MIMIC NLP TUTORIAL

CEP2763

Carter Patton

OVERALL PLAN

Game Plan

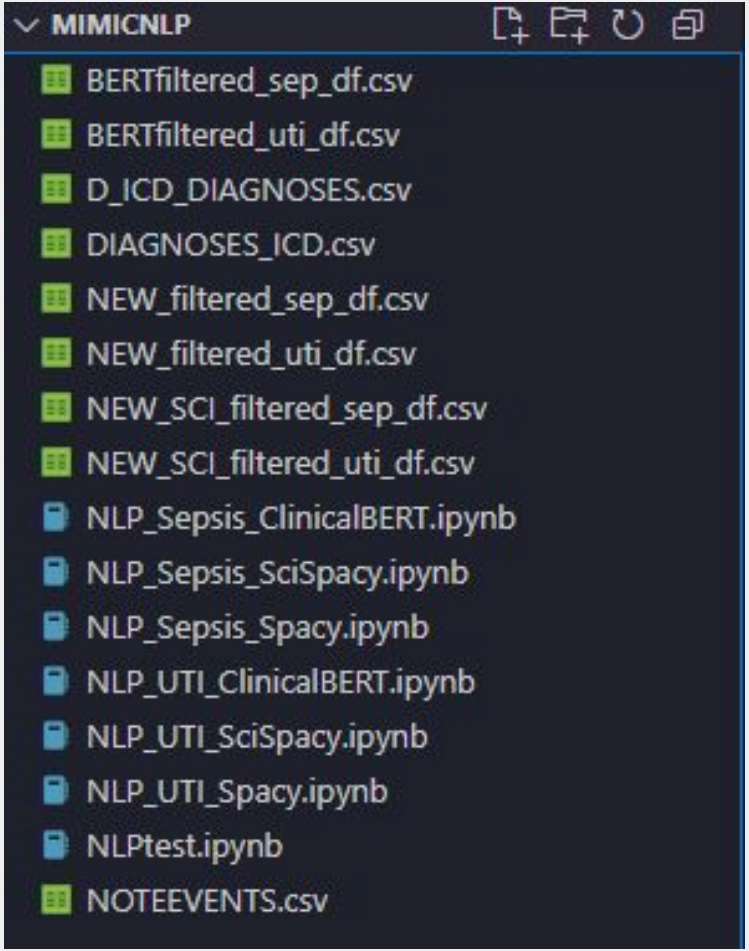
1. Extract notes & associated HADM_ID from NOTEVENTS.csv
2. For each HADM_ID, cross reference with diagnoses_ICD table and combine the tables on HADM_ID
3. Filter the data to include only rows that have a certain ICD code (i.e. the admissions that are pertaining to a certain disease, in this case, Urinary Tract Infection)
4. For each row in the resulting table, extract entities with Spacy
5. Embed the entities with Word2Vec
6. Create tSNE plot with these embeddings.
7. Repeat steps 4-6 with SciScapy instead of Scapy
8. Repeat steps 4-6 with ClinicalBERT instead of Scapy
9. Repeat steps 3-8 with a second disease ICD code (in this case, the ICD code for Severe Sepsis
10. Collect results of all 6 combinations, compare and contrast

CEP2763

File Structure & Rationale

I organized this project by creating a file for each of the 6 scenarios where each one is named by its NLP method and the disease it is examining. For example, NLP_SciSpacy_Sepsis.ipynb is the file that uses the SciSpacy NLP pipeline to extract entities from the data pertaining to Severe Sepsis, ICD code 99592. I laid my code base out this way so that I avoided misusing data that might have previously been transformed from other NLP pipelines. Each file contains a majority of the same code, with the main differences pertaining to the NLP pipeline and merging data based off of one of the two ICD9 codes I used to identify the 2 separate diseases I chose (UTI & Severe Sepsis). I chose these 2 diseases because they contained a large enough number of admissions that I could create a sizeable corpus, but did not contain so many admissions that the extraction process would take multiple hours on my local machine. I initially tested with Syphilis, which only had 23 admissions associated, and found that the corpus was too small to create meaningful embeddings. There is quite a bit of overlapping code between the 6 files, but from a development standpoint it helped me better visualize the slight tweaks I was making to optimize for each NLP pipeline

File Structure & Rationale

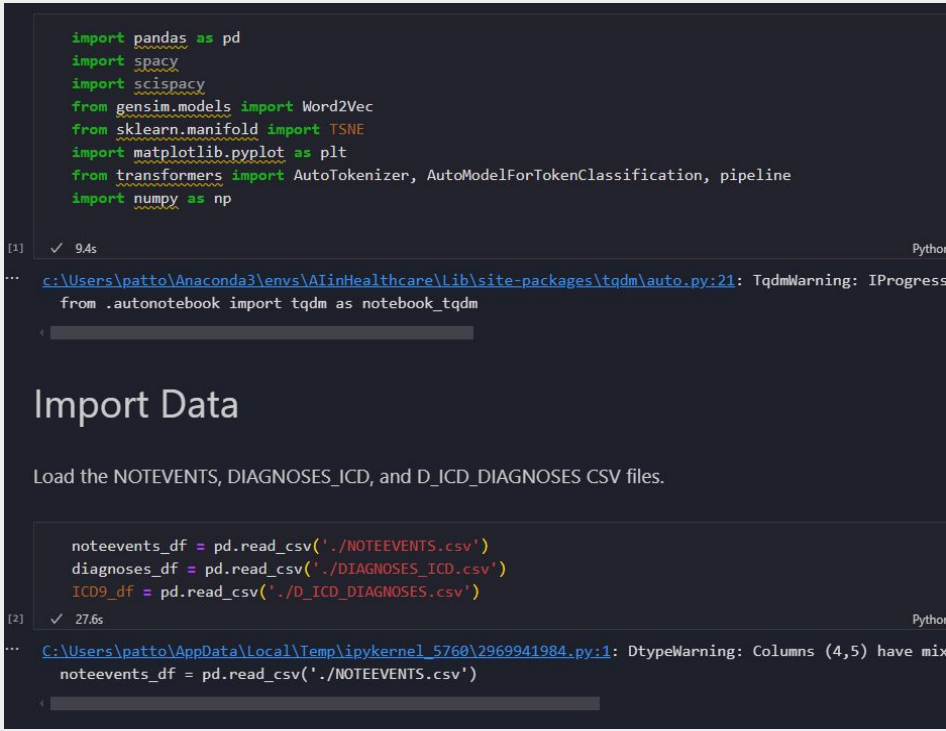


PROCESS

Load Desired Data

This code imports the necessary libraries for our entity extraction, data cleaning, embedding, and plotting.

It also loads the locally downloaded MIMIC III data to data frames for processing in proceeding code.



