MIMIC III: Natural Language Processing

AI 395T - AI in Healthcare - Dr. Ying Ding

Joseph Skrovan, 2/23/2025

Notes:

- I used natural language processing (NLP) modules to process notes from the MIMIC III dataset.
- I limited notes to those containing the word, “metformin”, to primarily select patients with type 2 diabetes.
- The Jupyter notebook and support program I wrote are available from this Github repo: git@github.com:jskrovan2/aihc.git
  - **`nlp.ipynb`** - Code to process clinical notes with natural language processing.
- I downloaded the MIMIC III dataset and accessed it locally with the duckdb Python module.
  - I distilled the large NOTEEVENTS table with, **`distill_notes.py`**, to produce a much smaller table of only notes including the word, “metformin”, a first-line type 2 diabetes drug.
- Some code was copied from class Jupyter notebooks from class Modules 6 and 7.
- Some code was produced by ChatGPT. No MIMIC III data was sent to ChatGPT.

Parsing Notes with spacy tokens

I started with a sub-set of notes that included “taking metformin” but not “diabetes” to create a smaller set. I parsed the text into “tokens” with spacy.

```
import duckdb

# Select notes that mention taking metformin but not diabetes to find a
# smaller set of notes to review.
con = duckdb.connect()
table = 'noteevents_metformin.parquet'
df = con.execute(f"""
    SELECT * FROM '{table}'
    WHERE text ILIKE '%taking metformin%' and text NOT ILIKE '%diabetes%';
""").df()
```

```
import spacy
nlp = spacy.load('en_core_web_sm')
lines = []
for text in df["TEXT"]:
    doc = nlp(text)
    tokens_without_punct = [token.orth_ for token in doc if not token.is_punct | token.is_space]
    lines.append(tokens_without_punct)

for line in lines:
    print(f"Token count: {len(line)}")
    print(line)
    print('-'*80)
```

```
Token count: 1740
['Admission', 'Date', '2196', '12', '13', 'Discharge', 'Date', '2196', '12', '15', 'Date', 'of', 'Birth', '2161', '1', '2
--------------------------------------------------------------------------------
Token count: 140
['Name', 'Known', 'lastname', 'Known', 'firstname', '3650', 'Unit', 'No', 'Numeric', 'Identifier', '17835', 'Admission',
--------------------------------------------------------------------------------
Token count: 724
['Chief', 'Complaint', '55yo', 'm', 'with', 'diverticulitis', 'c', 'b', 'colovesicular', 'fistula', 's', 'p', 'open', 'fi
--------------------------------------------------------------------------------
Token count: 771
['Chief', 'Complaint', '55yo', 'm', 'with', 'diverticulitis', 'c', 'b', 'colovesicular', 'fistula', 's', 'p', 'open', 'fi
--------------------------------------------------------------------------------
Token count: 1968
['Chief', 'Complaint', 'CHIEF', 'COMPLAINT', 'Initial', '>', 'BRBPR', 'Reason', 'for', 'transfer', '>', 'Hypoxia', 'HPI',
--------------------------------------------------------------------------------
Token count: 166
['MICU', 'Nursing', 'Progress', 'NOte-', '7a-7p', 'Patient', 'called', 'out', 'to', 'floor', 'transfer', 'note', 'done',
--------------------------------------------------------------------------------
Token count: 139
['Condition', 'Update', 'Please', 'see', 'carevue', 'for', 'specifics', 'Pt', 'alert', 'and', 'oriented', 'NSR', 'TMax',
--------------------------------------------------------------------------------
Token count: 445
['npn', '0700', '1900', 'code', 'status', 'full', 'all', 'nkda', 'pmh', 'stage', '3', 'rectal', 'ca', 'ileostomy', '2cycl
--------------------------------------------------------------------------------
```

Parsing Notes with spacy tokens cont.

I looked at the words around a few keywords.

