NLP for MIMIC: Tutorial For Liver Data Analysis

Jesus Minjares

UT AUSTIN: AI in HEALTHCARE

Overview

- Load and Filter Data
- SpaCy for NLP Processing
- SciSpaCy for Biomedical Text
- Word2Vec for Medical Embeddings
- t-SNE for Dimensionality Reduction
- MedSpaCy for Clinical NLP

Load and Filter Data

- **Overview**
  This script filters and extracts clinical notes for patients diagnosed with **ICD-9 Code 5715 (Liver Disease)** from the **MIMIC-III** dataset.
- **Process**
- **Load Data Efficiently** – Reads NOTEEVENTS.csv (clinical notes) and DIAGNOSES_ICD.csv (diagnoses) with optimized memory usage.
- **Standardize and Clean Data** – Ensures consistent column names and removes missing values.
- **Filter for Liver Disease** – Selects records where **ICD-9 Code = 5715** and extracts **SUBJECT_IDs**.
- **Retrieve Clinical Notes** – Matches **SUBJECT_IDs** to corresponding notes.
- **Save Processed Data** – Stores the filtered notes in **'liver_notes.csv'** for NLP analysis.
- **Outcome**
- **X** unique liver disease patients identified.
- Dataset ready for NLP processing using **SpaCy, SciSpaCy, Word2Vec, and MedSpaCy**.

```python
import pandas as pd

# Load MIMIC-III dataset files (optimize memory usage)
noteevents_df = pd.read_csv("data/NOTEEVENTS.csv", usecols=["SUBJECT_ID", "CATEGORY", "TEXT"])
diagnoses_icd_df = pd.read_csv("data/DIAGNOSES_ICD.csv", dtype={"ICD9_CODE": str})

# Ensure column names are consistent
diagnoses_icd_df.columns = diagnoses_icd_df.columns.str.upper()

# Drop missing values in ICD9_CODE before filtering
diagnoses_icd_df = diagnoses_icd_df.dropna(subset=["ICD9_CODE"])

# Filter for ICD-9 Code '5715' (Liver disease)
filtered_df = diagnoses_icd_df[diagnoses_icd_df["ICD9_CODE"] == "5715"]

# Extract unique SUBJECT_IDs & HADM_IDs
arr_subject_id = set(filtered_df["SUBJECT_ID"])  # Using set() for fast lookup

# Print summary
print(f"✅ Found {len(arr_subject_id)} unique patients with ICD-9 Code '5715'.")
print("📌 Sample SUBJECT_IDs:", list(arr_subject_id)[:10])  # Show first 10 IDs

# Optimized filtering using .isin()
filtered_notes_df = noteevents_df[noteevents_df["SUBJECT_ID"].isin(arr_subject_id)]

# Display filtered data
print(filtered_notes_df.head())

# Save filtered notes to CSV
filtered_notes_df.to_csv('liver_notes.csv', index=False)
print("✅ Filtered notes saved to 'liver_notes.csv'.")
```

SpaCy

```python
import spacy
import pandas as pd

df = pd.read_csv("liver_notes.csv")[:5] # reduce to 5 for performance

nlp = spacy.load("en_core_web_sm")

def extract_entities(text):
    doc = nlp(text)
    return [(ent.text, ent.label_) for ent in doc.ents], doc

# Process and extract entities
for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values

    if entities:  # Only display if there are entities
        print(f"Idx: {idx} Entities Found: {len(entities)}")
        for entity, label in entities:
            print(f"Entity: {entity}, Label: {label}")
        print('*' * 100)
```