Load Desired Data

```
import pandas as pd
import spacy
import scispacy
from gensim.models import Word2Vec
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from transformers import AutoTokenizer, AutoModelForTokenClassification, pipeline
import numpy as np
```

[1] ✓ 9.4s Python

```
c:\Users\patto\Anaconda3\envs\AIinHealthcare\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress
  from .autonotebook import tqdm as notebook_tqdm
```

Import Data

Load the NOTEVENTS, DIAGNOSES_ICD, and D_ICD_DIAGNOSES CSV files.

```
noteevents_df = pd.read_csv('./NOTEEVENTS.csv')
diagnoses_df = pd.read_csv('./DIAGNOSES_ICD.csv')
ICD9_df = pd.read_csv('./D_ICD_DIAGNOSES.csv')
```

[2] ✓ 27.6s Python

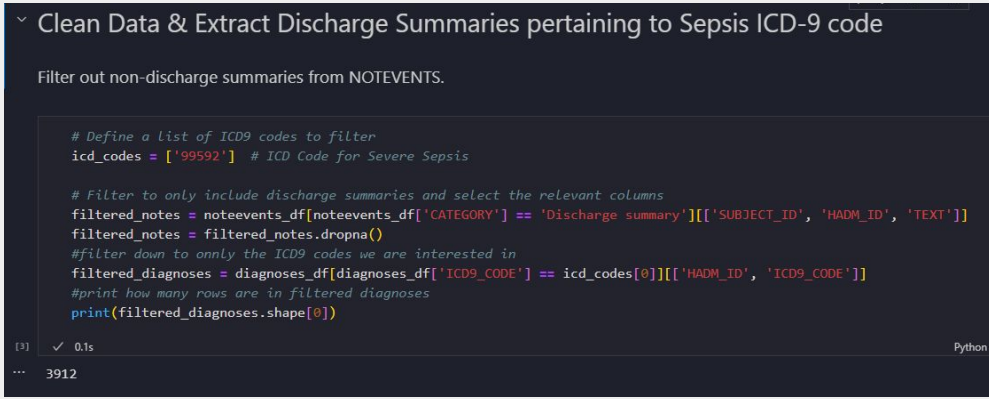
```
C:\Users\patto\AppData\Local\Temp\ipykernel_5760\2969941984.py:1: DtypeWarning: Columns (4,5) have mix
  noteevents_df = pd.read_csv('./NOTEEVENTS.csv')
```

Filter NOTEVENTS

This code grabs the ICD code for Severe Sepsis, 99592, which I got by querying the D_ICD_DIAGNOSES table in the MIMIC III Clinical dataset. The notevents_df is then filtered to only contain discharge summaries as TEXT, with rows missing data being dropped.

A filtered diagnoses df is then created by choosing only rows where the ICD9 code matched the selected row. I initially used icd_codes as a true list where I had multiple codes and iterated over each one, but this computation took a shockingly long time on my local PC so I stuck to just the 1 code for UTI & Severe Sepsis in their files respectively.

Between all 6 files the only difference in this block of code was the ICD code I selected for, 99592 here for Sepsis and 5990 in the UTI files for UTI diagnoses.



Filter NOTEEVENTS

Clean Data & Extract Discharge Summaries pertaining to Sepsis ICD-9 code

Filter out non-discharge summaries from NOTEVENTS.

```
# Define a list of ICD9 codes to filter
icd_codes = ['99592']  # ICD Code for Severe Sepsis

# Filter to only include discharge summaries and select the relevant columns
filtered_notes = noteevents_df[noteevents_df['CATEGORY'] == 'Discharge summary'][['SUBJECT_ID', 'HADM_ID', 'TEXT']]
filtered_notes = filtered_notes.dropna()
#filter down to onnly the ICD9 codes we are interested in
filtered_diagnoses = diagnoses_df[diagnoses_df['ICD9_CODE'] == icd_codes[0]][['HADM_ID', 'ICD9_CODE']]
#print how many rows are in filtered diagnoses
print(filtered_diagnoses.shape[0])
```

[3] ✓ 0.1s Python

```
3912
```

CEP2763

Filter NOTEVENTS

This code now merges the filtered diagnoses table and filtered noteevents table on the HADM_ID such that each noteevent can now be associated with the corresponding diagnosis from the admission.

This code is the same through all 6 files.

CEP2763

Filter NOTEEVENTS

Merge with Diagnoses Table

Merge the discharge summaries with the diagnoses data on `HADM_ID`

```python
# Merge the notes with diagnoses on HADM_ID
merged_df = filtered_notes.merge(filtered_diagnoses, on='HADM_ID', how='inner')
#remove duplicated rows
merged_df = merged_df.drop_duplicates(subset='HADM_ID')
print('Number of filtered records:', merged_df.shape[0])
print(merged_df.head())
```

✓ 0.0s Python

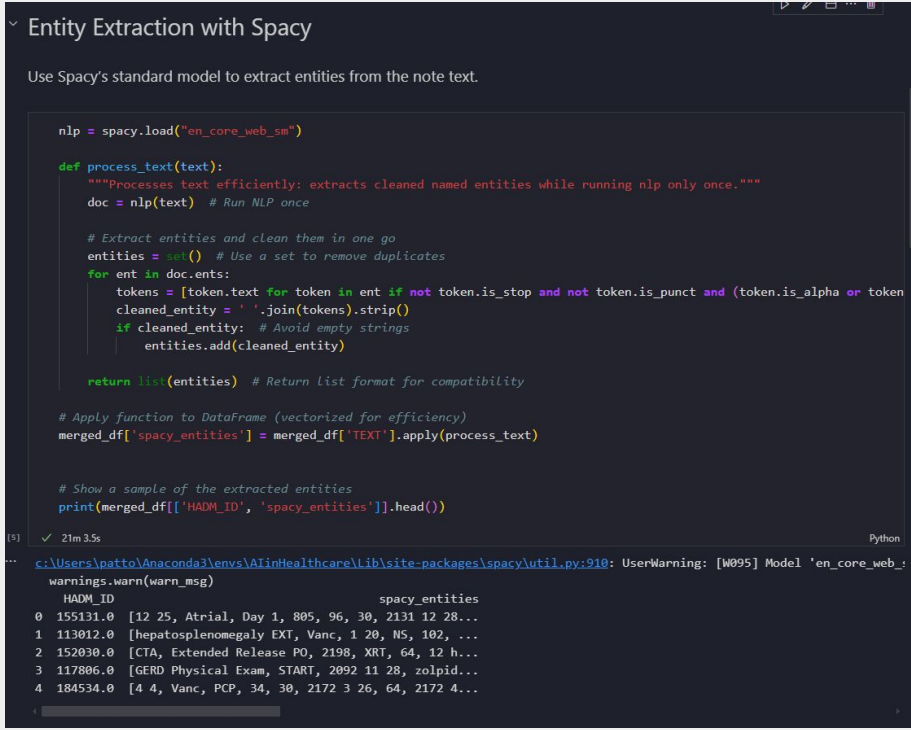
```
Number of filtered records: 3792
   SUBJECT_ID   HADM_ID                                              TEXT  \
0       26601  155131.0  Admission Date:  [**2131-12-23**]            ...
1       27051  113012.0  Admission Date:  [**2194-1-7**]             D...
2       75347  152030.0  Admission Date:  [**2174-5-3**]             D...
3       23224  117806.0  Admission Date:  [**2154-4-30**]            ...
4       19051  184534.0  Admission Date:  [**2172-3-26**]            ...

  ICD9_CODE
0     99592
1     99592
2     99592
3     99592
4     99592
```

