NLP Analysis of MIMIC-III Patient Infection Data

AI in Healthcare (AI 395T)

Jef DeWitt - February 19, 2024

Introduction

This presentation explores Natural Language Processing (NLP) tools for analyzing patient infections using MIMIC-III data. It includes Named Entity Recognition (NER), data visualizations, and Topic Modeling (LDA).

Dependencies & Setup

- Necessary Python packages are installed and imported, including:
- Spacy for Named Entity Recognition (NER).
- Gensim for Topic Modeling (shown in final graphic).
- Matplotlib for visualization.
- Scikit-learn for dimensionality reduction via t-distributed Stochastic Neighbor Embedding (t-SNE).

```
%pip install -U pip setuptools wheel
%pip install spacy==3.7.5
%pip install gensim
%pip install transformers
%pip install seaborn

import spacy
from spacy import displacy # Entity Visualizer
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import os
```

Loading Data

- Pandas loads the following datasets:
  - DIAGNOSES_ICD.csv for medical diagnoses
  - NOTEEVENTS.csv for clinical text analysis
- Merge diagnosis data with medical notes to create a discharge dataframe.

```
MIMIC_3_DIR = '../mimic/mimic-iii-clinical-database-1.4'
os.chdir(MIMIC_3_DIR)

# Load diagnosis & notes data
diagnosis_df = pd.read_csv('DIAGNOSES_ICD.csv').set_index('ROW_ID')
notes_df = pd.read_csv('NOTEEVENTS.csv', low_memory=False).set_index('ROW_ID')
```

```
# Create a dataframe with only discharge summaries
discharge_summaries_df = notes_df.loc[notes_df['CATEGORY'] == 'Discharge summary', ['SUBJECT_ID', 'HADM_ID', 'TEXT']]

# discharge_summaries_df.head()

# Common infection codes
infection_codes = ['0380', '03810', '0382', '0383', '03842', '03843']
# Filter for only infection codes
infection_df = diagnosis_df[diagnosis_df['ICD9_CODE'].isin(infection_codes)].copy()

infection_types = {
    "0380": "Streptococcal septicemia",
    "03810": "Staphylococcal septicemia, unspecified",
    "0382": "Pneumococcal septicemia [Streptococcus pneumoniae septicemia]",
    "0383": "Septicemia due to anaerobes",
    "03842": "Septicemia due to escherichia coli [E. coli]",
    "03843": "Septicemia due to pseudomonas",
}

# Map infection types to infection codes
infection_df['INFECTION_TYPE'] = infection_df['ICD9_CODE'].map(infection_types)
```

Infections Data

- Find the most common septicemia (blood) infections.
- Sort by most frequent occurrence.

```
# Merge infection data with discharge summaries
discharge_summaries_df = pd.merge(discharge_summaries_df, infection_df, on='HADM_ID', how='inner')
# Drop duplicates
discharge_summaries_df.drop_duplicates(subset='HADM_ID', inplace=True)

discharge_summaries_df['INFECTION_TYPE'].value_counts()
```

```
INFECTION_TYPE
Septicemia due to escherichia coli [E. coli]                    455
Streptococcal septicemia                                        353
Septicemia due to pseudomonas                                   116
Septicemia due to anaerobes                                     106
Pneumococcal septicemia [Streptococcus pneumoniae septicemia]    85
Staphylococcal septicemia, unspecified                           43
```

Tokenization & Entity Extraction

- Helper methods apply the specified NLP model.
- If no model is provided, the default value is spacy (en_core_web_md), a medium-sized English language model trained on written web text.

```python
# Load specified model (spacy/scispacy)
def load_nlp_model(model_name="en_core_web_md"): # Default spacy
    try:
        return spacy.load(model_name) # Loads specified model.
    except OSError:
        raise ValueError(f"Could not load model '{model_name}'. Make sure it is installed.")

# Tokenization function (supports both spaCy & SciSpaCy)
def tokenize_text(text, model_name="en_core_web_md"):
    nlp = load_nlp_model(model_name) # Loads specified model
    doc = nlp(text)

    tokens = [
        token.text.lower() # Lowercasing tokens
        for token in doc
        # Removing stopwords, punctuation, and keeping only words and numbers
        if (token.is_alpha or token.is_digit) and not token.is_stop and not token.is_punct
    ]

    return " ".join(tokens)

# Named Entity Recognition (NER) function
def extract_entities(text, model_name="en_core_web_md"):
    nlp = load_nlp_model(model_name)
    doc = nlp(text)
    # Extracts named entities from text using the specified NLP model
    entities = [(ent.text, ent.start_char, ent.end_char, ent.label_) for ent in doc.ents]
    # Returns a list of (text, start_char, end_char, label) tuples
    return entities
```