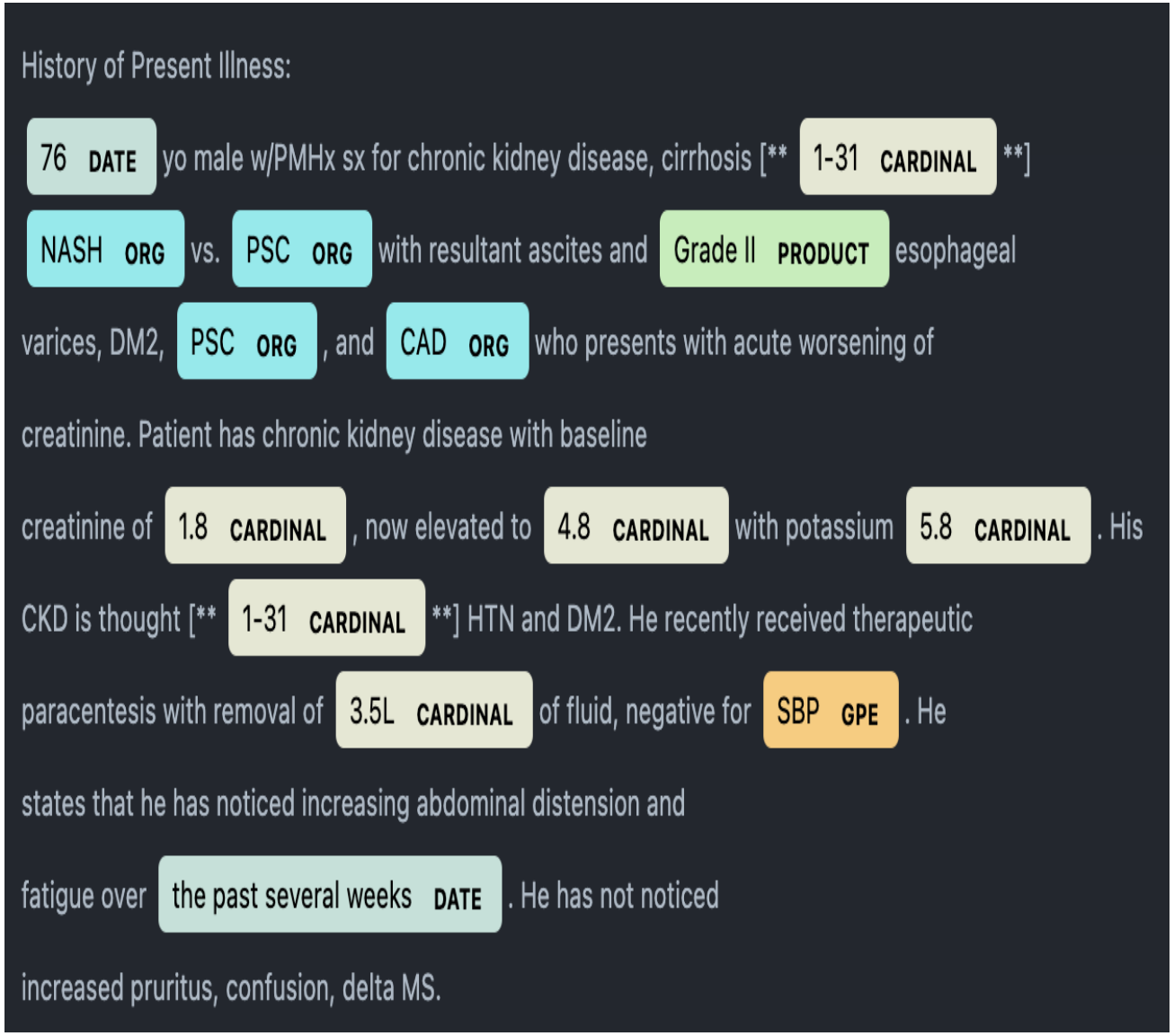
- This script extracts named entities from clinical notes using **SpaCy**. It loads the first five rows of liver_notes.csv, processes text with en_core_web_sm, and prints detected entities with their labels.
- Output:

```
Entity: 2137-3-7, Label: DATE
Entity: 2137-3-16, Label: DATE
Entity: 2060-10-8, Label: DATE
Entity: Name3 (LF, Label: ORG
Entity: 348, Label: CARDINAL
```

SpaCy with NER

- This script visualizes **NER** (Named Entity Recognition)
  in liver_notes.csv using **displaCy**.

```
from spacy import displacy

for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values

    if entities:  # Only display if there are entities
        displacy.render(doc, style="ent", jupyter=True)
        print('*' * 100)
```



SciSpaCy

```python
import scispacy
import pandas as pd
import en_core_sci_lg
import warnings

# Suppress SciSpaCy warning
warnings.filterwarnings("ignore", category=UserWarning)

df = pd.read_csv("liver_notes.csv")[:5] # only use the 5 for performance
nlp = en_core_sci_lg.load()

# Function to extract named entities using SciSpaCy
def extract_entities(text):
    doc = nlp(text)
    return [(ent.text, ent.label_) for ent in doc.ents], doc

# Process and extract entities
for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values

    if entities:  # Only display if there are entities
        print(f"Idx: {idx} Entities Found: {len(entities)}")
        for entity, label in entities:
            print(f"Entity: {entity}, Label: {label}")
        print('*' * 100)
```