Extract Entities from Note Text (Spacy)

This code uses Spacy's en)core_web_sm model for extraction of entities. It does this by creating a new column in the merged df called spacy_entities that applies the process_text function to each row's noteevent and subsequently stores it.

The process_text function runs nlp on each note, extracts entities from this long string, tokenizes them, makes sure the token is a digit or alpha to remove stop words and unnecessary tokens, joins them into a string, and adds it to a list that is returned. This is used for the 2 files that utilize the Spacy NLP pipeline

CEP2763



Extract Entities from Note Text (Spacy)

Entity Extraction with Spacy

Use Spacy's standard model to extract entities from the note text.

```
nlp = spacy.load("en_core_web_sm")

def process_text(text):
    """Processes text efficiently: extracts cleaned named entities while running nlp only once."""
    doc = nlp(text)  # Run NLP once

    # Extract entities and clean them in one go
    entities = set()  # Use a set to remove duplicates
    for ent in doc.ents:
        tokens = [token.text for token in ent if not token.is_stop and not token.is_punct and (token.is_alpha or token
        cleaned_entity = ' '.join(tokens).strip()
        if cleaned_entity:  # Avoid empty strings
            entities.add(cleaned_entity)

    return list(entities)  # Return list format for compatibility

# Apply function to DataFrame (vectorized for efficiency)
merged_df['spacy_entities'] = merged_df['TEXT'].apply(process_text)


# Show a sample of the extracted entities
print(merged_df[['HADM_ID', 'spacy_entities']].head())
```

[5] ✓ 21m 3.5s Python

```
c:\Users\patto\Anaconda3\envs\AIinHealthcare\Lib\site-packages\spacy\util.py:910: UserWarning: [W095] Model 'en_core_web_s
  warnings.warn(warn_msg)
    HADM_ID                                      spacy_entities
0  155131.0  [12 25, Atrial, Day 1, 805, 96, 30, 2131 12 28...
1  113012.0  [hepatosplenomegaly EXT, Vanc, 1 20, NS, 102, ...
2  152030.0  [CTA, Extended Release PO, 2198, XRT, 64, 12 h...
3  117806.0  [GERD Physical Exam, START, 2092 11 28, zolpid...
4  184534.0  [4 4, Vanc, PCP, 34, 30, 2172 3 26, 64, 2172 4...
```

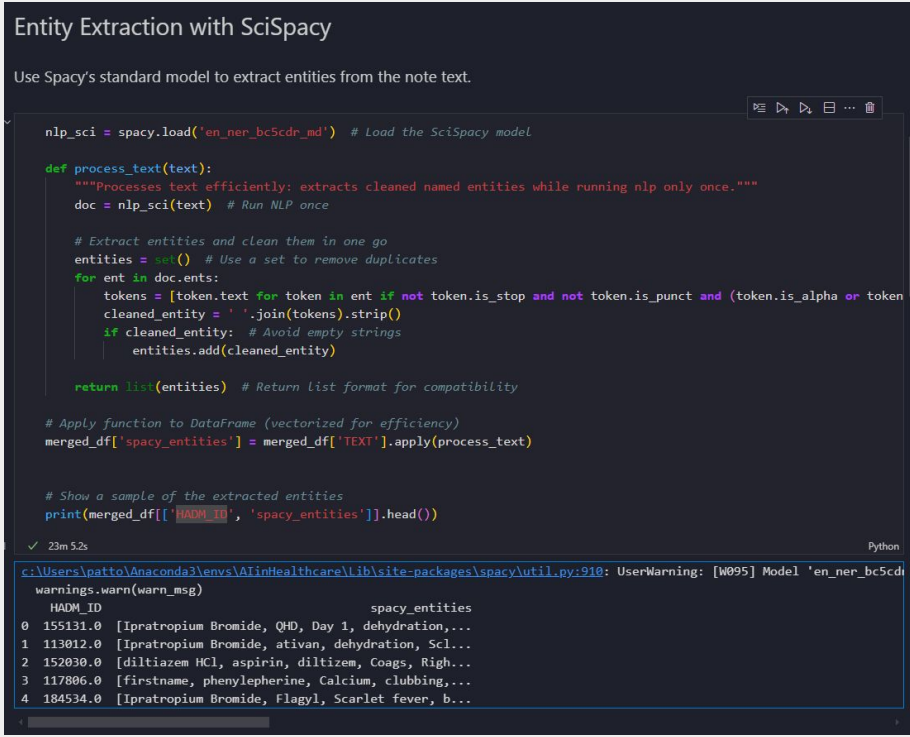
CEP2763

Extract Entities from Note Text (SciSpacy)

This code uses Sci_Spacy's `en_ner_bc5cdr_md` model for extraction of entities. It does this by using the same setup as I used with the Spacy model, but I instead utilize the nlp_sci pipeline which utilizes the aforementioned `en_ner_bc5cdr_md` model.

The process_text function runs nlp on each note, extracts entities from this long string, tokenizes them, makes sure the token is a digit or alpha to remove stop words and unnecessary tokens, joins them into a string, and adds it to a list that is returned. This is used for the 2 files that utilize the SciSpacy NLP pipeline

CEP2763



Extract Entities from Note Text (SciSpacy)

Entity Extraction with SciSpacy

Use Spacy's standard model to extract entities from the note text.

```python
nlp_sci = spacy.load('en_ner_bc5cdr_md')  # Load the SciSpacy model

def process_text(text):
    """Processes text efficiently: extracts cleaned named entities while running nlp only once."""
    doc = nlp_sci(text)  # Run NLP once

    # Extract entities and clean them in one go
    entities = set()  # Use a set to remove duplicates
    for ent in doc.ents:
        tokens = [token.text for token in ent if not token.is_stop and not token.is_punct and (token.is_alpha or token
        cleaned_entity = ' '.join(tokens).strip()
        if cleaned_entity:  # Avoid empty strings
            entities.add(cleaned_entity)

    return list(entities)  # Return list format for compatibility

# Apply function to DataFrame (vectorized for efficiency)
merged_df['spacy_entities'] = merged_df['TEXT'].apply(process_text)


# Show a sample of the extracted entities
print(merged_df[['HADM_ID', 'spacy_entities']].head())
```

