NLP MIMIC III

MAZIYAR MIRZAEI

AI IN HEALTHCARE

FALL 2024 UT AUSTIN

ANALYZING DIABETES PATIENT NOTES WITH NLP TECHNIQUES

- **Objective:** To process clinical notes of diabetes patients from the MIMIC-III dataset using NLP methods such as SpaCy, SciSpaCy, Word2Vec, and BERT models for entity recognition and embedding visualization.

DATA LOADING

- Loading Clinical Notes and Diagnoses Data
- **Explanation:** Load the first 50,000 rows of the NOTEEVENTS.csv file and the entire DIAGNOSES_ICD.csv file into pandas DataFrames for processing.

```
import pandas as pd

# Load the data
data = pd.read_csv('NOTEEVENTS.csv', nrows=50000)

# Load diagnoses data
diagnoses = pd.read_csv('DIAGNOSES_ICD.csv')
```

FILTERING FOR DIABETES DIAGNOSES

Title: Identifying Diabetes Patients

- Explanation: Filter the diagnoses to include only those with ICD-9 codes corresponding to Diabetes Mellitus.

```python
# Load diagnoses data
diagnoses = pd.read_csv('DIAGNOSES_ICD.csv')

# Filter for Diabetes Mellitus ICD-9 codes
diabetes_codes = [f'250{str(i).zfill(2)}' for i in range(0, 100)]
diabetes_diagnoses = diagnoses[diagnoses['ICD9_CODE'].isin(diabetes_codes)].copy()

# Create a unique identifier for SUBJECT_ID and HADM_ID
diabetes_diagnoses.loc[:, 'SUBJ_HADM_ID'] = diabetes_diagnoses['SUBJECT_ID'].astype(str) + '_' +
diabetes_diagnoses['HADM_ID'].astype(str)

# Similarly, prepare the notes data
data['SUBJ_HADM_ID'] = data['SUBJECT_ID'].astype(str) + '_' + data['HADM_ID'].astype(str)

# Merge the notes with the Diabetes diagnoses
diabetes_notes = data[data['SUBJ_HADM_ID'].isin(diabetes_diagnoses['SUBJ_HADM_ID'])].copy()

# Reset index
diabetes_notes = diabetes_notes.reset_index(drop=True)
```

MERGING NOTES WITH DIAGNOSES

Title: Merging Clinical Notes with Diabetes Diagnoses

- **Explanation:** Create a unique identifier by combining SUBJECT_ID and HADM_ID, then merge the notes with the diabetes diagnoses based on this identifier.

```python
# Create a unique identifier for SUBJECT_ID and HADM_ID
diabetes_diagnoses['SUBJ_HADM_ID'] = diabetes_diagnoses['SUBJECT_ID'].astype(str) + '_' + diabetes_diagnoses['HADM_ID'].astype(str)

# Similarly, prepare the notes data
data['SUBJ_HADM_ID'] = data['SUBJECT_ID'].astype(str) + '_' + data['HADM_ID'].astype(str)

# Merge the notes with the Diabetes diagnoses
diabetes_notes = data[data['SUBJ_HADM_ID'].isin(diabetes_diagnoses['SUBJ_HADM_ID'])]

# Reset index
diabetes_notes = diabetes_notes.reset_index(drop=True)
```

DATA CLEANING

Title: Cleaning Clinical Notes Text

- **Explanation:** Remove null entries in the TEXT column and clean the text by removing de-identification brackets and extra whitespace.

```python
import re

# Remove null TEXT entries
diabetes_notes = diabetes_notes.dropna(subset=['TEXT'])

# Function to clean text
def clean_text(text):
    text = re.sub(r'\[\*\*.*?\*\*\]', '', text)  # Remove de-identification brackets
    text = re.sub(r'\s+', ' ', text)  # Remove extra whitespace
    return text.strip()

diabetes_notes['CLEAN_TEXT'] = diabetes_notes['TEXT'].apply(clean_text)
```