SpaCy NER

- First, E. Coli infections are targeted.
- SpaCy extracts medical entities from the medical notes (NOTEEVENTS.csv) related to E. Coli.
- NER helps identify key medical terms like diseases, symptoms, and medications.
- The following page contains the respective entity visualizations.

```
spacy_nlp = load_nlp_model("en_core_web_md")

# Get E. coli infection discharge summaries
ecoli_df = discharge_summaries_df[discharge_summaries_df['ICD9_CODE'] == '03842']

# Preprocess text for E. coli infection discharge summaries using spacy
ecoli_df['SPACY_PROCESSED_TEXT'] = ecoli_df['TEXT'].apply(lambda text: tokenize_text(text, "en_core_web_md"))

# Extract entities from E. coli infection df's first discharge summary
entities = extract_entities(ecoli_df.iloc[0]['SPACY_PROCESSED_TEXT'], "en_core_web_md")
# pretty print entities
for entity in entities:
    print(entity)
```

```
# Visualize entities in the first discharge summary PROCESSED_TEXT
doc = spacy_nlp(ecoli_df['SPACY_PROCESSED_TEXT'].iloc[0])
displacy.render(doc, style="ent", jupyter=True)
print('**********************************************************
```

spaCy NER Results

```
('2194', 15, 19, 'CARDINAL')
('7', 22, 23, 'CARDINAL')
('2194', 39, 43, 'CARDINAL')
('2 2', 44, 47, 'CARDINAL')
('290', 121, 124, 'CARDINAL')
('age 90', 233, 239, 'DATE')
('russian', 242, 249, 'NORP')
('ems', 319, 322, 'ORG')
('101 bp', 396, 402, 'QUANTITY')
('20 98', 406, 411, 'CARDINAL')
('4 l ns vanco 1 g', 454, 470, 'QUANTITY')
('1 g', 486, 489, 'QUANTITY')
('500 mg', 500, 506, 'QUANTITY')
('mg ativan 2 mg', 519, 533, 'PRODUCT')
('153', 582, 585, 'CARDINAL')
('10 day', 596, 602, 'DATE')
('7 day', 643, 648, 'DATE')
('90', 941, 943, 'CARDINAL')
('brady', 1008, 1013, 'PERSON')
('4', 1030, 1031, 'CARDINAL')
```

admission date 2194 CARDINAL 1 7 CARDINAL discharge date 2194 CARDINAL 2 2 CARDINAL service medicine allergies patient recorded having known allergies drugs 290 CARDINAL chief complaint altered mental status hypotension major surgical invasive procedure history present illness age 90 DATE f russian NORP speaking h o refractory nodular sclerosing hodgkins lymphoma brought ems ORG admitted home health care aide noted hypotensive confused ed t rectal hr 101 bp QUANTITY rr 20 98 CARDINAL incontinant guaiac positive stool treated 4 l ns vanco 1 g QUANTITY iv ceftazadime 1 g QUANTITY iv flagyl 500 mg QUANTITY iv received mg ativan 2 mg PRODUCT iv morphine agitation pt admitted hospital unit 153 CARDINAL completed 10 day DATE course ceftazadime vancomycin urosepsis 7 day DATE course metronidazole completed empiric treatment diff given loose stools setting elevated wbc count diff assays negative pt stabilized transferred floor care time transfer active issues poor nutritional status thrombocytopenia anemia floor pt experienced episode new afib rvr hypotension sbp 90 CARDINAL respiratory distress received fluid resuscitation concern tachy brady PERSON syndrome pauses 4 CARDINAL sec ORG telemetry ep curbside felt digoxin recommended pt readmitted micu micu PERSON started vancomycin piperacillin tazobactam 1 CARDINAL sites possible infection erosions breasts right hip question pna 2 CARDINAL rising wbc reaching high nearly 17 CARDINAL micu GPE course marked 1 CARDINAL hypotension responded gentle ns boluses 2 CARDINAL low uop believed 2 22 hypovolemia low baseline nitrogenous load obligate urine output 3 CARDINAL recurrent afib transitioned amiodarone 400 mg po MONEY