✓ 23m 5.2s Python

```
c:\Users\patto\Anaconda3\envs\AIinHealthcare\Lib\site-packages\spacy\util.py:910: UserWarning: [W095] Model 'en_ner_bc5cd
  warnings.warn(warn_msg)
    HADM_ID                                    spacy_entities
0  155131.0  [Ipratropium Bromide, QHD, Day 1, dehydration,...
1  113012.0  [Ipratropium Bromide, ativan, dehydration, Scl...
2  152030.0  [diltiazem HCl, aspirin, diltizem, Coags, Righ...
3  117806.0  [firstname, phenylepherine, Calcium, clubbing,...
4  184534.0  [Ipratropium Bromide, Flagyl, Scarlet fever, b...
```

CEP2763

Extract Entities from Note Text (ClinicalBERT)

This code uses ClinicalBERT's standard model for extraction of entities. The process text function is less stringent that SciSpacy and Spacy in that it does not strip tokens, but instead chunks the text since ClinicalBERT's max window size is 512 tokens. I utilize the ClinicalBERT AutoTokenizer in order to do this chunking, and then the process of appending these combined tokens to a set and then returning them in list form is very similar to SciSpacy & Spacy files. This is used for the 2 files that utilize the ClinicalBERT NLP pipeline.

CEP2763

Extract Entities from Note Text (ClinicalBERT)

Entity Extraction with ClinicalBERT

Use Spacy's standard model to extract entities from the note text.

```python
# Load ClinicalBERT tokenizer & model
tokenizer = AutoTokenizer.from_pretrained('emilyalsentzer/Bio_ClinicalBERT')
model = AutoModelForTokenClassification.from_pretrained('emilyalsentzer/Bio_ClinicalBERT')
nlp_bert = pipeline('ner', model=model, tokenizer=tokenizer)

def chunk_text(text, tokenizer, max_length=500, overlap=50):
    """Splits text into overlapping chunks using ClinicalBERT's tokenizer."""
    tokens = tokenizer.encode(text, add_special_tokens=False)  # Convert text to token IDs
    chunks = []
    i = 0
    while i < len(tokens):
        chunk = tokens[i : i + max_length]  # Extract a chunk of tokens
        chunks.append(tokenizer.decode(chunk))  # Convert back to text
        i += max_length - overlap  # Move forward with overlap
    return chunks

def process_text(text):
    """Processes text in chunks for ClinicalBERT."""
    chunks = chunk_text(text, tokenizer)  # Use ClinicalBERT tokenizer to chunk text
    entities = set()

    for chunk in chunks:
        results = nlp_bert(chunk)  # Run ClinicalBERT NER on chunk
        for ent in results:  # `results` is a LIST of dictionaries
            entities.add(ent['word'])  # Extract named entity correctly

    return list(entities)  # Return list format for compatibility

# Apply to DataFrame
merged_df['spacy_entities'] = merged_df['TEXT'].apply(process_text)
```

[5] ✓ 14m 58.0s Python

```
Some weights of BertForTokenClassification were not initialized from the model checkpoint at emilyalsentzer/Bio_ClinicalBI
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Device set to use cuda:0
Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
```

CEP2763

Create Embeddings from Extracted Entities (SciSpacy & Spacy)

This code creates a corpus from the spacy_entities column in merged_df. It does so by iterating over each row in this column, joining the list, extracting the entities again through an NLP pipeline, and appending them to the list that is the corpus. The corpus is then flattened and passed to the Word2Vec function to create an embeddings model which was tuned with custom parameters via trial and error based on the resulting tSNE plot.

Word2Vec Embeddings for Spacy Entities

Train a Word2Vec model using the Spacy-extracted entities and compute embeddings.

```
uti_corpus = []
for row in range(0, len(merged_df)):
    ent_str = []
    #combine all the entities into one string
    string = ' '.join(merged_df.iloc[row]['spacy_entities'])
    ents = nlp_sci(string).ents
    for i in range(0, len(ents)):
        ent_str.append(ents[i].text)
    uti_corpus.append(list(ent_str))

#flatten corpus
flat_corpus = [word for sublist in uti_corpus for word in sublist]

# Train Word2Vec on the Spacy entities
model_scispacy = Word2Vec(uti_corpus, min_count=40, vector_size=200, window=10, workers=4)

#print(merged_df[['HADM_ID', 'spacy_embeddings']].head())
```

Create Embeddings from Extracted Entities (SciSpacy & Spacy)

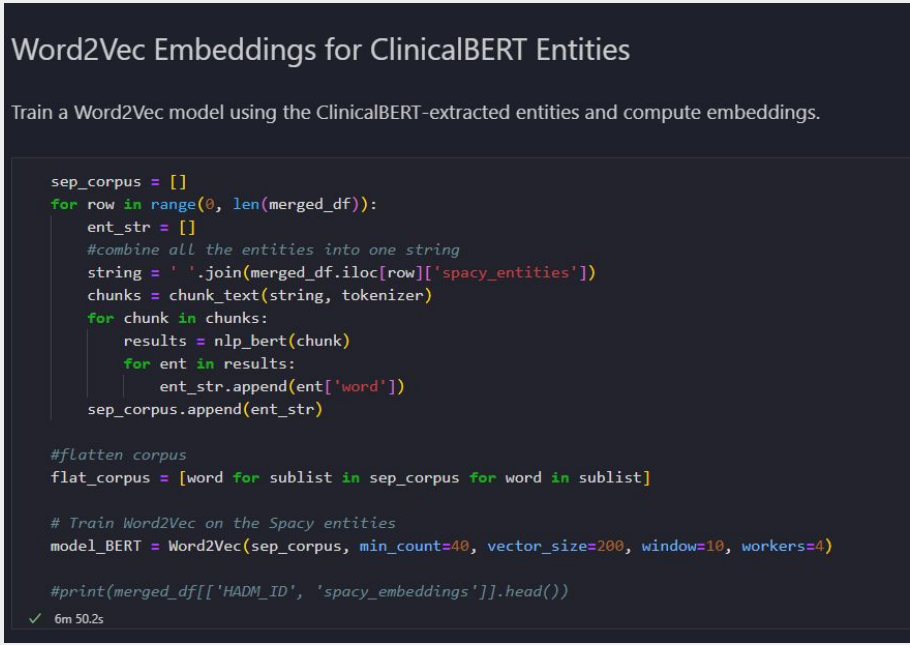
Word2Vec Embeddings for Spacy Entities

Train a Word2Vec model using the Spacy-extracted entities and compute embeddings.

```
uti_corpus = []
for row in range(0, len(merged_df)):
    ent_str = []
    #combine all the entities into one string
    string = ' '.join(merged_df.iloc[row]['spacy_entities'])
    ents = nlp_sci(string).ents
    for i in range(0, len(ents)):
        ent_str.append(ents[i].text)
    uti_corpus.append(list(ent_str))

#flatten corpus
flat_corpus = [word for sublist in uti_corpus for word in sublist]

# Train Word2Vec on the Spacy entities
model_scispacy = Word2Vec(uti_corpus, min_count=40, vector_size=200, window=10, workers=4)

#print(merged_df[['HADM_ID', 'spacy_embeddings']].head())
```