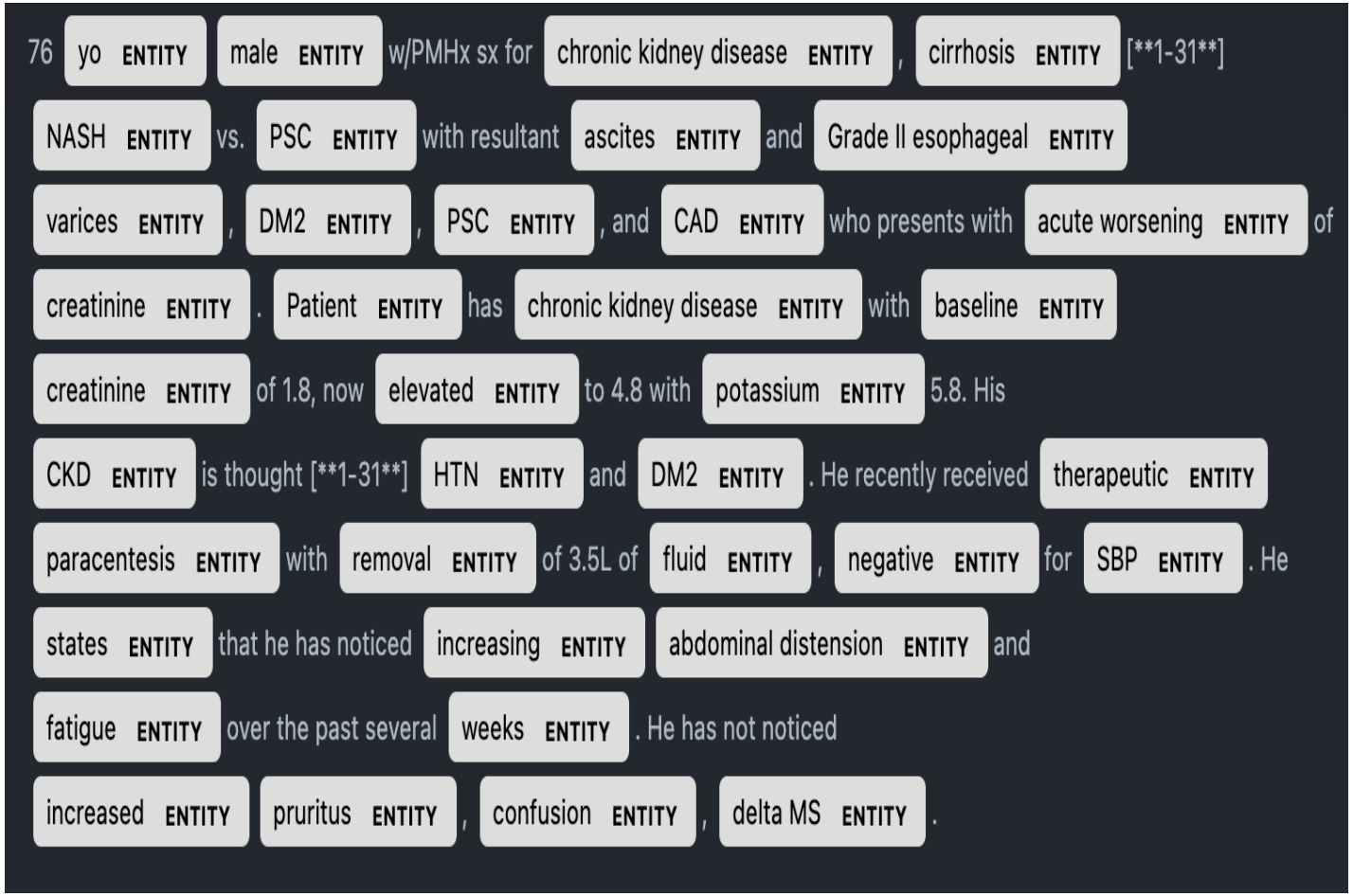
- This script extracts named entities from clinical notes using **SciSpaCy**. It loads the first five rows of liver_notes.csv, processes text with en_core_sci_lg, and prints detected entities with their labels.

```
Entity: Lisinopril, Label: ENTITY
Entity: Name3, Label: ENTITY
Entity: LF, Label: ENTITY
Entity: Acute renal failure, Label: ENTITY
Entity: Surgical, Label: ENTITY
Entity: Invasive Procedure, Label: ENTITY
```

SciSpaCy with NER

- This script visualizes **NER** (Named Entity Recognition)
  in liver_notes.csv using **displaCy**.