I printed the keyword in the center and the words before and after it in the notes.

```
context = 4
words_of_interest = {"metformin", "cocaine", "alcohol"}
max_length = max(len(word) for word in words_of_interest)
for line in lines:
    metformin_indices = [i for i, token in enumerate(line) if token.lower() in words_of_interest]
    for i in metformin_indices:
        before = ' '.join(line[i-context:i])
        after = ' '.join(line[i+1:i+context+1])
        print(f'{before:>40}  {line[i].center(max_length)}  {after}')
    print('-'*100)
```

Tragically, the metformin overdose is not her only problem.

```
      LF 1257 Chief Complaint  Metformin  Overdose Major Surgical or
           took 10 pills of  metformin  thinking that it was
      recent relapse of crack   cocaine   and heroin The patient
         for BZD opiates and   cocaine   Her glucose remained at
      relapse with heroin and   cocaine   use Family History Unable
      barbitr NEG opiates POS   cocaine   POS amphetm NEG mthdone
back pain following accidental  metformin  overdose Plan Overdose The
         taking 10 pills of  metformin  Denies suicidal ideation preceding
  an expected complication of  metformin  overdose Patient was found
   arrival Pt reported taking  metformin  but this is not
        history of drug and   alcohol   abuse Tox screen +
       Tox screen + opiates   cocaine   benzos on admission Placed
----------------------------------------------------------------------------
        that patient was on  metformin  at home prior to
         she is not taking  metformin  and will not in
----------------------------------------------------------------------------
      by the patient taking  metformin  and glyburide both of
----------------------------------------------------------------------------
      by the patient taking  metformin  and glyburide both of
----------------------------------------------------------------------------
           and thought 1 6   alcohol   2 Alcohol abuse long
             1 6 alcohol 2   Alcohol   abuse long standing alcohol
  Alcohol abuse long standing   alcohol   abuse has denied AAA
     bypass now only taking  metformin  6 HTN 7 Cholelithiasis
     recently Tobacco 1 ppd   Alcohol   Significant past history including
----------------------------------------------------------------------------
      101 1800=161 p taking  metformin  500 mg Will discuss
----------------------------------------------------------------------------
     all insulin but taking  metformin  Hospital1 26 Pt oob
----------------------------------------------------------------------------
      because pt was taking  metformin  at home so this
```

Parsing Notes with spacy entities

```python
import spacy
nlp = spacy.load('en_core_web_sm')
ent_lines = []
for text in df["TEXT"]:
    doc = nlp(text)
    ents = [ent.text for ent in doc.ents]
    ent_lines.append(ents)

for line in ent_lines:
    print(f"Ent count: {len(line)}")
    print(line)
    print('-'*80)
```

✓ 1.1s

```
Ent count: 149
['Admission Date:  ', '2196-12-13', '2196-12-15', '2161', 'F\n\nService', 'Name3 (LF', '1257', 'Metformin Overdose', '35 year-old',
--------------------------------------------------------------------------------
Ent count: 15
['3650', 'Identifier 17835', '2152-8-22', '2152-9-9', '2081-12-8', 'F\n\nService', 'Name3 (LF', '1472', 'Addendum', 'Patient', 'Home
--------------------------------------------------------------------------------
Ent count: 133
['55yo', '8-10', '24 Hour', 'last night', 'Tylenol', 'PO', 'un', '276', 'FS', '2', '100-140s', '2150-8-21', '08:10 AM', 'Levofloxac
--------------------------------------------------------------------------------
Ent count: 145
['55yo', '8-10', '24 Hour', 'last night', 'Tylenol', 'PO', 'un', '276', 'FS', '2', '100-140s', '160', '100cc', 'NS', 'later today',
--------------------------------------------------------------------------------
Ent count: 269
['HPI', '60', '1-26', '2-4', 'E. Coli', '2169', '2-19', 'C.', 'GI', 'C.', 'IV', 'PO', 'HCAP', 'Surgery', 'HRS', 'US', '2-26', '2-25
--------------------------------------------------------------------------------
Ent count: 23
['MICU Nursing Progress', 'Neuro-\n', 'OOB', 'a great day', 'Ambulated w/', '20-22', 'prod white', 'Cont', '12-12', 'NPO', 'tonight
--------------------------------------------------------------------------------
Ent count: 17
['Condition Update\nPlease', 'NSR', '99', '02', '88-98%', 'RA', 'N/C. Wheezes', 'Interpreter', 'this pm', 'Foley', 'urine w/', 'LE',
--------------------------------------------------------------------------------
Ent count: 31
['0700-1900', '3', '2cycles', 'the past two weeks', 'ARF', '6.3', '8.3', 'ed', '2liters', '1 liter', 'first', 'md', '2-22', 'md', '2
--------------------------------------------------------------------------------
```

Visualizing spacy entities

Spacy entities are not too useful in this case.

```
# Entity Visualizer
for text in df["TEXT"]:
    doc = nlp(text)
    print(f'{len(doc.ents)} entities')
    print('-'*100)
# Display last document
displacy.render(doc, style="ent", jupyter=True)
```

✓ 1.1s

```
149 entities
----------------------------------------------------------------------------------------------------
15 entities
----------------------------------------------------------------------------------------------------
133 entities
----------------------------------------------------------------------------------------------------
145 entities
----------------------------------------------------------------------------------------------------
269 entities
----------------------------------------------------------------------------------------------------
23 entities
----------------------------------------------------------------------------------------------------
17 entities
----------------------------------------------------------------------------------------------------
31 entities
----------------------------------------------------------------------------------------------------
```

Good job husband; that's where you're supposed to be.