Create Embeddings from Extracted Entities (ClinicalBERT)

This code creates a corpus from the spacy_entities column which holds the entities extracted by ClinicalBERT in merged_df. It does so by iterating over each row in this column, joining the list, chunking the entities so they fit in the ClinicalBERT window, extracting the entities again through an NLP pipeline, and appending them to the list that is the corpus. The corpus is then flattened and passed to the Word2Vec function to create an embeddings model which was tuned with custom parameters via trial and error based on the resulting tSNE plot.



CEP2763

Create Embeddings from Extracted Entities (ClinicalBERT)

Word2Vec Embeddings for ClinicalBERT Entities

Train a Word2Vec model using the ClinicalBERT-extracted entities and compute embeddings.

```
sep_corpus = []
for row in range(0, len(merged_df)):
    ent_str = []
    #combine all the entities into one string
    string = ' '.join(merged_df.iloc[row]['spacy_entities'])
    chunks = chunk_text(string, tokenizer)
    for chunk in chunks:
        results = nlp_bert(chunk)
        for ent in results:
            ent_str.append(ent['word'])
    sep_corpus.append(ent_str)

#flatten corpus
flat_corpus = [word for sublist in sep_corpus for word in sublist]

# Train Word2Vec on the Spacy entities
model_BERT = Word2Vec(sep_corpus, min_count=40, vector_size=200, window=10, workers=4)

#print(merged_df[['HADM_ID', 'spacy_embeddings']].head())
```

✓ 6m 50.2s

Create tSNE Plots for Embeddings

This code creates a tSNE plot from the Word2Vec model we created for each pipeline. This code is the same for all 6 files except the model name is different to call the correct Word2Vec model. This plot is made based on the code from the lectures. The calling of this function loads the vocabs from the model and converts them to an numpy array so that this function can properly plot them with the model created from the flattened corpus.

t-SNE Visualization for SciSpacy Embeddings

Flatten the embeddings and use t-SNE to visualize them.

```
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()
```

✓ 0.0s Python

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

✓ 1.5s

Create tSNE Plots for Embeddings

t-SNE Visualization for SciSpacy Embeddings

Flatten the embeddings and use t-SNE to visualize them.

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

✓ 1.5s

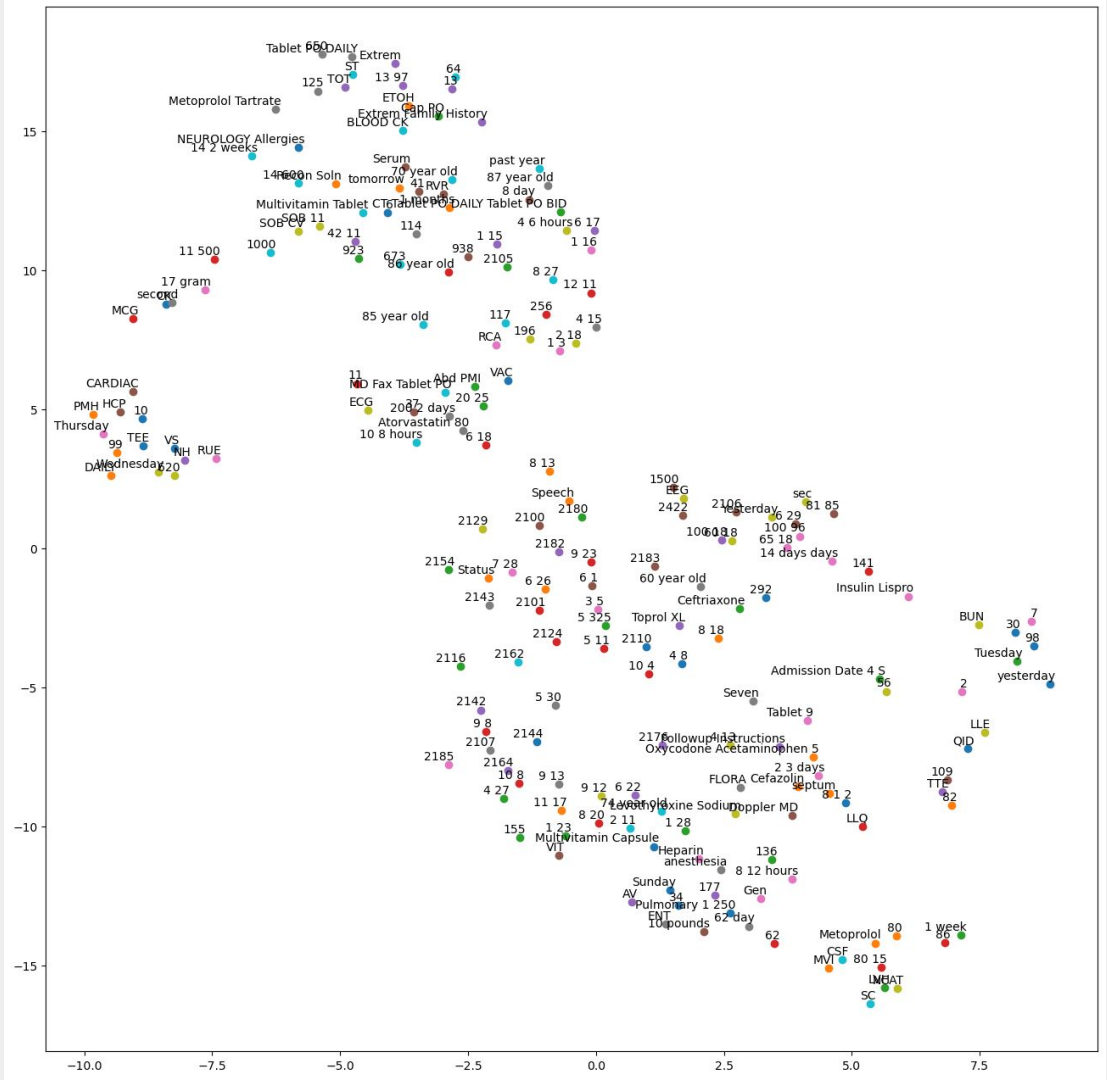
```
def tsne_plot(model,words, preTrained=False):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    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()
```