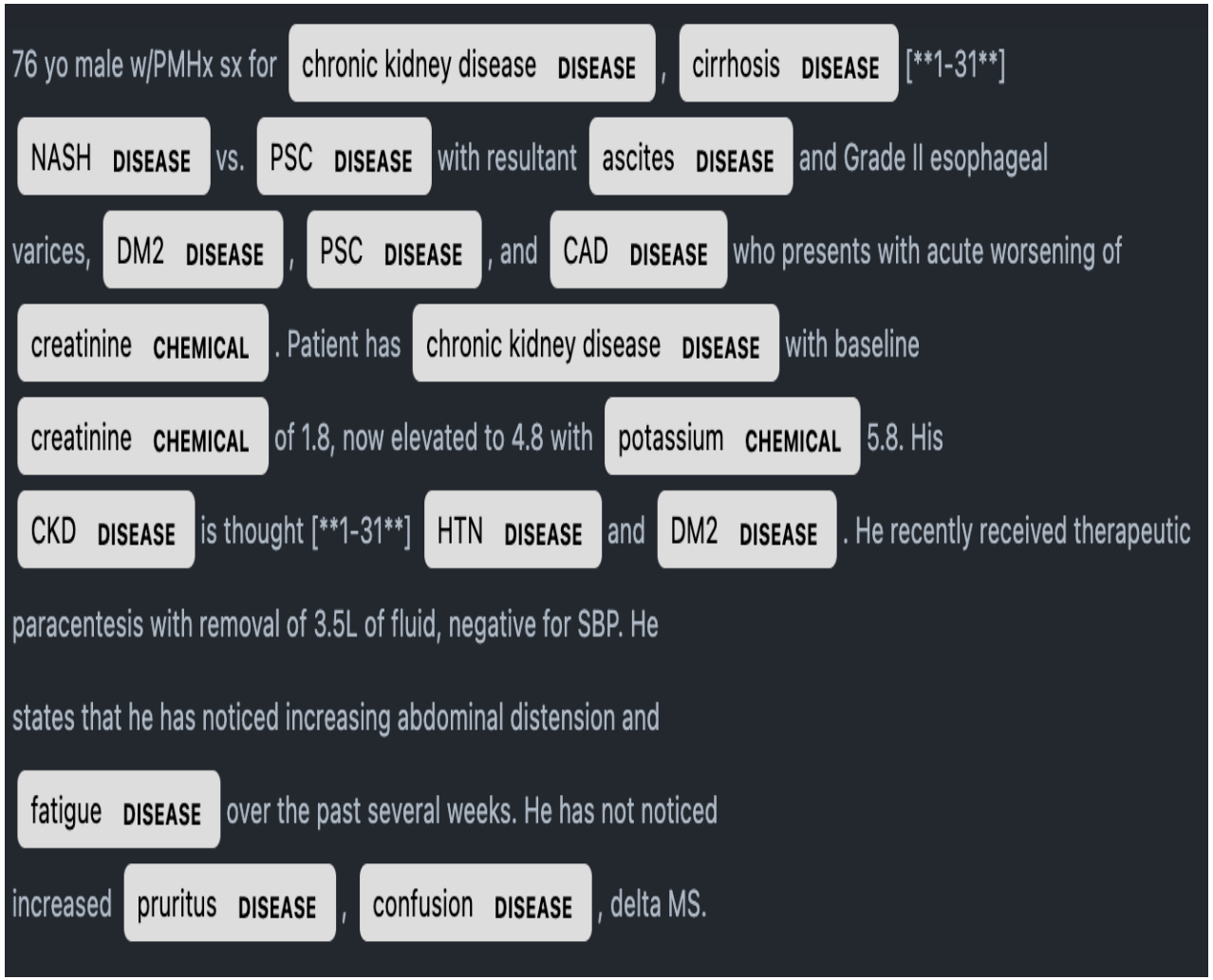
```
from spacy import displacy

for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values

    if entities:  # Only display if there are entities
        displacy.render(doc, style="ent", jupyter=True)
        print('*' * 100)
```



SciSpaCy with NER cont.

- This script extracts named entities from clinical notes using **SciSpaCy** using ***en_ner_bc5cdr_md*** model.

```
import en_ner_bc5cdr_md
import pandas as pd

df = pd.read_csv("liver_notes.csv")[:5] # only use the 5 for performance
nlp = en_ner_bc5cdr_md.load()
# Process and extract entities
for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values

    if entities:  # Only display if there are entities
        displacy.render(doc, style="ent", jupyter=True)
        print('*' * 100)
```