Entity Counts of scispacy models

Scispacy entities are much more pertinent. I printed a table of entity counts found by the various models.

```
import scispacy
import en_core_sci_md
import en_ner_craft_md
import en_ner_jnlpba_md
import en_ner_bc5cdr_md
import en_ner_bionlp13cg_md
nlps = {
    'core_sci_md': en_core_sci_md.load(),
    'ner_craft_md': en_ner_craft_md.load(),
    'ner_jnlpba_md': en_ner_jnlpba_md.load(),
    'ner_bc5cdr_md': en_ner_bc5cdr_md.load(),
    'ner_bionlp13cg_md': en_ner_bionlp13cg_md.load(),
}
for model in nlps:
    print(f'{model.center(20)}', end=' ')
print()
print('-'*100)
for text in df["TEXT"]:
    for nlp in nlps.values():
        doc = nlp(text)
        print(f'  {len(doc.ents):10}        ', end=' ')
    print()
    print('-'*100)
# Display last document with ner_bionlp13cg_md
doc = nlps['ner_bionlp13cg_md'](df["TEXT"].iloc[-1])
displacy.render(doc, style="ent", jupyter=True)
```

✓ 7.6s

core_sci_md	ner_craft_md	ner_jnlpba_md	ner_bc5cdr_md	ner_bionlp13cg_md
560	45	33	96	163
29	3	3	3	11
258	13	3	66	100
264	13	3	61	96
699	47	26	157	232
60	0	1	3	15
51	2	3	12	12
112	2	4	14	25

pmh: stage 3 rectal ca, ileostomy, 2cycles chemotherapy, severe depression requiring ect tx, dm, dementia, arf

reason for admission: brought into EW by husband for inc lethargy and general malaise over the past two weeks. pt was found to be in ARF GENE_OR_GENE_PRODUCT with a cr of 6.3 and a k of 8.3, in the ed pt recieved 2liters of fluid and kayexolate. pt had a foley SIMPLE_CHEMICAL cath placed and immed drained 1 liter. pt was sent to micu for monitoring.

signicant events: pt had renal CANCER u/s this am which was negative except for small cyst in kidney ORGAN . at first md's were questioning blockage [**2 SIMPLE_CHEMICAL -22**] the fact pt drained 1000cc upon placement of foley. however, u/s was negative. md's believe pt was extremely dehydrated and uremic which may be the cause of the ms change. nephrology was consulted and questioning possiblity of neurogenic bladder ORGAN . pt lactate SIMPLE_CHEMICAL level has been high although pt has no signs of infections. this is thought to be because pt was taking metformin SIMPLE_CHEMICAL at home so this has been placed on hold. pt recieved bicarb SIMPLE_CHEMICAL drip which helped with the acidemia. gtt was d/c'ed this am. pt continues with agressive hydration with ivf SIMPLE_CHEMICAL at 200cc/hr. 1300 labs were wnl. pt had multiple large bm's second to kayexolate. surgery was made aware that pt was having bm's out of rectum MULTI_TISSUE_STRUCTURE despite placement of ileostomy. [**2-22**] ms changes pt has been npo. pt attempted to drink water and pudding and coughed both times. md's aware may do swallow study in am. pt was unable to swallow pills.

neuro: pt awake/alert/oriented times 3. pt is slow to answer questions and has a very flat affect. husband says this baseline for pt. pt able move all ext. pt denies pain. pt has psych ORGANISM meds on hold [**2 SIMPLE_CHEMICAL -22**] lethargy and unable to swallow properly. ? swallow ORGANISM_SUBDIVISION eval in am.

cv: nsr-st. hr 90-115. pt abp GENE_OR_GENE_PRODUCT is 105-140's. pt is ns at 200cc/hr. pt has ppp bilaterally. generalized trace edema PATHOLOGICAL_FORMATION noted.