LOADING NLP MODELS

Title: Loading SpaCy and SciSpaCy Models

- **Explanation:** Load the necessary SpaCy and SciSpaCy models for NLP processing, including the biomedical NER model.

```
import spacy

# Load SpaCy's English model
nlp_spacy = spacy.load('en_core_web_sm')

# Load SciSpaCy's models
nlp_scispacy = spacy.load('en_core_sci_sm')

# Load SciSpaCy's NER model for biomedical data
nlp_scispacy_ner = spacy.load('en_ner_bc5cdr_md')
```

ENTITY RECOGNITION WITH SPACY

Title: Extracting Entities with SpaCy

- **Explanation:** Use SpaCy's English model to extract entities from a sample clinical note.

```
doc_spacy =
nlp_spacy(diabetes_notes['CLEAN_TEXT'][0])
print("SpaCy Entities:")
for ent in doc_spacy.ents:
    print(f"{ent.text} - {ent.label_}")
```

Output:
List of entities identified by SpaCy with their labels (e.g., Zocor / Lescol Attending - ORG).

```
SpaCy Entities:
Zocor / Lescol Attending - ORG
History of Present Illness -
WORK_OF_ART
84 - CARDINAL
1 - CARDINAL
28 - CARDINAL
25-30% - PERCENT
CAD - ORG
s/p CABG - ORG
SVG - ORG
LIMA - GPE
SVG-OM - ORG
...
```

ENTITY RECOGNITION WITH SCISPACY

Title: Extracting Entities with SciSpaCy

- **Explanation:** Use SciSpaCy's model to extract scientific and biomedical entities from the same clinical note.

```
doc_scispacy =
nlp_scispacy(diabetes_notes['CLEAN_TEXT'][0])
print("\nSciSpaCy Entities:") for ent in
doc_scispacy.ents: print(f"{ent.text} -
{ent.label_}")
```

Output:

•List of entities identified by SciSpaCy with the label ENTITY.

```
SciSpaCy Entities:
Admission - ENTITY
Discharge Date - ENTITY
Service - ENTITY
MEDICINE Allergies - ENTITY
Zocor - ENTITY
Lescol - ENTITY
Attending - ENTITY
Chief Complaint - ENTITY
Chest pain - ENTITY
Surgical - ENTITY
Invasive Procedure - ENTITY
Central venous line insertion - ENTITY
right internal jugular vein) History of Present -
ENTITY
Illness - ENTITY
Mr. - ENTITY
man – ENTITY
...
```

BIOMEDICAL NER WITH SCISPACY

Title: Extracting Biomedical Entities with SciSpaCy NER

- **Explanation:** Use SciSpaCy's biomedical NER model to extract chemical and disease entities.

```
doc_scispacy_ner =
nlp_scispacy_ner(diabetes_notes['CLEAN_TEXT'][0
])
print("\nSciSpaCy NER Entities:")
for ent in doc_scispacy_ner.ents:
    print(f"{ent.text} - {ent.label_}")
```

Output:

- List of entities with labels CHEMICAL and DISEASE (e.g., Zocor - CHEMICAL, chest pain - DISEASE).

```
SciSpaCy NER Entities:
Allergies - DISEASE
Zocor - CHEMICAL
Lescol - CHEMICAL
Chest pain - DISEASE
stenosis - DISEASE
mitral regurgitation - DISEASE
aortic insufficiency - DISEASE
chronic left ventricular systolic heart failure - DISEASE
hypertension - DISEASE
hyperlipidemia - DISEASE
diabetes mellitus - DISEASE
CAD - DISEASE
SVG-OM - CHEMICAL
SVG-OM - CHEMICAL
peripheral arterial disease - DISEASE
...
```