Word2Vec with SpaCy

- This script extracts named entities, tokenizes text, and trains a **Word2Vec** model on liver_notes.csv using **SpaCy**. It then finds similar words for a given medical term.
- Output:

```
HS                   0.9968
Solution             0.9955
HCl                  0.9954
Divalproex           0.9949
QHS                  0.9942
Plavix               0.9942
b.i.d                0.9941
QAM                  0.9939
Nadolol              0.9937
Capsule              0.9936
```

```python
import pandas as pd
import spacy
import scispacy
import en_core_sci_lg
from spacy import displacy
from gensim.models import Word2Vec

# Load dataset
df = pd.read_csv("liver_notes.csv")[:100] # set to 100

# Load spacy NLP model
nlp = spacy.load("en_core_web_sm")  # Using standard SpaCy model

# Function to extract named entities using SciSpaCy
def extract_entities(text):
    doc = nlp(text)
    return [(ent.text, ent.label_) for ent in doc.ents], doc

sentences = []
for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values
    sentences.append([token.text for token in doc if not token.is_stop and not token.is_punct])

# Train Word2Vec model on extracted text
w2v_model_spacy = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Print word vector and similar words in a formatted output
word = 'Furosemide'
if word in w2v_model_spacy.wv:
    similar_words = w2v_model_spacy.wv.similar_by_word(word)
    # Print each tuple in a row
    for word, score in similar_words:
        print(f"{word:<20} {score:<20.4f}")  # Align left, 4 decimal places for score

else:
    print(f"Word '{word}' not found in vocabulary.")
```

Word2Vec with SciSpaCy

- This script extracts named entities, tokenizes text, and trains a **Word2Vec** model on liver_notes.csv using **SciSpaCy**. It then finds similar words for a given medical term.
- Output:

```
500                  0.9980
Lasix                0.9979
po                   0.9976
bedtime              0.9974
q.                   0.9974
q                    0.9974
mg                   0.9973
p.o                  0.9971
QID                  0.9971
Aldactone            0.9968
```

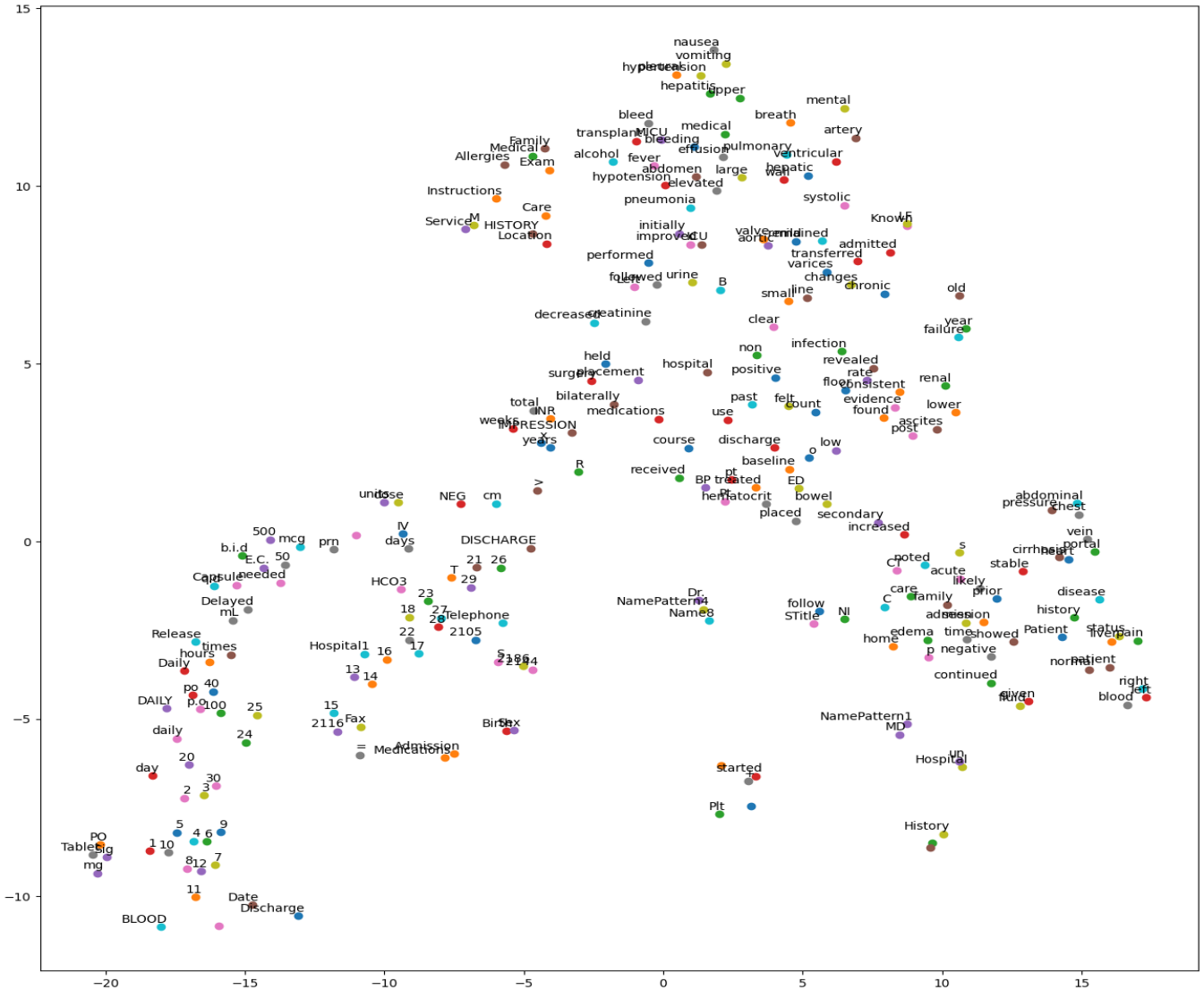
```python
import pandas as pd
import spacy
import scispacy
import en_core_sci_lg
from spacy import displacy
from gensim.models import Word2Vec

# Load dataset
df = pd.read_csv("liver_notes.csv")[:100] # set to 100

# Load SciSpaCy NLP model
nlp = en_core_sci_lg.load()  # Using larger SciSpaCy model for better medical term recognition

# Function to extract named entities using SciSpaCy
def extract_entities(text):
    doc = nlp(text)
    return [(ent.text, ent.label_) for ent in doc.ents], doc

sentences = []
for idx, row in df.iterrows():
    text = row['TEXT']
    entities, doc = extract_entities(text)  # Now correctly returning two values
    sentences.append([token.text for token in doc if not token.is_stop and not token.is_punct])

# Train Word2Vec model on extracted text
w2v_model_scispacy = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Print word vector and similar words in a formatted output
word = 'Furosemide'
if word in w2v_model_scispacy.wv:
    similar_words = w2v_model_scispacy.wv.similar_by_word(word)
    # Print each tuple in a row
    for word, score in similar_words:
        print(f"{word:<20} {score:<20.4f}")  # Align left, 4 decimal places for score
else:
    print(f"Word '{word}' not found in vocabulary.")
```