resp: pt o2 sats 97-100% on ra GENE_OR_GENE_PRODUCT . lungs ORGAN clear.

gi/gu: pt is npo. pt has ileostomy draining brown liquid/loose stool. stool ORGANISM sample for c-diff sent to lab. guaic positive. abd soft PATHOLOGICAL_FORMATION distended with positive bowel ORGAN sounds. foley in place draining adequate amounts of uop. urine ORGANISM_SUBSTANCE is clear yellow. responding well to fluids.

access: pt has 3 piv's and an a line.

skin ORGAN : intact, lips PATHOLOGICAL_FORMATION dry/cracked from dehydration.

endo: fsq6hours CANCER .

social: husband at bedside.

plan: monitor labs, cont icu monitoring, may be c/o in am.

Build scispacy corpus

I scanned 1000 notes and printed the most common entities. The notes (by selection) are guaranteed to have “metformin”, but “cardiac” was the most common term. I won’t speculate.

I removed terms I was not interested in.

```
# Build corpus of all the entities extracted from the notes using bionlp13cg_md model
# The corpus is a list of lists where each of the nested lists corresponds to a note.
df_met = con.execute(f"""
    SELECT * FROM 'noteevents_metformin.parquet';
""").df()
remove_terms = {'patient', 'tablet sig', 'tablet po', 'hospital1', 'hospital2', 'hospital3',
                'tablet', 'blood', 'capsule', 'refills:*2', 'p.o', '[*', 'po', 'oral', 'chewable',
                'q.d', 'b.i.d', 'tid', 't.i.d', 'q4h', 'qhs', 'prn', 'q6h', 'q8h', 'q12h', 'q24h'}
term_counts = {}
corpus=[]
for i, text in enumerate(df_met["TEXT"].iloc[:1000]):
    ents = nlps['ner_bionlp13cg_md'](text).ents
    ents = [ent.lemma_ for ent in ents if
            '**' not in ent.lemma_ and
            not re.match('\d', ent.lemma_) and
            ent.label_ not in {'DATE', 'TIME'} and
            ent.lemma_.lower() not in remove_terms and
            'tablet' not in ent.lemma_.lower()
           ]
    corpus.append(ents)
    if i % 100 == 0:
        print(f'{i}',
              end=' ',
              flush=True)
    for ent in ents:
        term_counts[ent] = term_counts.get(ent, 0) + 1

print()
common_ents = sorted(term_counts, key=term_counts.get, reverse=True)
for ent in common_ents[:10]:
    print(f'{ent:20} {term_counts[ent]}')

print(corpus)
```

✓ 7m 51.8s

```
0 100 200 300 400 500 600 700 800 900
cardiac              1640
metformin            1621
heart                1516
aspirin              1314
edema                1120
coronary artery      1101
insulin              1071
lisinopril           938
pulmonary            862
lung                 854
[['Zocor', 'lescol', 'jugular vein', 'man', 'mitral', 'aortic insufficiency', 'ventricular systolic heart', 'svg-
```

Scispacy Chemical and Organ corpus

Built a similar corpus of only “chemical” and “organ” entities. I was interested in this because they are more common terms I am familiar with.

I removed the same set of uninteresting terms.

```python
# Build corpus of all the entities extracted from the notes using bionlp13cg_md model
term_counts_organ_chemical = {}
corpus_organ_chemical=[]
for i, text in enumerate(df_met["TEXT"].iloc[:1000]):
    ents = nlps['ner_bionlp13cg_md'](text).ents
    ents = [ent.lemma_ for ent in ents if
            '**' not in ent.lemma_ and
            not re.match('\d', ent.lemma_) and
            ent.label_ in {'SIMPLE_CHEMICAL', 'ORGAN'} and
            ent.lemma_.lower() not in remove_terms and
            'tablet' not in ent.lemma_.lower()
           ]
    corpus_organ_chemical.append(ents)
    if i % 100 == 0:
        print(f'{i}',
              end=' ',
              flush=True)
    for ent in ents:
        term_counts_organ_chemical[ent] = term_counts_organ_chemical.get(ent, 0) + 1

print()
common_organ_chemical_ents = sorted(term_counts_organ_chemical, key=term_counts_organ_chemical.get, reverse=True)
for ent in common_organ_chemical_ents[:50]:
    print(f'{ent:20} {term_counts_organ_chemical[ent]}')
```

✓ 7m 53.9s

```
0 100 200 300 400 500 600 700 800 900
metformin            1619
cardiac              1518
heart                1461
aspirin              1309
lisinopril           938
pulmonary            862
lung                 826
lasix                806
bowel                688
bp                   673
coumadin             648
creatinine           576
oxygen               506
```

Related Terms with Word2Vec

The “window” parameter to Word2Vec was very helpful. It sets the context size around each word that is considered when training the embedding. It is the maximum distance between the target word and the surrounding words (context) considered.