SciSpacy NER

- Again, E. Coli infections are targeted for a side-by-side comparison with the spaCy model.
- SciSpacy is a Python package built on top of spacy, specialized for biomedical, scientific, and clinical text. The specific model version used is en_core_sci_md.
- ScispaCy extracts medical entities from clinical text.
- Note the additional entities highlighted on the following page, indicating higher specialization/performance for processing medical texts.

```
# Load a biomedical NLP model
scispacy_nlp = load_nlp_model("en_core_sci_md")

# Preprocess text for Ecoli infection discharge summaries using scispacy
ecoli_df['SCISPACY_PROCESSED_TEXT'] = ecoli_df['TEXT'].apply(lambda text: tokenize_text(text, "en_core_sci_md"))

# Use scispacy to extract entities from the first discharge summary
entities = extract_entities(ecoli_df.iloc[0]['SCISPACY_PROCESSED_TEXT'], "en_core_sci_md")

# pretty print entities
for entity in entities:
    print(entity)
    print('****************')
```

```
# Visualize entities in the first discharge summary PROCESSED_TEXT
doc = scispacy_nlp(ecoli_df['SCISPACY_PROCESSED_TEXT'].iloc[0])
displacy.render(doc, style="ent", jupyter=True)
print('************************************************************
```

scispaCy NER Results

```
('admission', 0, 9, 'ENTITY')
**********
('discharge', 24, 33, 'ENTITY')
**********
('service medicine', 48, 64, 'ENTITY')
**********
('patient', 75, 82, 'ENTITY')
**********
('allergies drugs', 105, 120, 'ENTITY')
**********
('chief complaint', 125, 140, 'ENTITY')
**********
('mental status', 149, 162, 'ENTITY')
**********
('hypotension', 163, 174, 'ENTITY')
**********
('surgical invasive procedure history', 181, 216, 'ENTITY')
**********
('illness age', 225, 236, 'ENTITY')
**********
```

admission ENTITY date 2194 1 7 discharge ENTITY date 2194 2 2 service medicine ENTITY allergies patient ENTITY recorded having known allergies drugs ENTITY 290 chief complaint ENTITY altered mental status ENTITY hypotension ENTITY major surgical invasive procedure history ENTITY present illness age ENTITY 90 f refractory ENTITY nodular sclerosing hodgkins lymphoma ENTITY brought ems ENTITY admitted home health care aide noted hypotensive ENTITY confused ed t rectal hr 101 bp rr 20 98 incontinant ENTITY stool ENTITY treated ENTITY 4 l ns vanco ENTITY 1 g iv ceftazadime 1 g iv flagyl 500 mg iv received mg ativan ENTITY 2 mg iv morphine ENTITY agitation ENTITY pt admitted hospital unit ENTITY 153 completed course ENTITY ceftazadime ENTITY vancomycin ENTITY urosepsis ENTITY course ENTITY metronidazole ENTITY completed empiric treatment ENTITY diff given loose stools ENTITY setting elevated ENTITY wbc count diff assays ENTITY negative ENTITY pt stabilized ENTITY transferred floor ENTITY care time transfer ENTITY active issues ENTITY poor nutritional status thrombocytopenia ENTITY anemia ENTITY floor ENTITY pt experienced episode ENTITY new afib rvr ENTITY hypotension ENTITY sbp 90 respiratory distress ENTITY received fluid resuscitation concern syndrome ENTITY pauses ENTITY 4 sec telemetry ep curbside felt digoxin ENTITY recommended pt readmitted ENTITY micu ENTITY micu ENTITY started vancomycin ENTITY 1 sites ENTITY possible infection ENTITY erosions ENTITY breasts right ENTITY hip question pna 2 rising wbc reaching high nearly 17 micu ENTITY course ENTITY