VISUALIZING ENTITIES

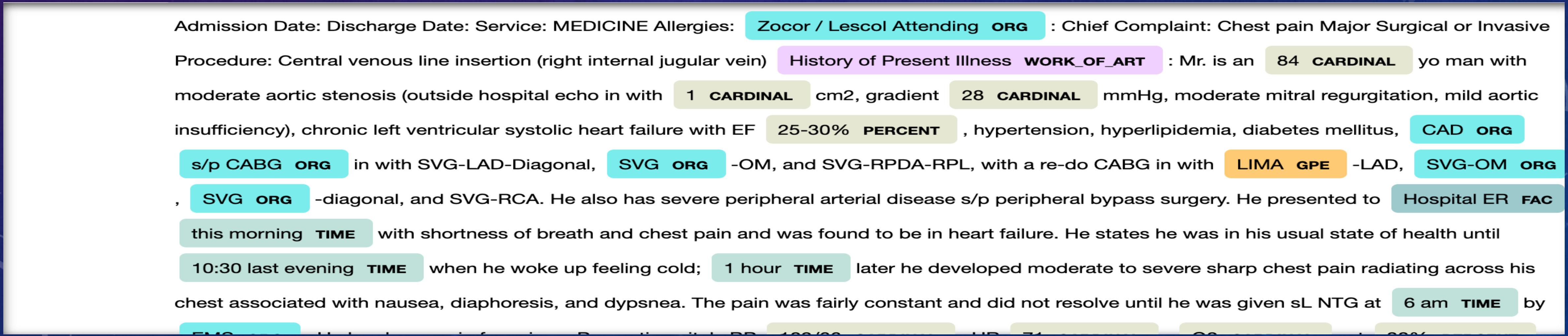
Title: Visualizing Entities with displaCy

- **Explanation:** Use SpaCy's displaCy visualizer to display entities in the text.

Visualization:

- Annotated text highlighting entities such as medications, diseases, and procedures.

```python
from spacy import displacy

# SpaCy Visualization
displacy.render(doc_spacy, style="ent", jupyter=True)

# SciSpaCy Visualization
displacy.render(doc_scispacy_ner, style="ent", jupyter=True)
```



Admission Date: Discharge Date: Service: MEDICINE Allergies: Zocor / Lescol Attending ORG : Chief Complaint: Chest pain Major Surgical or Invasive Procedure: Central venous line insertion (right internal jugular vein) History of Present Illness WORK_OF_ART : Mr. is an 84 CARDINAL yo man with moderate aortic stenosis (outside hospital echo in with 1 CARDINAL cm2, gradient 28 CARDINAL mmHg, moderate mitral regurgitation, mild aortic insufficiency), chronic left ventricular systolic heart failure with EF 25-30% PERCENT , hypertension, hyperlipidemia, diabetes mellitus, CAD ORG s/p CABG ORG in with SVG-LAD-Diagonal, SVG ORG -OM, and SVG-RPDA-RPL, with a re-do CABG in with LIMA GPE -LAD, SVG-OM ORG , SVG ORG -diagonal, and SVG-RCA. He also has severe peripheral arterial disease s/p peripheral bypass surgery. He presented to Hospital ER FAC this morning TIME with shortness of breath and chest pain and was found to be in heart failure. He states he was in his usual state of health until 10:30 last evening TIME when he woke up feeling cold; 1 hour TIME later he developed moderate to severe sharp chest pain radiating across his chest associated with nausea, diaphoresis, and dypsnea. The pain was fairly constant and did not resolve until he was given sL NTG at 6 am TIME by

PREPROCESSING TEXT FOR WORD2VEC

Title: Preprocessing Text Data

- **Explanation:** Tokenize and preprocess the cleaned text for Word2Vec modeling, removing stopwords and short tokens.

```python
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS

def preprocess(text):
    return [token for token in simple_preprocess(text) if token not in STOPWORDS and len(token) > 3]

documents = diabetes_notes['CLEAN_TEXT'].map(preprocess)
```