Without this, all words in a long note are considered, and correlations are more random.

“max_vocab_size=1000” prevented my plots (see next slides) from being too congested to read individual labels.

```
from gensim.models import Word2Vec
model1 = Word2Vec(corpus, min_count=50, max_vocab_size=1000, window=5)
model2 = Word2Vec(corpus_organ_chemical, min_count=50, max_vocab_size=1000, window=5)
```

✓ 0.1s

```
for similar in model1.wv.similar_by_word('insulin')[:20]:
    print(similar)
```

✓ 0.0s

```
('blood sugar', 0.9877884387969971)
('nph', 0.9615339636802673)
('protonix', 0.9523585438728333)
('lantus', 0.9522298574447632)
('humalog', 0.9518577456474304)
('electrolyte', 0.942333996295929)
('iron', 0.942203164100647)
('urinary tract', 0.94068843126297)
('ciprofloxacin', 0.9258120059967041)
('pcp', 0.9181954264640808)
```

```
for similar in model2.wv.similar_by_word('heart')[:20]:
    print(similar)
```

✓ 0.0s

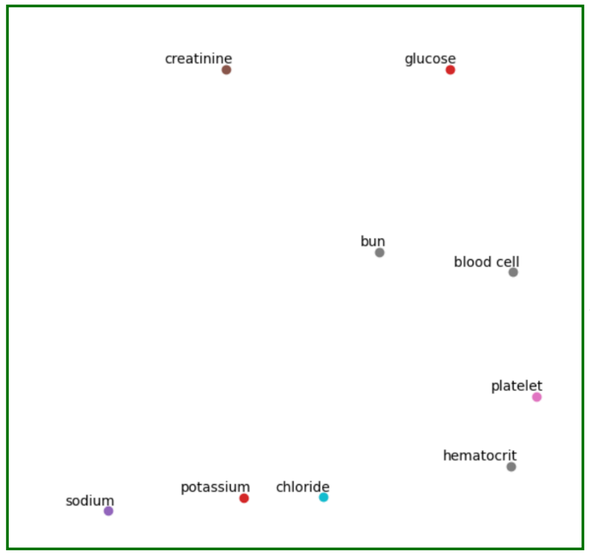
```
('dopamine', 0.985397219657898)
('cardiac', 0.9802132248878479)
('ETOH', 0.9747747182846069)
('alcohol', 0.9740748405456543)
('abdoman', 0.9735362529754639)
('Skin', 0.972573459148407)
('pleural', 0.9722948670387268)
('muscle', 0.9705457091331482)
('gen', 0.9697707891464233)
('pulmonary edema', 0.9697343111038208)
```

tSNE Plot

Used this code from class to make tSNE plots of my two models (shown on next two slides.)