Corpus & Word Vectors

- A corpus of text was built that contains the previously identified entities.
- Word2Vec shows the word vector for ‘antibiotics’ (next page, left image).
- Find similar words & their similarity scores (next page, right image).

```
# Build a corpus of processed text
corpus = []

# Iterate through PROCESSED_TEXT on only the first 5 rows
for row in range(0, 5):
    str_tokens = []
    text = ecoli_df['SCISPACY_PROCESSED_TEXT'].iloc[row]
    print(f"Processing row {row}: {text}")
    tokens = scispacy_nlp(text).ents
    if not tokens:
        print(f"No entities found in row {row}")
    for i in range(0, len(tokens)):
        str_tokens.append(tokens[i].text)
    corpus.append(list(str_tokens))

print(corpus)
```

```
from gensim.models import Word2Vec
word2vec_model = Word2Vec(corpus, min_count=1)
word2vec_model.wv['antibiotics']
```

```
word2vec_model.wv.similar_by_word('antibiotics')
```

Corpus & Word Vectors Results

```
array([ 0.00127839,  0.00132863, -0.00432331, -0.00381365,  0.00630039,
        0.00587752, -0.00165661,  0.00383083, -0.00406939, -0.00390675,
        0.0061278 , -0.00633481,  0.00861772,  0.00040693,  0.00759922,
        0.00331962,  0.00091629,  0.00103476,  0.00483985, -0.00023541,
        0.0010482 , -0.00449519, -0.00946564, -0.00085228,  0.00969883,
       -0.0017203 , -0.00411841, -0.0055458 , -0.00393655, -0.00337988,
       -0.00130955,  0.00345893, -0.00108947,  0.00660341, -0.00199067,
       -0.00040092, -0.00051604,  0.00150951, -0.00960396,  0.00096202,
        0.00748328, -0.0067242 ,  0.0078347 ,  0.00856227, -0.00885135,
       -0.00232461, -0.01002464, -0.00561671,  0.00651266,  0.00814657,
       -0.00342166,  0.00508335, -0.0049611 ,  0.00051807, -0.00736328,
       -0.00173554,  0.00910663, -0.00050654, -0.00411689, -0.00306152,
        0.00971283,  0.00723693,  0.00765387,  0.00404767,  0.00866525,
        0.00653758,  0.00890762,  0.00347636, -0.00887182,  0.00340475,
        0.00478211,  0.00422526,  0.00358734,  0.00483433,  0.00084302,
       -0.00906825,  0.00973294,  0.00988762,  0.00014675, -0.00827446,
       -0.00983679,  0.00079985, -0.00306615, -0.00483288,  0.00544199,
       -0.009021  ,  0.00229997, -0.00025632,  0.00578564,  0.00221817,
       -0.00795531, -0.00219638,  0.00579097,  0.00637872, -0.00499631,
       -0.00245395, -0.00930085,  0.00671605,  0.00994765,  0.00858904],
      dtype=float32)
```

```
[('discharge', 0.2776538133621216),
 ('reflexes bilaterally physical exam transfer', 0.2771392166614532 5),
 ('lungs ctab', 0.2666809558868408),
 ('dose', 0.26522624492645264),
 ('surgeries', 0.2628735899925232),
 ('metronidazole', 0.25460758805274963),
 ('trend lfts', 0.24225164949893951),
 ('autopsy', 0.24105584621429443),
 ('gram positive cocci bacteremia', 0.2357354760169983),
 ('daily', 0.23296909034252167)]
```