✓ 0.0s Python

RESULTS: Disease 1 Urinary Tract Infection (UTI)

tSNE plot for Spacy

UTI

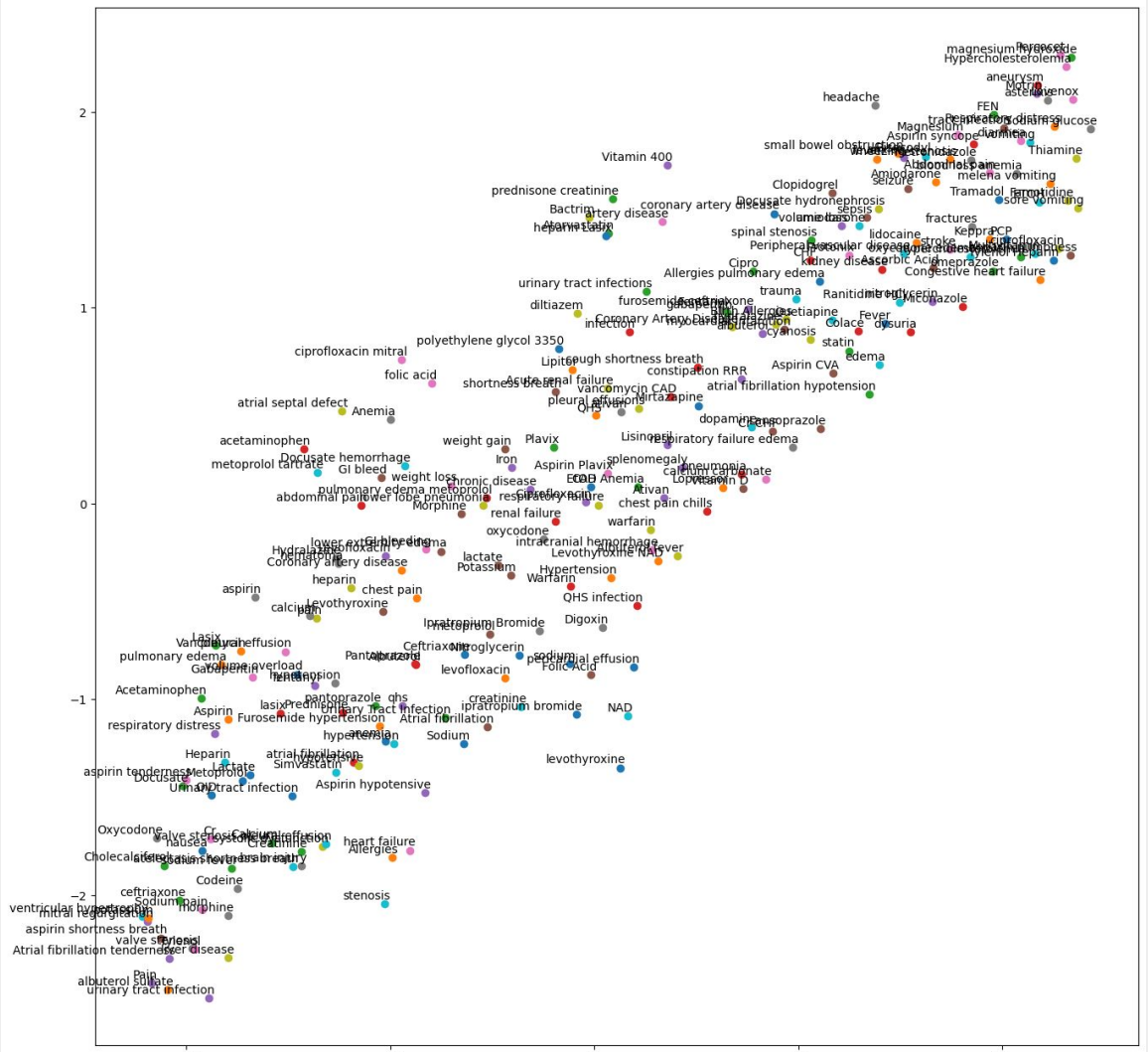


CEP2763

MIMIC NLP TUTORIAL

tSNE plot for SciSpacy

UTI

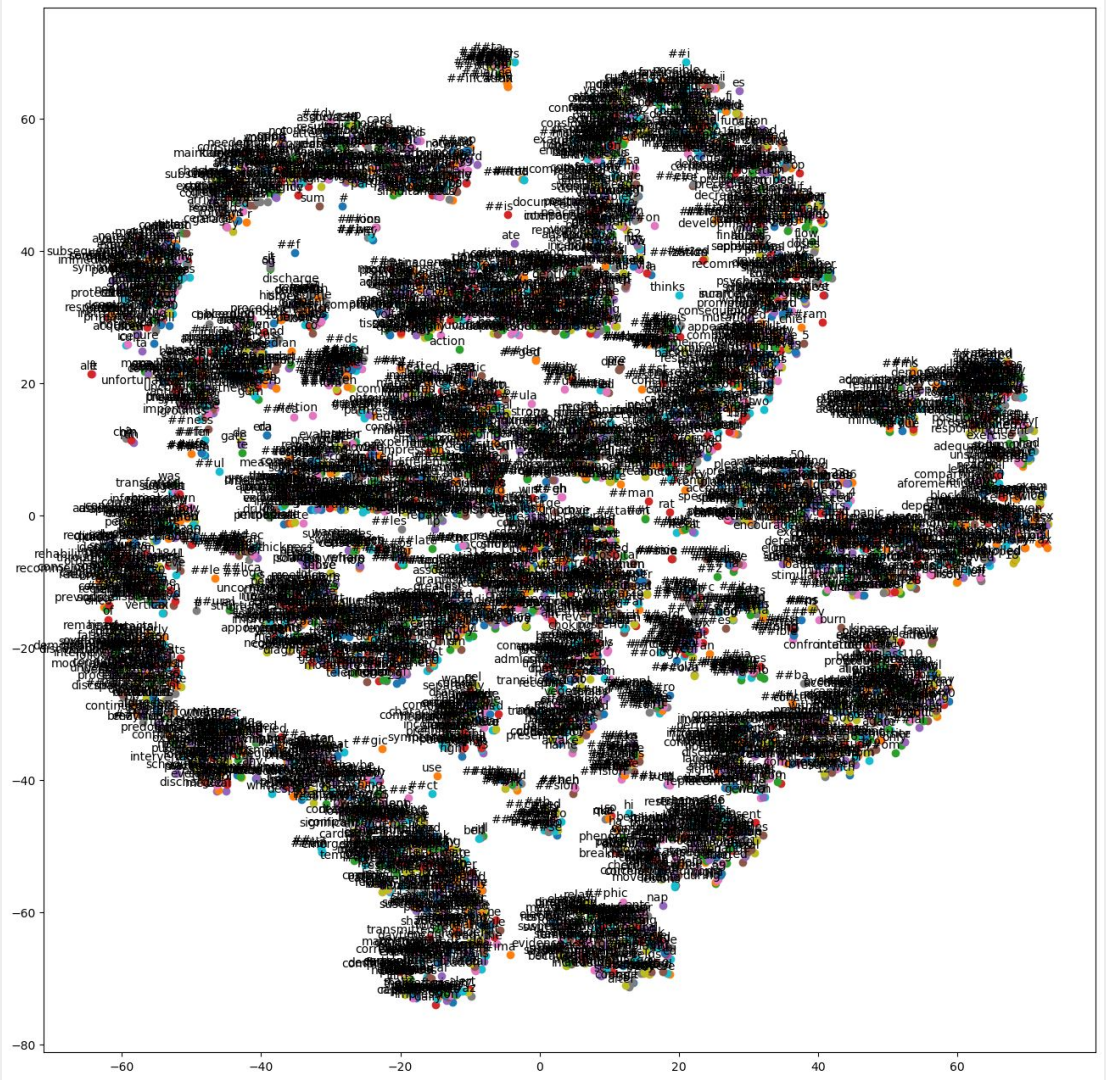