TRAINING WORD2VEC MODEL

Title: Training a Word2Vec Model on Clinical Notes

- **Explanation:** Train a Word2Vec model using the preprocessed documents to learn word embeddings.

```
from gensim.models import Word2Vec

model = Word2Vec(sentences=documents,
vector_size=100, window=5, min_count=2, workers=4)

# Most similar words to 'insulin'
print("Words similar to 'insulin':")
print(model.wv.most_similar('insulin'))
```

Output:

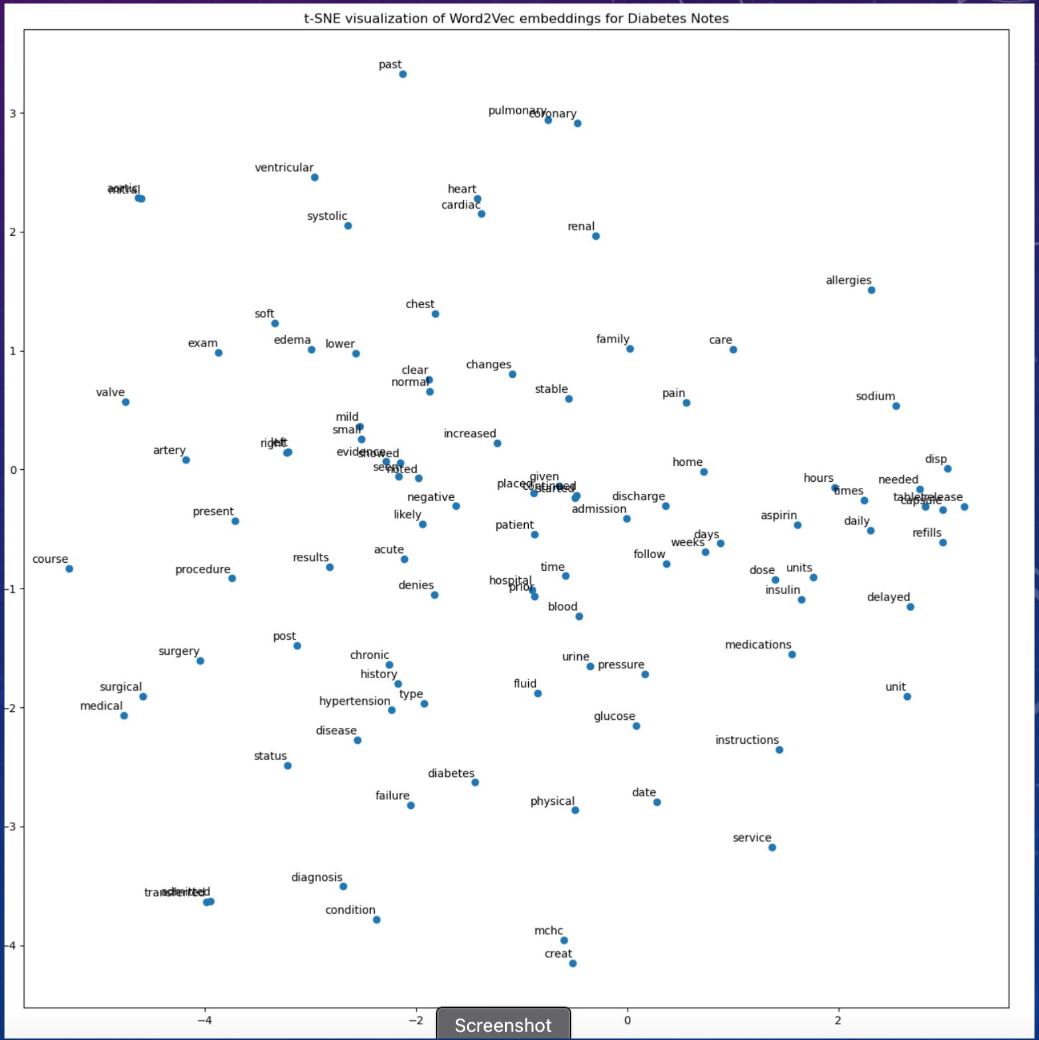
```
Words similar to 'insulin’:
[
('humalog', 0.7126625180244446),
('lantus', 0.6957399845123291),
('insuling', 0.6520899534225464),
('hiss', 0.6348435878753662),
('insuln', 0.6166067719459534),
('riss', 0.595574140548706),
('fingersticks', 0.5858661532402039),
('glargin', 0.5817741751670837),
('units', 0.555779755115509),
('ultralente', 0.5463814735412598)
]
```