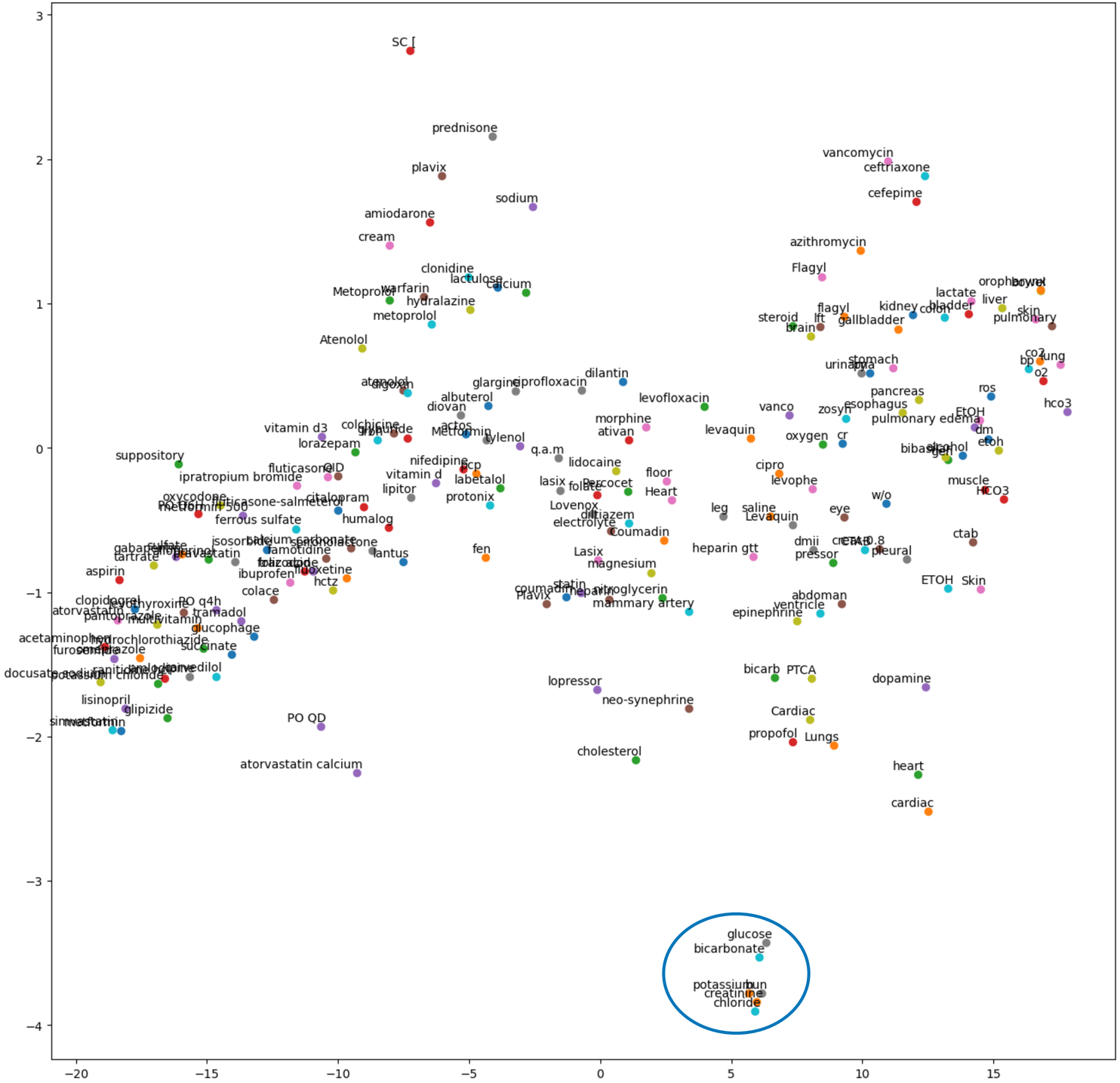
```
# Borrowed from Word2VECandtSNE.ipynb from AI 395 T, Module 6
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

def tsne_plot(model, words):
    "Creates and TSNE model and plots it"
    labels = words
    tokens = np.array([model.wv[word] for word in words])
    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 = zip(*new_values)

    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')
    # Zoom in to see this clump
    # plt.xlim(-2, 3)
    # plt.ylim(-10, -8)
    plt.show()

vocabs = model1.wv.key_to_index.keys()
new_v = np.array(list(vocabs))
tsne_plot(model1,new_v)

vocabs2 = model2.wv.key_to_index.keys()
new_v2 = np.array(list(vocabs2))
tsne_plot(model2, new_v2)
```

tSNE Plot of All entities



“brain” is literally inside of “head”!

