of the word embeddings in 2D space
plt.figure(figsize=(25, 25))
for i in range(len(word_embs_2d)):
    plt.scatter(word_embs_2d[i, 0], word_embs_2d[i, 1])
    plt.annotate(input_tokens[i], (word_embs_2d[i, 0], word_embs_2d[i, 1]))
plt.title("t-SNE Visualization of Word Embeddings")
plt.show()
```

ClinicalBERT Tutorial – tSNE plot with no filtering



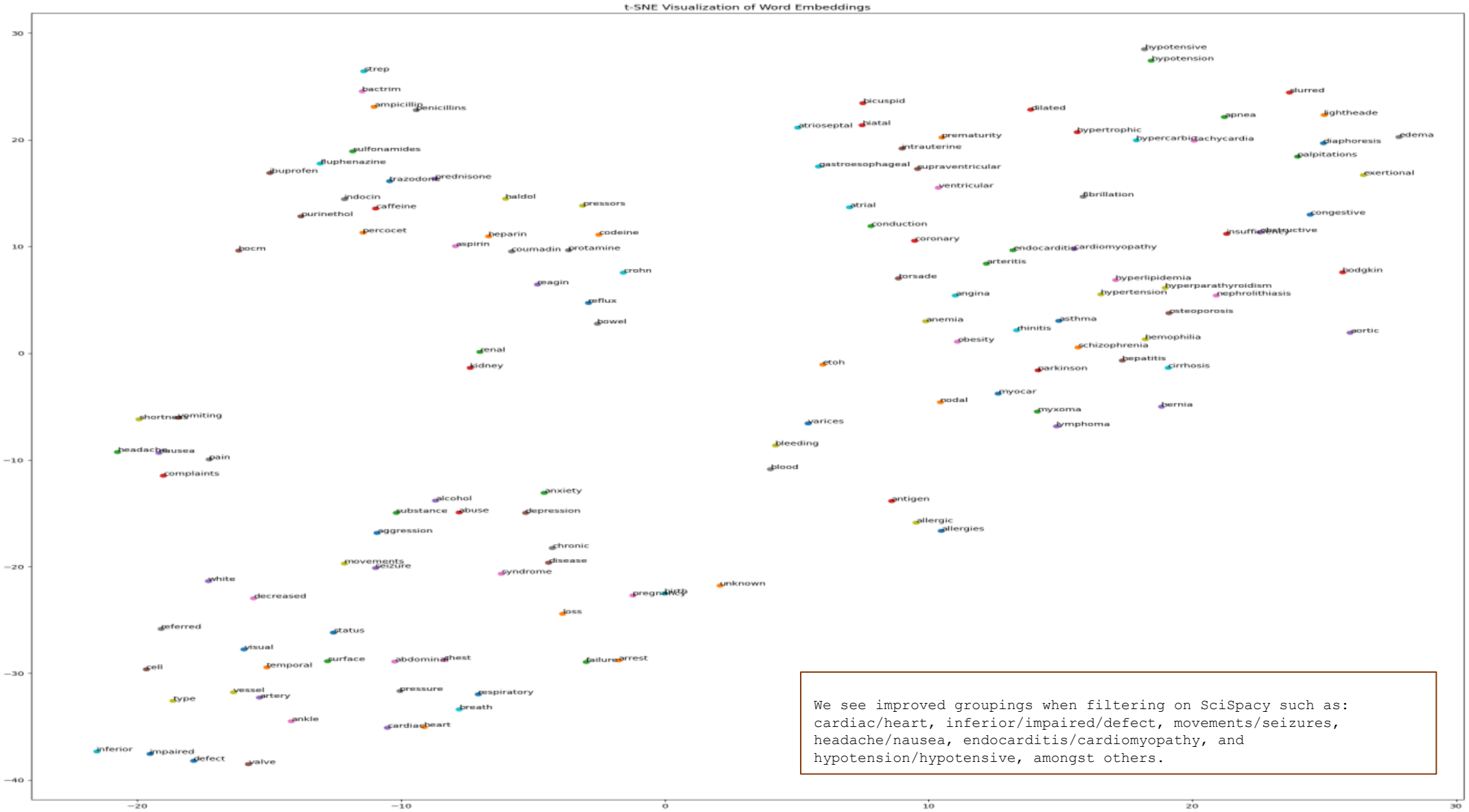
We do see groupings are generally sensible and more informative than what we saw in both the Spacy and SciSpacy tSNE tutorials, but we will run again filtering with SciSpacy.

ClinicalBERT Tutorial – visualization of notes filtered on SciSpacy

This tutorial will cover using ClinicalBERT to analyze a subset of medical notes covering the discharge summaries of patients diagnosed with Wolf Parkinson White (WPW) syndrome.

```
# Visualization of notes filtered with SciSpacy using ClinicalBert
import numpy as np
from sklearn.manifold import TSNE
import string
import matplotlib.pyplot as plt
from transformers import AutoModel, AutoTokenizer
# Load the BERT model and tokenizer
clinical_model = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
clinical_tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
clinical_model.eval()
# Example input text
input_text = clean_text(notes_combined)
# Tokenize the input text using the BERT tokenizer
#input_tokens = clinical_tokenizer.tokenize(input_text)
input_tokens = input_text.split()
# Initialize an empty list to store word embeddings
word_embs = []
for token in input_tokens:
    # Check if the token is a valid word
    if token not in string.punctuation:
        # Encode the token using the BERT model
        inputs = clinical_tokenizer(token, return_tensors="pt")
        with torch.no_grad():
            outputs = clinical_model(**inputs)
        token_emb = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
        word_embs.append(token_emb)
# Perform t-SNE dimensionality reduction
tsne_model = TSNE(n_components=2, perplexity=10, random_state=42)
word_embs_2d = tsne_model.fit_transform(np.array(word_embs))

# Create a scatter plot of the word embeddings in 2D space
plt.figure(figsize=(25, 25))
for i in range(len(word_embs_2d)):
    plt.scatter(word_embs_2d[i, 0], word_embs_2d[i, 1])
    plt.annotate(input_tokens[i], (word_embs_2d[i, 0], word_embs_2d[i, 1]))

plt.title("t-SNE Visualization of Word Embeddings")
plt.show()
```

ClinicalBERT Tutorial – tSNE plot using SciSpacy filtering



We see improved groupings when filtering on SciSpacy such as: cardiac/heart, inferior/impaired/defect, movements/seizures, headache/nausea, endocarditis/cardiomyopathy, and hypotension/hypotensive, amongst others.