t-SNE Visualization

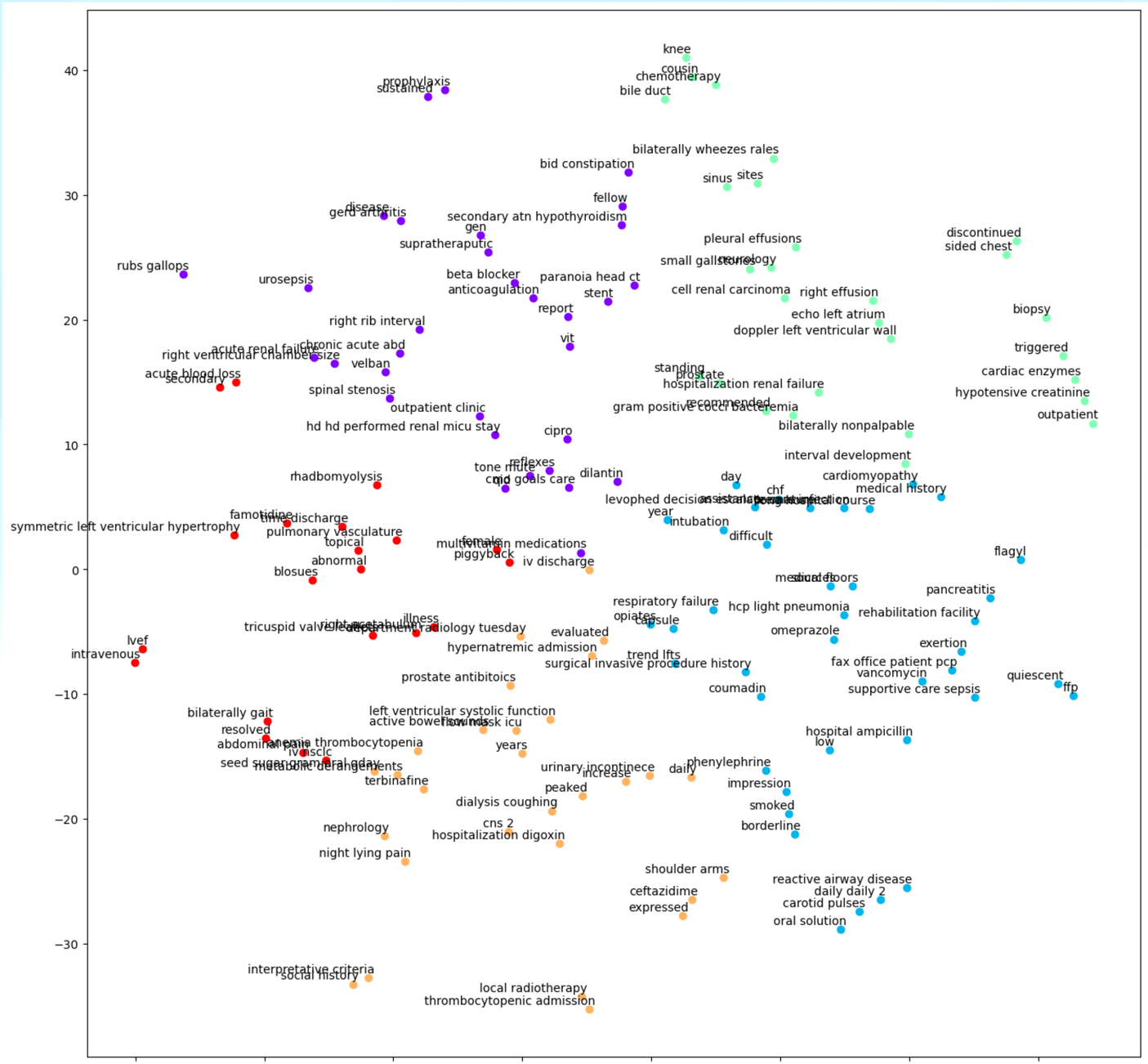
- t-SNE projects high-dimensional text embeddings typically into a 2D or 3D space.
- This helps visualize clustering patterns. On the following page, related medical terms will appear closer.

```python
from sklearn.cluster import KMeans
import random

def tsne_plot(model, words, preTrained=False, max_words=100, n_clusters=5):
    "Creates and TSNE model and plots it with K-Means clustering"
    labels = []
    tokens = []

    # Sample a subset of words if there are too many
    if len(words) > max_words:
        words = random.sample(words, max_words)

    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=5, early_exaggeration=50, n_components=2, init='pca', n_iter=1000, random_state=23)
    new_values = tsne_model.fit_transform(tokens)

    # Perform K-Means clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=23)
    kmeans.fit(new_values)
    cluster_labels = kmeans.labels_

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])

    plt.figure(figsize=(16, 16))
    colors = plt.cm.rainbow(np.linspace(0, 1, n_clusters))
    for i in range(len(x)):
        plt.scatter(x[i], y[i], color=colors[cluster_labels[i]])
        plt.annotate(labels[i],
                     xy=(x[i], y[i]),
                     xytext=(5, 2),
                     textcoords='offset points',
                     ha='right',
                     va='bottom')
    plt.show()
```

```python
vocabs = word2vec_model.wv.index_to_key
new_v = list(vocabs)
tsne_plot(word2vec_model, new_v, preTrained=False, max_words=150)
```