VISUALIZING WORD EMBEDDINGS WITH T-SNE

Title: Visualizing Word2Vec Embeddings

- **Explanation:** Use t-SNE to reduce the dimensionality of word embeddings and visualize them.

```python
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import numpy as np

# Get vocabulary and limit to top N words
vocab = list(model.wv.key_to_index)[:100]

# Get word vectors
X = model.wv[vocab]

tsne = TSNE(n_components=2, perplexity=30, random_state=42)
X_tsne = tsne.fit_transform(X)

plt.figure(figsize=(16, 16))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1])

for i, word in enumerate(vocab):
    plt.annotate(word, xy=(X_tsne[i, 0], X_tsne[i, 1]), textcoords='offset
points', xytext=(0, 5), ha='right')
plt.title('t-SNE Visualization of Word2Vec Embeddings for Diabetes Notes')
plt.show()
```



USING CLINICALBERT FOR EMBEDDINGS

Title: Generating Embeddings with ClinicalBERT

- **Explanation:** Use the emilyalsentzer/Bio_ClinicalBERT model to generate embeddings for a clinical note.

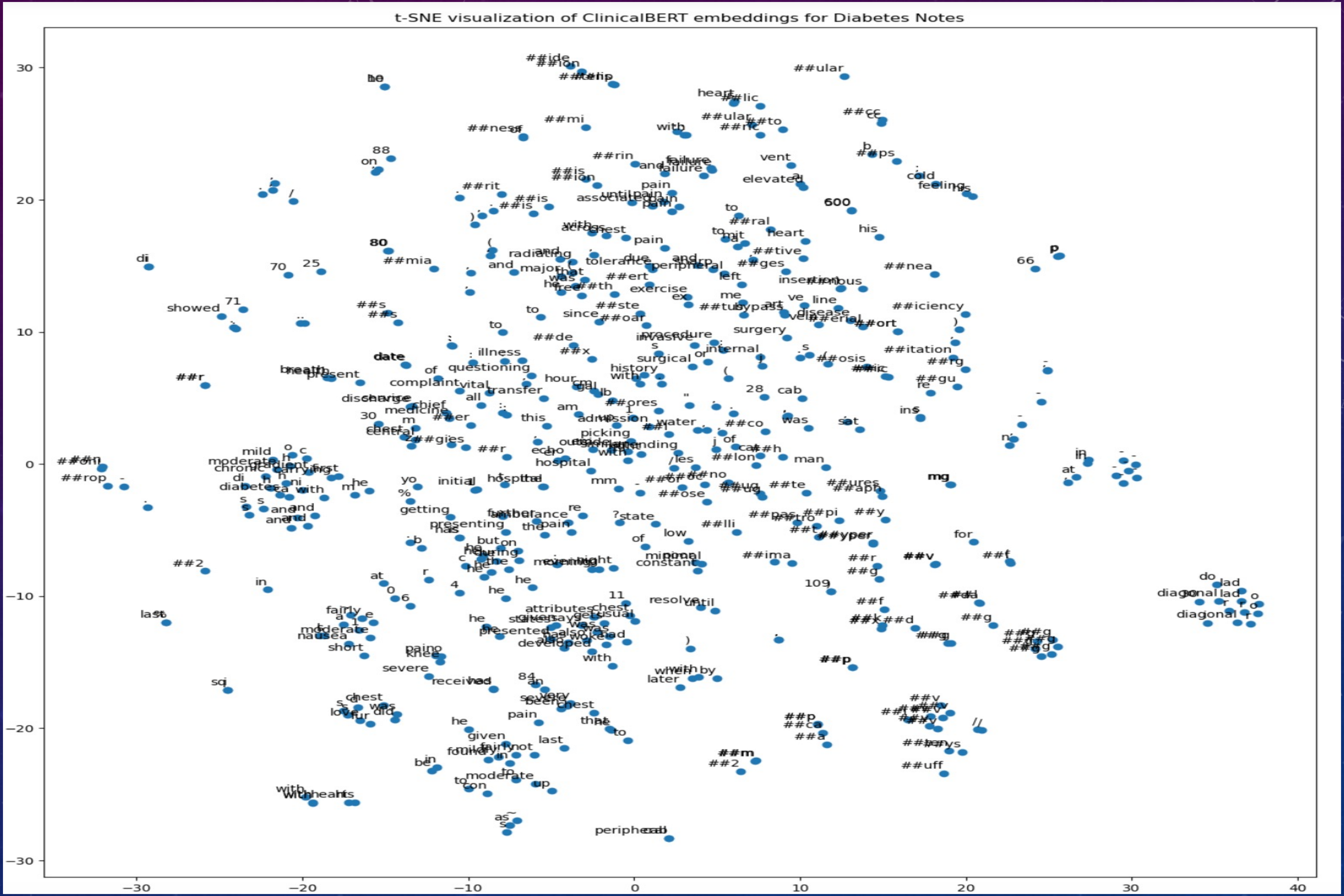
```python
from transformers import AutoTokenizer, AutoModel
import torch

tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
model_bert = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")

# Tokenize and encode a sample note
inputs = tokenizer(diabetes_notes['CLEAN_TEXT'][0], return_tensors="pt", max_length=512, truncation=True)

# Get the embeddings
with torch.no_grad():
    outputs = model_bert(**inputs)

embeddings = outputs.last_hidden_state  # Shape: [batch_size, sequence_length, hidden_size]
```