t-SNE

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

- This code uses **t-SNE** to reduce word embeddings to **2D space**, then plots them with **Matplotlib**, labeling each word to visualize relationships and clustering patterns.

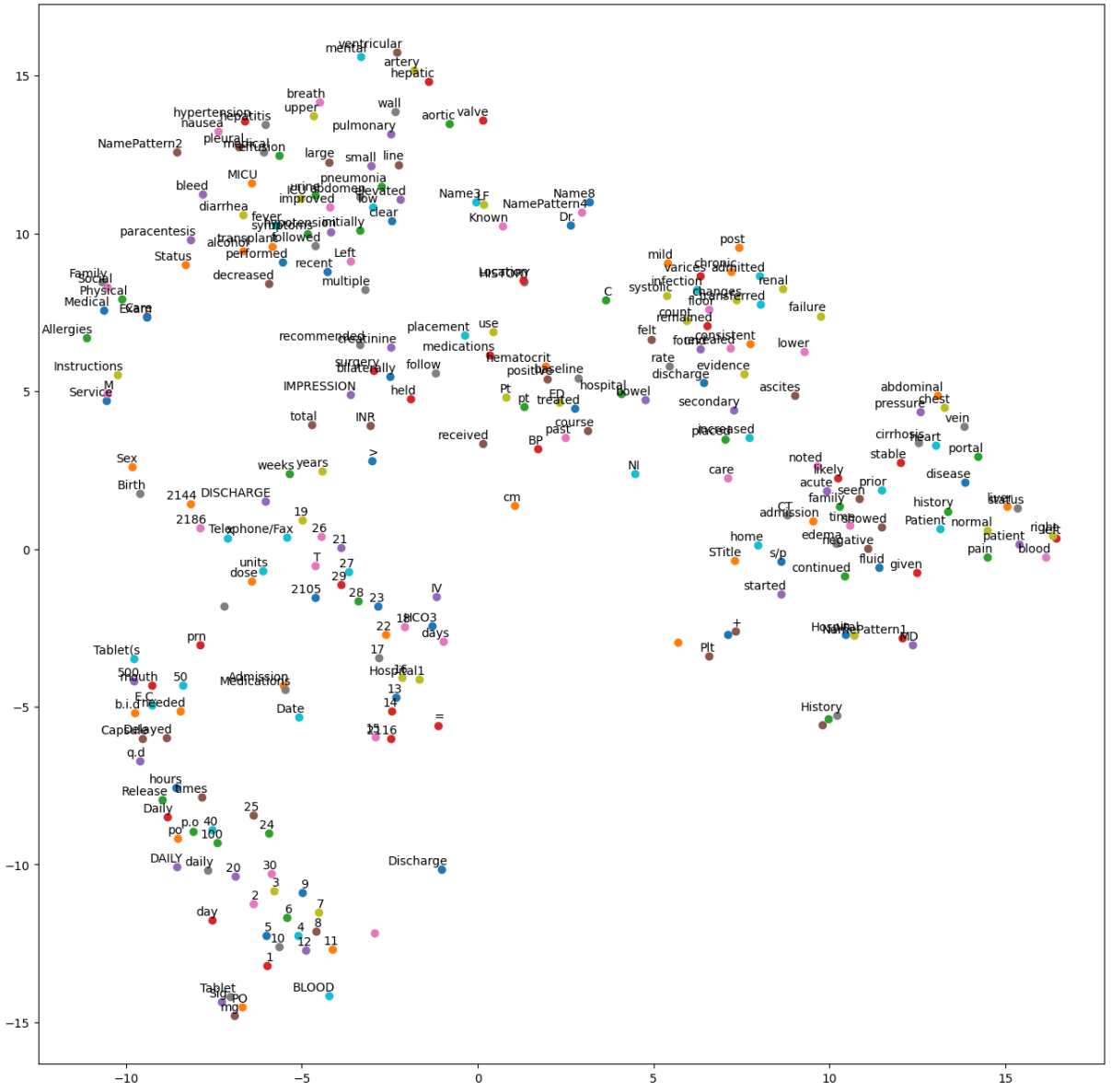
t-SNE: SpaCy



- This code selects the top **250 words** from the **SpaCy-based Word2Vec** model, converts them into a NumPy array, and visualizes their embeddings using **t-SNE**.

```