t-SNE Visualization Results

- Related entity clusters are grouped by color.



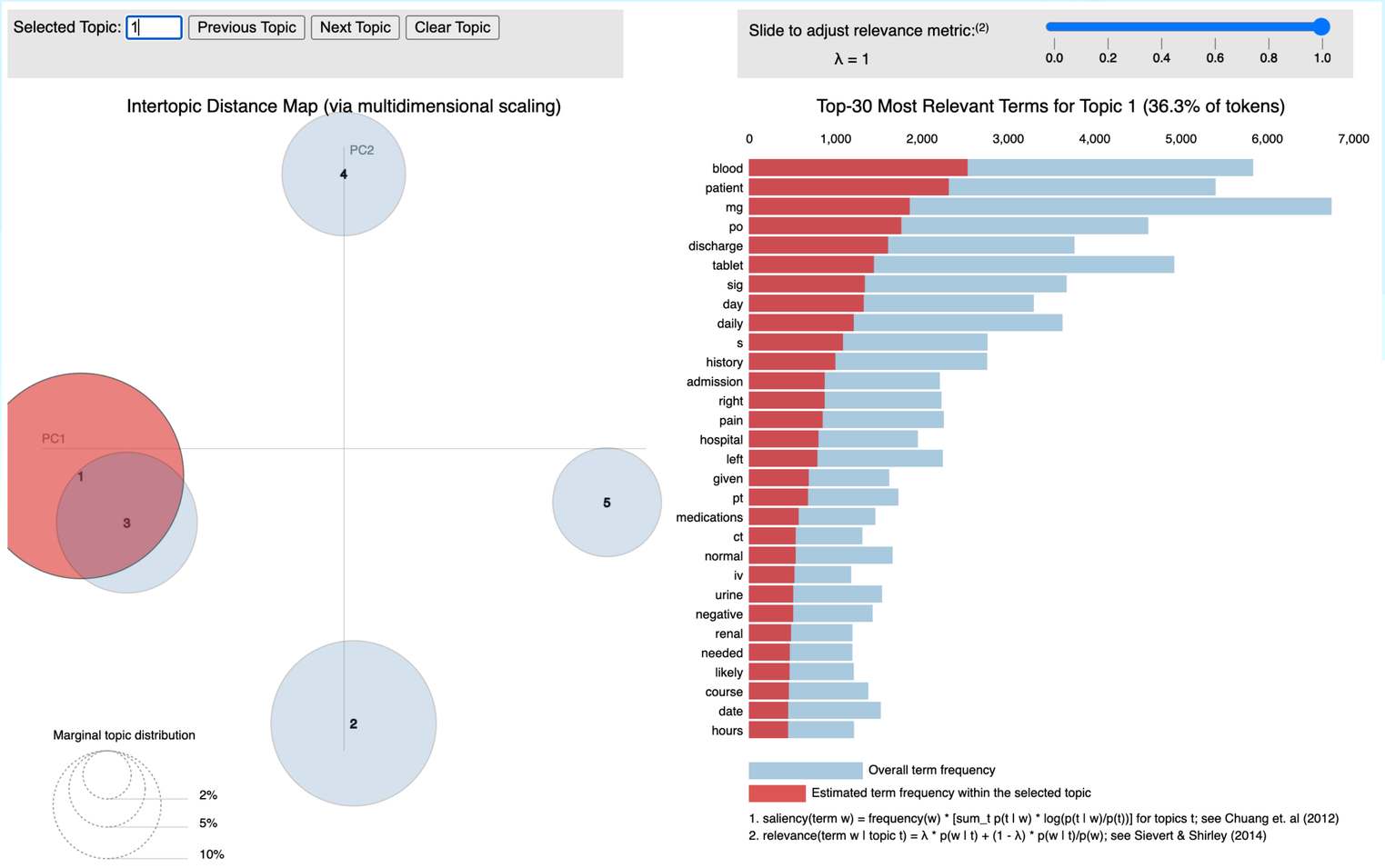
Bonus NLP Tool: Topic Modeling with LDA

- Latent Dirichlet Allocation (LDA) is used to uncover hidden topics in clinical notes.
- The preprocess_texts function removes numbers and tokenizes the texts.
- lda_topic_modeling creates a dictionary and corpus, fits the LDA model, and prints the top words for each topic (the latter task using get_top_words).
- The topics are visualized using pyLDAvis, and the visualization is saved to an HTML file.

```python
from sklearn.cluster import KMeans
import random
import pyLDAvis
import pyLDAvis.gensim_models
from gensim import corpora
from gensim.models import LdaModel
from sklearn.feature_extraction.text import CountVectorizer
import re

def preprocess_texts(texts):
    # Remove numbers from the texts
    texts = [re.sub(r'\d+', '', text) for text in texts]

    # Tokenize and clean the texts
    vectorizer = CountVectorizer(stop_words='english')
    text_matrix = vectorizer.fit_transform(texts)
    tokens = [text.split() for text in texts]
    return tokens, vectorizer

def get_top_words(lda_model, num_words=5):
    top_words = {}
    for idx, topic in lda_model.print_topics(-1):
        words = topic.split(' + ')
        top_words[idx] = ', '.join([word.split('*')[1].strip('"') for word in words[:num_words]])
    return top_words

def lda_topic_modeling(texts, n_topics=5):
    tokens, vectorizer = preprocess_texts(texts)

    # Create a dictionary and corpus for LDA
    dictionary = corpora.Dictionary(tokens)