VISUALIZING CLINICALBERT EMBEDDINGS

Title: Visualizing Token Embeddings with t-SNE

- **Explanation:** Reduce the dimensions of the token embeddings and visualize them using t-SNE.

```python
# Convert embeddings to numpy for visualization
token_embeddings = embeddings[0].numpy()

# Get tokens
tokens = tokenizer.tokenize(diabetes_notes['CLEAN_TEXT'][0], max_length=512, truncation=True)

# Reduce dimensions
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
embeddings_tsne = tsne.fit_transform(token_embeddings)

# Plot
plt.figure(figsize=(16, 16))
plt.scatter(embeddings_tsne[:, 0], embeddings_tsne[:, 1])

for i, token in enumerate(tokens):
    plt.annotate(token, xy=(embeddings_tsne[i, 0], embeddings_tsne[i, 1]), textcoords='offset points', xytext=(0, 5), 
ha='right')
plt.title('t-SNE Visualization of ClinicalBERT Embeddings for Diabetes Notes')
plt.show()
```


t-SNE visualization of ClinicalBERT embeddings for Diabetes Notes
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30
40

COMPARING CLINICALBERT WITH STANDARD BERT

Title: Tokenization Comparison between ClinicalBERT and BERT

- **Explanation:** Compare the tokens generated by ClinicalBERT and standard BERT models to highlight differences in handling clinical text.

```python
# Load standard BERT tokenizer
from transformers import BertTokenizer, BertModel

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

# Tokenize with standard BERT
bert_tokens = bert_tokenizer.tokenize(diabetes_notes['CLEAN_TEXT'][0], max_length=512, truncation=True)

# Compare tokens
print("Standard BERT Tokens:")
print(bert_tokens)

print("\nClinicalBERT Tokens:")
print(tokens)
```