vocabs = list(w2v_model_spacy.wv.index_to_key)[:250]
new_v = np.array(vocabs)
tsne_plot(w2v_model_spacy,new_v)
```

t-SNE: SciSpaCy



- This code selects the top **250 words** from the **SciSpaCy -based Word2Vec** model, converts them into a NumPy array, and visualizes their embeddings using **t-SNE**.

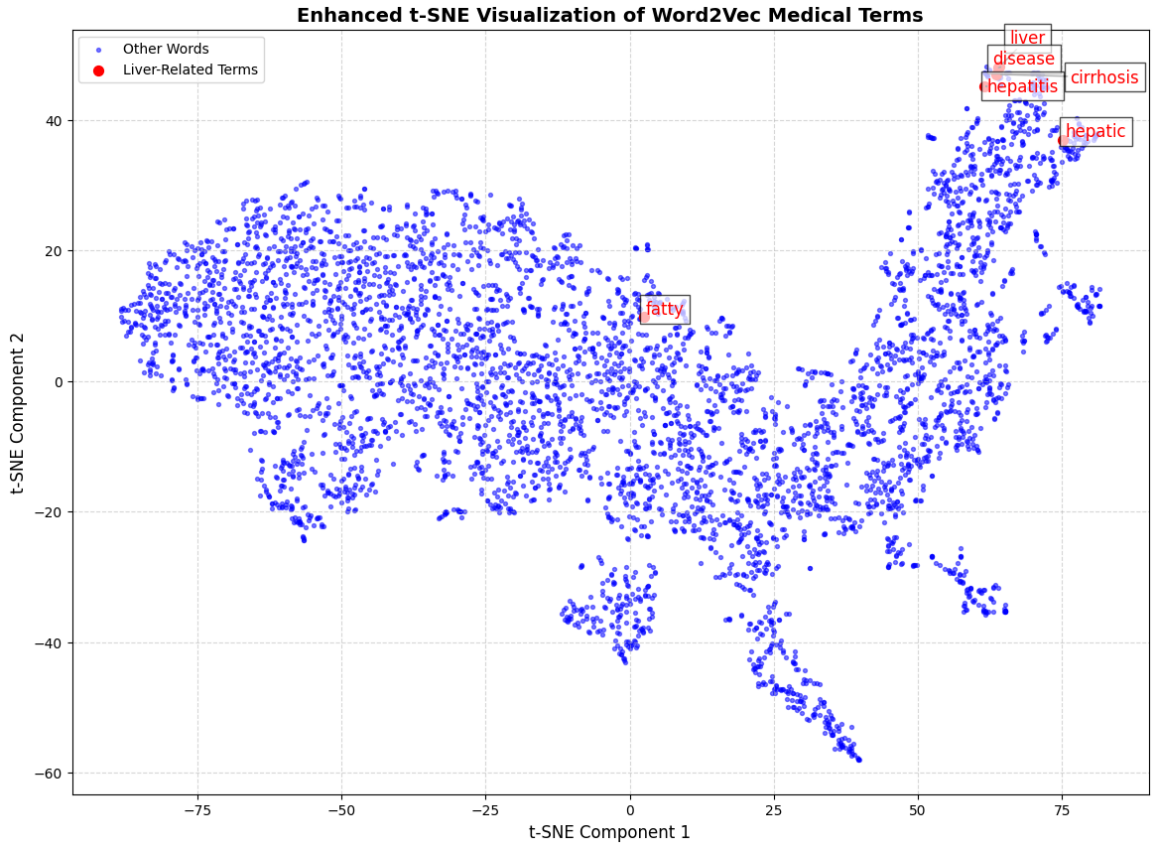
```
vocabs = list(w2v_model_scispacy.wv.index_to_key)[:250]
new_v = np.array(vocabs)
tsne_plot(w2v_model_scispacy,new_v)
```