tSNE Plot of Organ and Chemical Entities



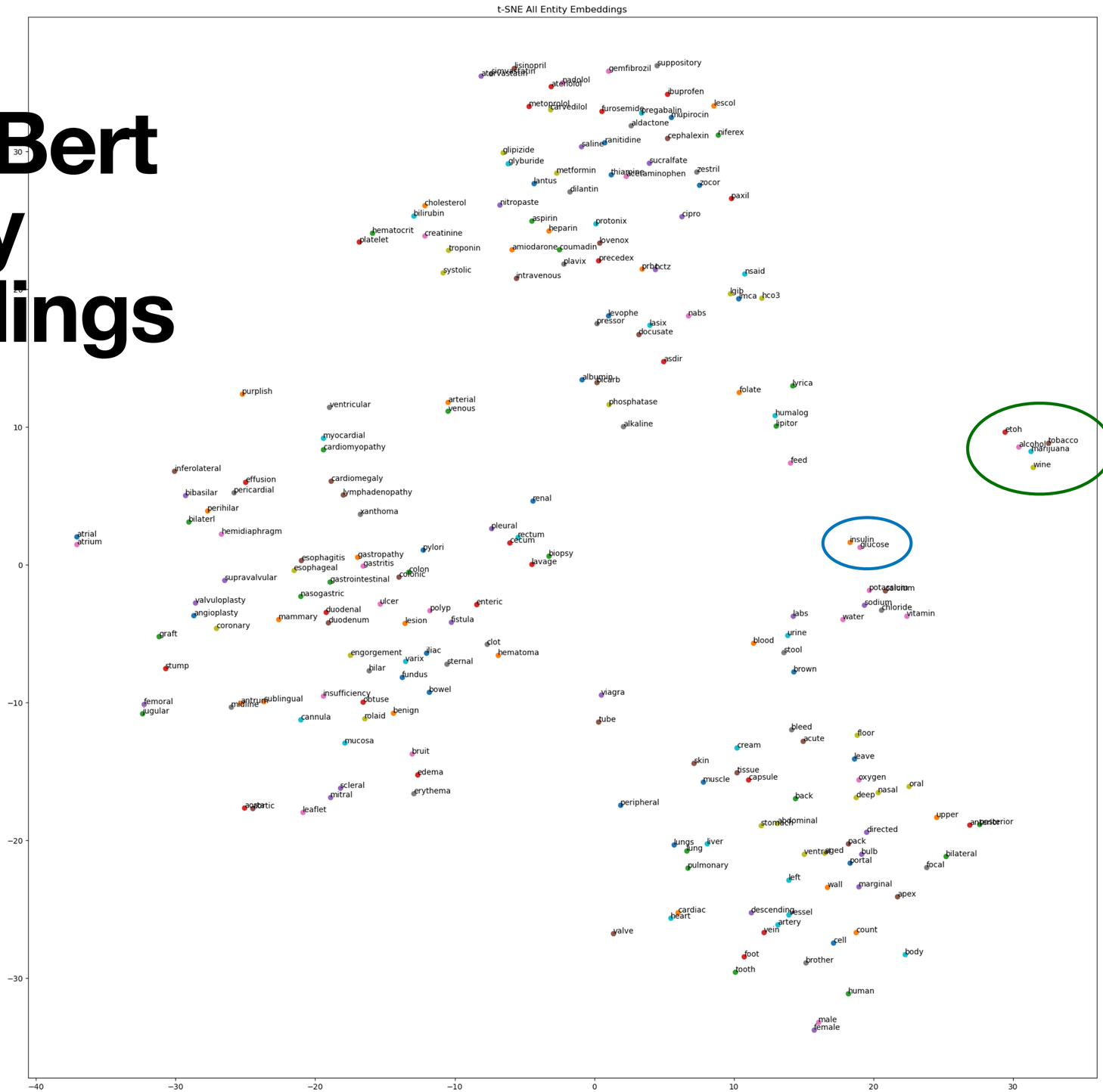
Clinical Bert

```python
# Borrowed from ClinicalBERT.ipynb from AI 395 T, Module 7
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 for plotting t-SNE plots
    cleaned_text = " ".join(dict.fromkeys(filtered_words))

    return cleaned_text
```

Used this code from class to make 2D scatter plots of the entity embeddings of my two sets (shown on next two slides.)

```python
# Visualization of notes filtered with SciSpacy using ClinicalBert
# Borrowed from ClinicalBERT.ipynb from AI 395 T, Module 7
import numpy as np
from sklearn.manifold import TSNE
import string
import matplotlib.pyplot as plt
from transformers import AutoModel, AutoTokenizer

# Visualization of notes filtered with SciSpacy using ClinicalBert
for a_corpus, title in [(corpus, 't-SNE All Entity Embeddings'),
                        (corpus_organ_chemical, 't-SNE Organ/Chemical Entity Embeddings')]:
    # 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()

    flat_corpus = []
    for note in a_corpus:
        flat_corpus.extend(note)
    notes_combined = ' '.join(flat_corpus)
    notes_combined = notes_combined[:5000]
    # 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(title)
    plt.show()
```

Clinical Bert All Entity Embeddings



This unsavory lot hangs out together. :D

Clinical Bert Organ and Chemical Embeddings

I circled a few that caught my eye.