MIMIC NLP TUTORIAL

CEP2763

tSNE plot for Clinical BERT

UTI

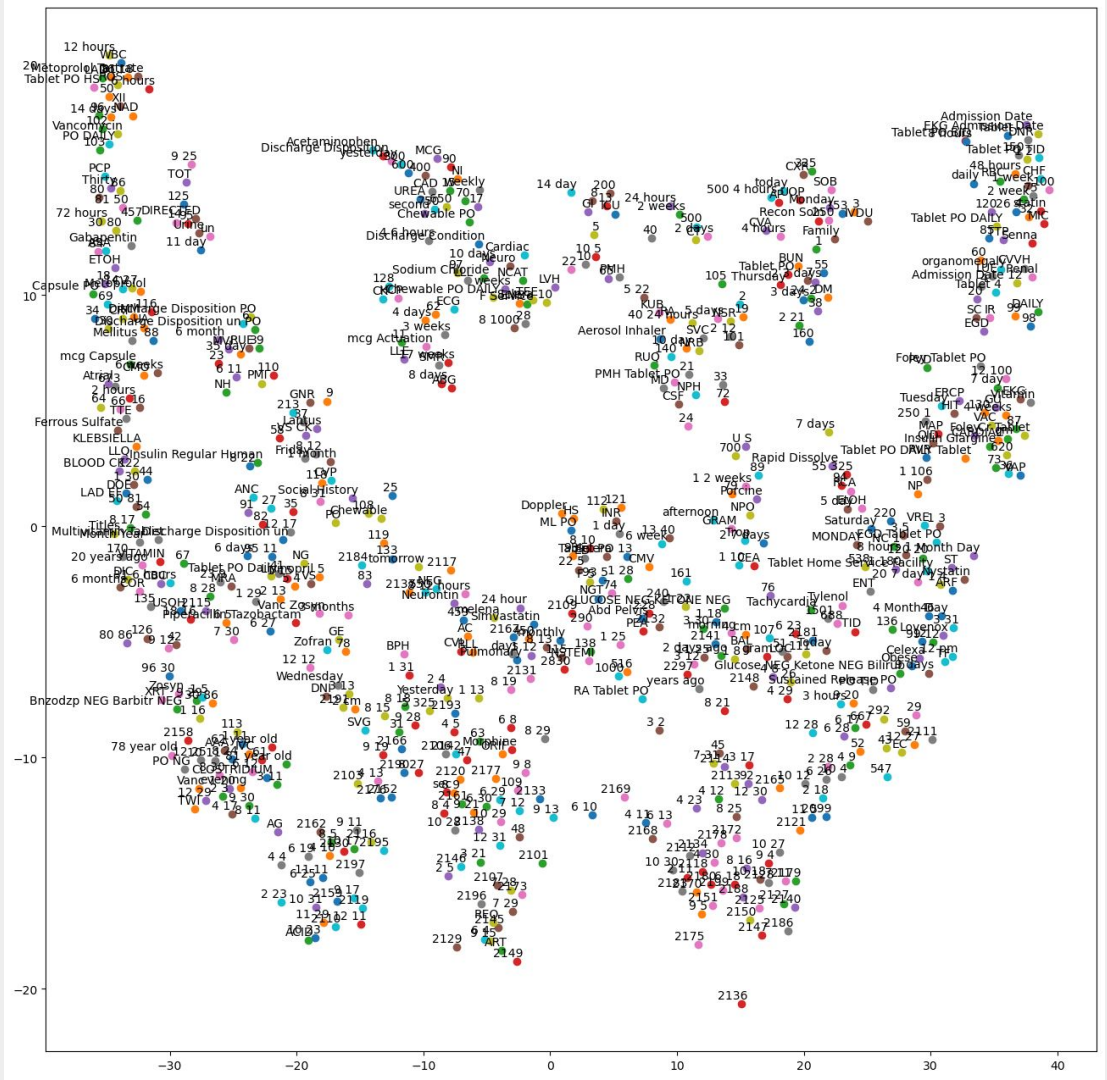


CEP2763

RESULTS: Disease 2 Sepsis

tSNE plot for Spacy

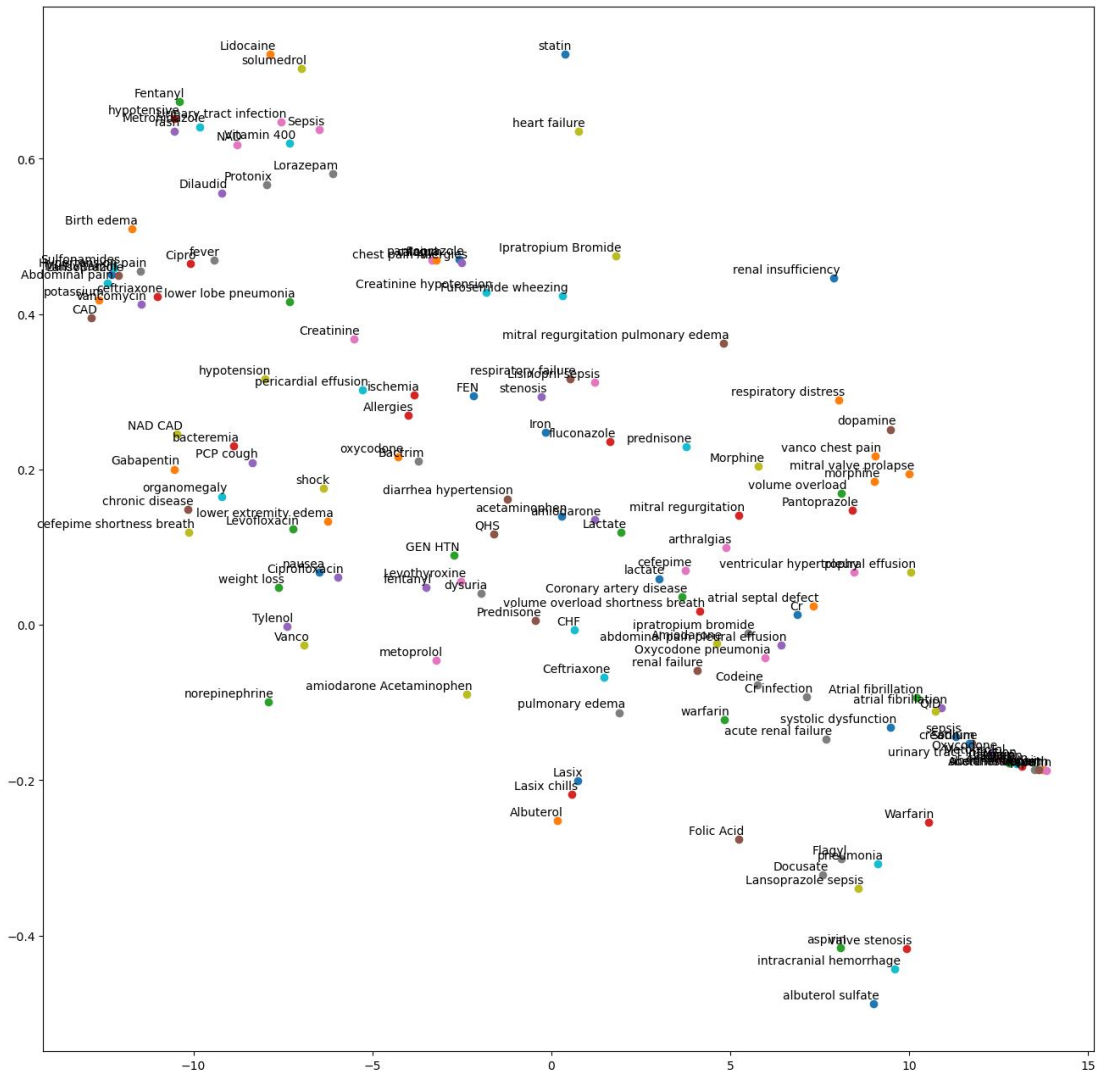
Severe Sepsis