MedSpaCy

- This script uses **MedSpaCy** to extract medical entities from clinical notes, tokenizes the text, and trains a **Word2Vec** model to generate word embeddings, which serve as the vocabulary for **t-SNE** visualization.

```
import medspacy
from medspacy.ner import TargetRule
import pandas as pd
import gensim
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import re
import nltk
from nltk.tokenize import word_tokenize
from adjustText import adjust_text  # Import adjustText for non-overlapping labels

# Load dataset (Limit to first 100 samples)
df = pd.read_csv("liver_notes.csv")[:100]

# Initialize MedSpacy NLP pipeline
nlp = medspacy.load(enable=['sentencizer', 'medspacy_target_matcher'])

# Define Target Rules for entity extraction (Liver-related conditions)
target_rules = [
    TargetRule('hyperlipidemia', 'DISEASE'),
    TargetRule('lipid', 'SUBSTANCE'),
    TargetRule('hypertension', 'DISEASE'),
    TargetRule('obesity', 'CONDITION'),
    TargetRule('cardiac', 'ENTITY'),
    # Liver-related conditions
    TargetRule('fatty liver', 'DISEASE'),
    TargetRule('cirrhosis', 'DISEASE'),
    TargetRule('hepatitis', 'DISEASE'),
    TargetRule('liver disease', 'DISEASE'),
    TargetRule('hepatic failure', 'DISEASE'),
]

# Add entity rules to MedSpacy pipeline
nlp.get_pipe('medspacy_target_matcher').add(target_rules)

# Extract Entities and Tokenize Text for Word2Vec
tokenized_texts = []
entities = []

for text in df['TEXT']:
    doc = nlp(text)

    # Tokenize text (remove special characters)
    tokens = [re.sub(r'\W+', '', token.text.lower()) for token in doc if token.text.isalpha()]
    tokenized_texts.append(tokens)

    # Extract disease entities
    entities.extend([ent.text for ent in doc.ents if ent.label_ == 'DISEASE'])

# Train Word2Vec Model
word2vec_model = Word2Vec(sentences=tokenized_texts, vector_size=100, window=5, min_count=2, workers=4)


# Get vectors for words in the vocabulary
words = list(word2vec_model.wv.index_to_key)
word_vectors = word2vec_model.wv[words]
```

MedSpaCy TSNE Plot




```python
# Reduce dimensions using t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30)
word_vectors_2d = tsne.fit_transform(word_vectors)

# Plot Word Embeddings
plt.figure(figsize=(14, 10))
plt.scatter(word_vectors_2d[:, 0], word_vectors_2d[:, 1], alpha=0.5, s=8, color='blue', label="Other
Words")

# Highlight Liver-Related Terms in Red
liver_terms = {'liver', 'fatty', 'cirrhosis', 'hepatitis', 'hepatic', 'disease'}
word_positions = {word: word_vectors_2d[i] for i, word in enumerate(words)}

texts = []  # Store text annotations for adjustment

for word in liver_terms:
    if word in word_positions:
        x, y = word_positions[word]
        plt.scatter(x, y, color='red', s=50, label="Liver-Related Terms" if "Liver-Related Terms" not
in plt.gca().get_legend_handles_labels()[1] else "")
        text = plt.text(x, y, word, fontsize=12, color='red', bbox=dict(facecolor='white', alpha=0.7,
edgecolor='black'))
        texts.append(text)

# Adjust text labels to avoid overlap
adjust_text(texts, arrowprops=dict(arrowstyle="->", color='black', lw=1))

# Final Touches
plt.title("Enhanced t-SNE Visualization of Word2Vec Medical Terms", fontsize=14, fontweight='bold')
plt.xlabel("t-SNE Component 1", fontsize=12)
plt.ylabel("t-SNE Component 2", fontsize=12)
plt.legend(loc="upper left")
plt.grid(True, linestyle='--', alpha=0.5)

# Show the Improved Plot
plt.show()

# Display Extracted Entities
print("Extracted Entities:", entities)
```