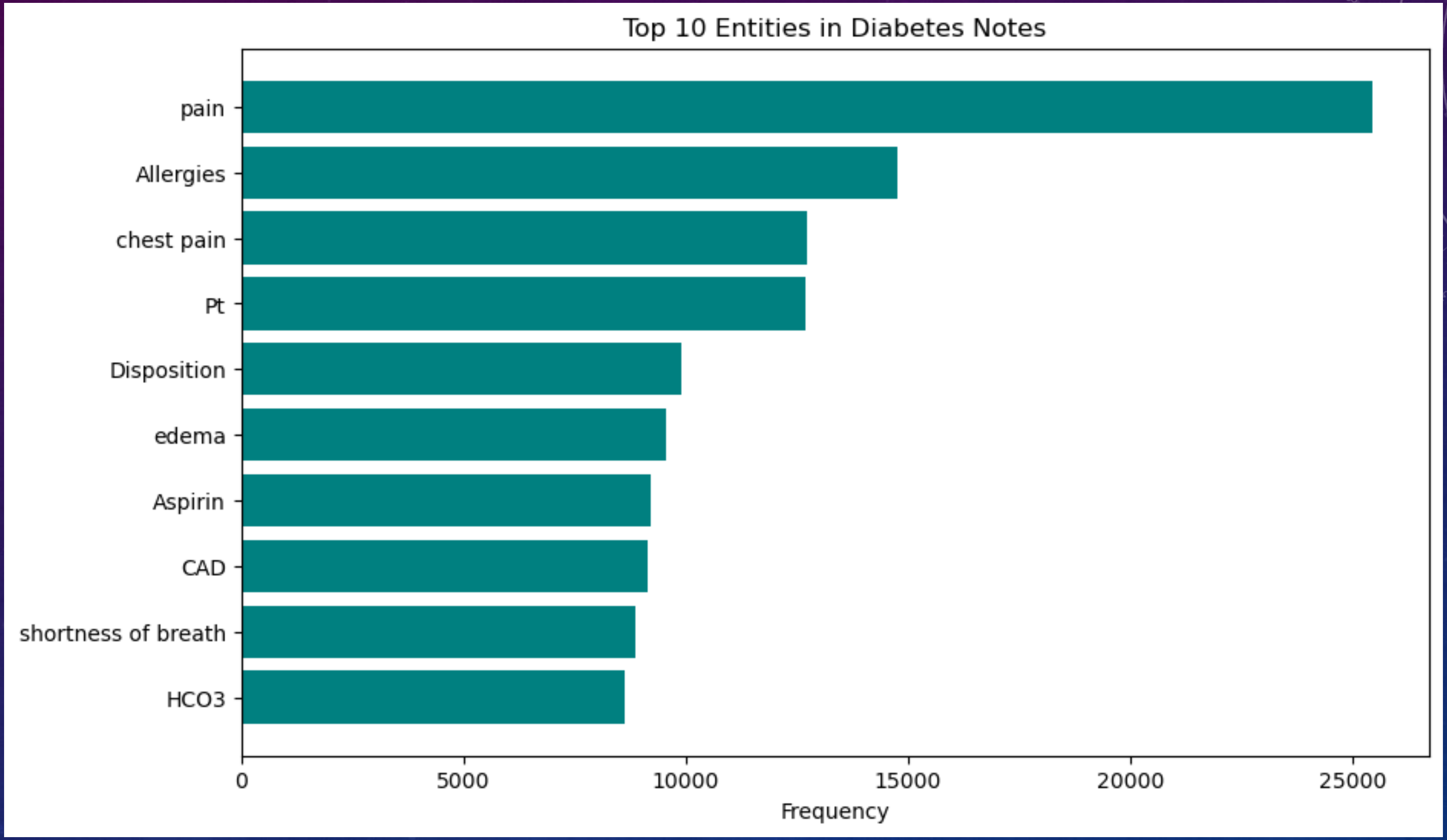
Output:

- Lists of tokens from both models showing that ClinicalBERT better handles medical terminology.

```
Standard BERT Tokens:
['admission', 'date', ':', 'discharge', 'date', ':', 'service', ':', 'medicine', 'all', '##er',
'##gies', ':', 'z', '##oco', '##r', '/', 'les', '##col', 'attending', ':', 'chief', 'complaint',
':', 'chest', 'pain', 'major', 'surgical', 'or', 'invasive', 'procedure', ':', 'central', 've',
'##nous', 'line', 'insertion', '(', 'right', 'internal', 'jug', '##ular', 'vein', ')', 'history',
'of', 'present', 'illness', ':', 'mr', '.', 'is', 'an', '84', 'yo', 'man', 'with', 'moderate',
'ao', '##rti', '##c', 'ste', '##nosis', '(', 'outside', 'hospital', 'echo', 'in', 'with', '1',
'cm', '##2', ',', 'gradient', '28', 'mm', '##hg', ',', 'moderate', 'mit', '##ral', 'reg',
'##urg', '##itation', ',', 'mild', 'ao', '##rti', '##c', 'ins', '##uf', '##fi', '##ciency', ')',
',', 'chronic', 'left', 'vent', '##ric', '##ular', 'sy', '##sto', '##lic', 'heart', 'failure',
'with', 'e', '##f', '25', '-', '30', '%', ',', 'hyper', '##tension', ',', 'hyper', '##lip',
'##ide', '##mia', ',', 'diabetes', 'mel', '##lit', '##us', ',', 'cad', 's', '/', 'p', 'cab',
'##g', 'in', 'with', 'sv', '##g', '-', 'lad', '-', 'diagonal', ',', 'sv', '##g', '-', 'om', ',',
'and', 'sv', '##g', '-', 'r', '##pd', '##a', '-', 'r', '##pl', ',', 'with', 'a', 're', '-', 'do',
'cab', '##g', 'in', 'with', 'lima', '-', 'lad', ',', 'sv', '##g', '-', 'om', ',', 'sv', '##g', '-
', 'diagonal', ',', 'and', 'sv', '##g', '-', 'rca', '.', 'he', 'also', 'has', 'severe',
'peripheral', 'arterial', 'disease', 's', '/', 'p', 'peripheral', 'bypass', 'surgery', '.', 'he',
'presented', 'to', 'hospital', 'er', 'this', 'morning', 'with', 'short’,...
```

TOP 10 ENTITLES FROM ALL DIABETES NOTES

```python
# Extract entities from all diabetes notes
entities = []
for text in diabetes_notes['CLEAN_TEXT']:
    doc = nlp_scispacy_ner(text)
    entities.extend([ent.text for ent in doc.ents])

# Count entity frequencies
from collections import Counter
entity_counts = Counter(entities)
most_common_entities = entity_counts.most_common(10)

# Separate labels and counts
labels, counts = zip(*most_common_entities)

# Plot
plt.figure(figsize=(10, 6))
plt.barh(labels, counts, color='teal')
plt.gca().invert_yaxis()
plt.xlabel('Frequency')
plt.title('Top 10 Entities in Diabetes Notes')
plt.show()
```


Top 10 Entities in Diabetes Notes
pain
Allergies
chest pain
Pt
Disposition
edema
Aspirin
CAD
shortness of breath
HCO3
0
5000
10000
15000
20000
25000
Frequency

OBSERVATIONS AND INSIGHTS

Title: Key Takeaways from NLP Analysis

- **Explanation:**
  - **Entity Recognition:**
    - SpaCy provides general-purpose entity recognition.
    - SciSpaCy and its NER model effectively identify biomedical entities like diseases and chemicals.
  - **Word Embeddings:**
    - Word2Vec captures semantic relationships between medical terms (e.g., 'insulin' is similar to 'lantus' and 'humalog').
    - ClinicalBERT generates contextual embeddings tailored for clinical text.
  - **Tokenization Differences:**
    - ClinicalBERT tokenizes medical terms more appropriately than standard BERT, leading to better representations.