    corpus = [dictionary.doc2bow(token) for token in tokens]

    # Fit the LDA model
    lda_model = LdaModel(corpus, num_topics=n_topics, id2word=dictionary, random_state=23)

    # Print the top words for each topic
    top_words = get_top_words(lda_model)
    for idx, words in top_words.items():
        print(f"Topic {idx}: {words}")
        print("\n")

    # Visualize the topics
    pyLDAvis.enable_notebook()
    panel = pyLDAvis.gensim_models.prepare(lda_model, corpus, dictionary)
    return panel

texts = ecoli_df['SCISPACY_PROCESSED_TEXT'].tolist()
panel = lda_topic_modeling(texts, n_topics=5)

# Save the visualization to an HTML file
output_filepath = '/Users/jefdewitt/Documents/education/UT-Austin/courses/ai-in-healthcare/module-6/lda_visualization.html'
pyLDAvis.save_html(panel, output_filepath)
print(f"LDA visualization saved to {output_filepath}")
```

Bonus NLP Tool: Topic Modeling with LDA Results

- Latent Dirichlet Allocation (LDA) uncovers hidden topics in clinical notes.
- Topics are grouped based on word frequency.
- The result is an interactive visualization that showcases different topic clusters from LDA analysis.



Key Takeaways

- NLP tools like SciSpacy and LDA help analyze clinical text.
- Salient details from medical notes can easily be identified, organized, and visualized, improving insights into medical conditions.
- Machine learning techniques aid pattern recognition and organization, reducing the burden on providers.
- Future work: Enhancing entity recognition accuracy and expanding topic modeling.