CEP2763

tSNE plot for SciSpacy

Severe Sepsis

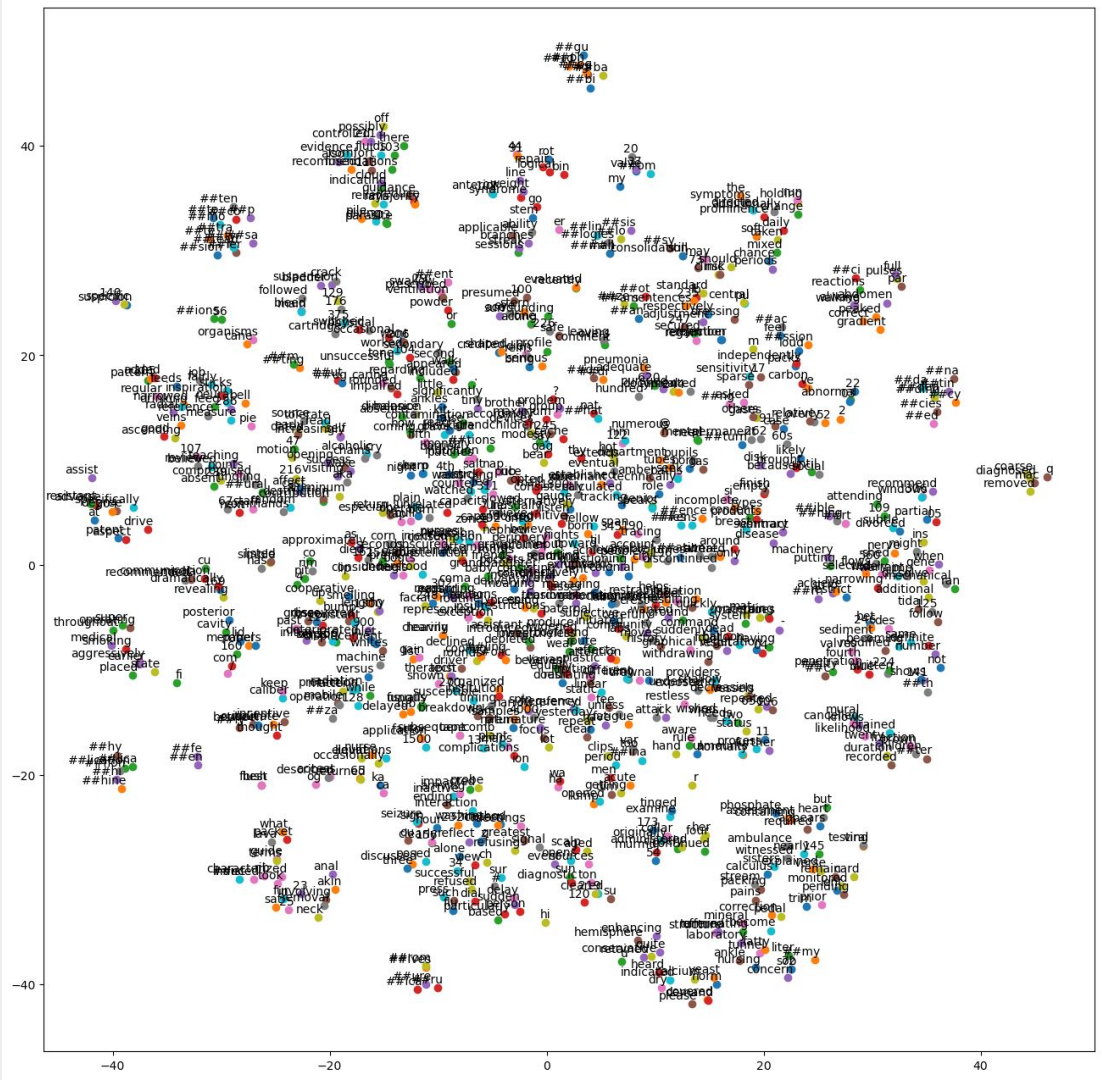


CEP2763

MIMIC NLP TUTORIAL

tSNE plot for Clinical BERT

Severe Sepsis



CEP2763

tSNE plot for Clinical BERT (full corpus)

Severe Sepsis