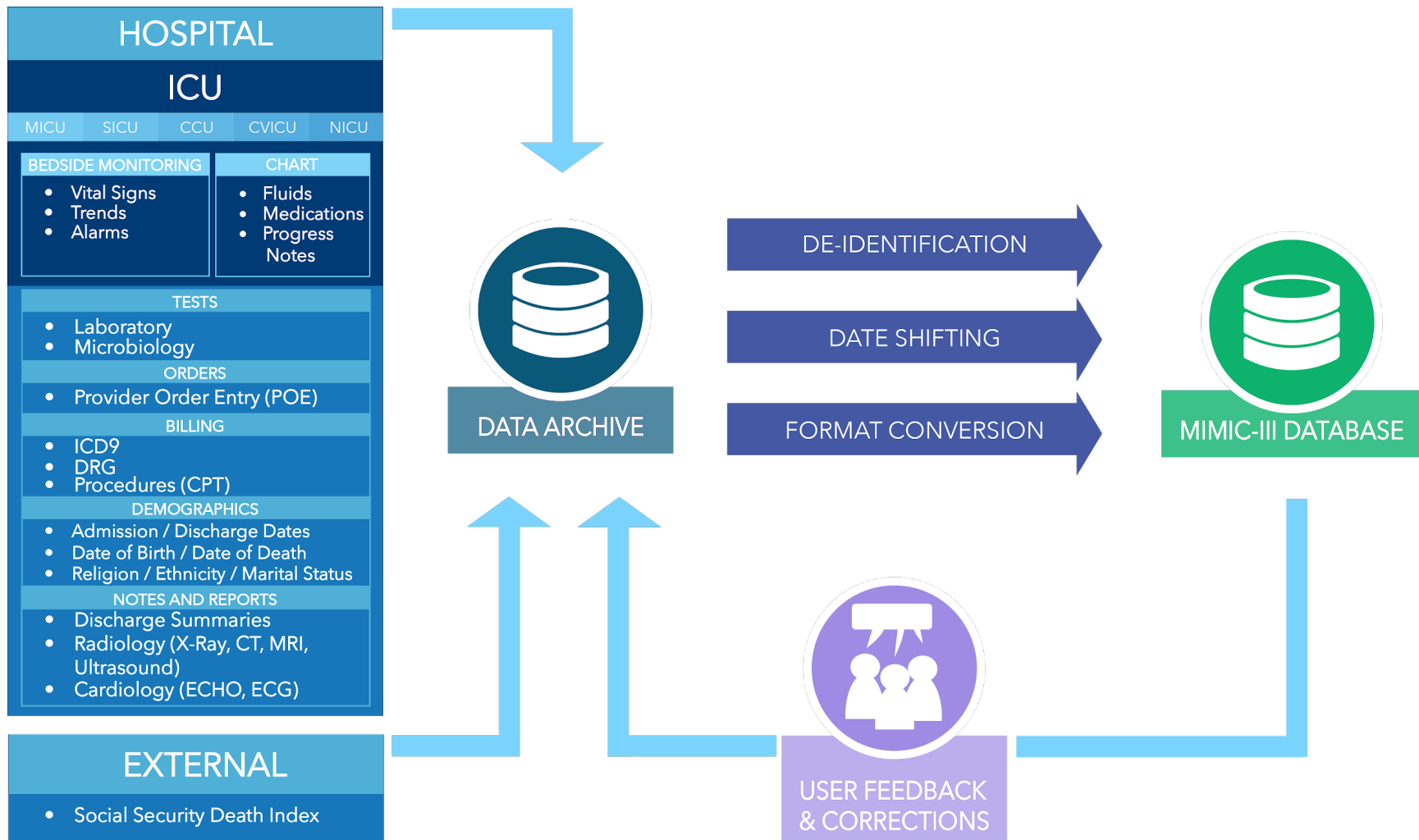


**Top 3 Diseases and
Medications Analysis
with Medical Corpus
Using NLP**

Data

Database - MIMIC III (Medical Information Mart for Intensive Care III)

- A large, single-center database of information related to patients admitted to critical care units at a large tertiary care hospital (by the MIT Lab for Computational Physiology)
- Compared to other corpora: 1. freely available to researchers worldwide; 2. encompasses a diverse and very large population of ICU patients; 3. contains high temporal resolution data including lab results, electronic documentation, and bedside monitor trends and waveforms
- Compared to MIMIC II: anticipated by its developers to be widely used internationally in areas such as academic and industrial research, quality improvement initiatives, and higher education coursework



Significance

Our scientific question contains two parts:

- What are the top three diseases that cause elderly (65+) to go to hospitals?
- What are the most common medications corresponding to those three geriatric diseases?

How it affects human health:

- In the US, nearly 29% of the 46 million community-dwelling elderly live alone
- This study objective may contribute to improving public awareness of the common geriatrics, daily preventions, and the preparation of common medications at home

Rationale

Why we used this corpus:

- MIMIC III has health-related data associated with approximately 60,000 admissions of patients → The sufficient information in this corpus allows us to extract enough clinical notes as our sample and find answers to our scientific question
- MIMIC III was used as the underlying database for the study, *A data-driven approach to optimized medication dosing: a focus on heparin*

Main Method, Implementation, & Results

Generating Study Sample:

- Our corpus is NoteEvents from MIMIC III, which is 4.01 GB in size. Clinical notes for 40,000 randomly selected patients are our sample.

Extracting geriatric patients (older than 65 years old) and Diagnosis and Medication Notes

Method: RegEx

- Rationale or details of application: (in 40,000 notes, the results was around 9000 notes that fit our criteria of >65.)
- Diagnoses = `r"Discharge Diagnosis:(.*?)(?:(?:\r*\n){2})"`

The word cloud features the following terms, categorized by color:

- Heart Failure (Red):** chronic, heart, failure, congestive, acute, coronary, atrial, aortic, diastolic, ventricular, post, aspiration, first, elevation, loss, chf, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired, diastolic, ventricular, post, aspiration, first, elevation, loss, chf, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired.
- Heart Disease (Blue):** and, left, right, with, stenosis, fibrillation, status, fracture, lung, last, the, pulmonary, urinary, copd, hypertension, cancer, congestive, valve, severe, bleeding, tract, anemia, htn, infarction, blood, hyperlipidemia, from, gastrointestinal, ep, acromioclavicular, colon, kidney, diabetes, exacerbation, respiratory, mellitus, myocardial, bilateral, bleed, history, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired.
- Heart Failure Complications (Green):** disease, sepsis, upper, non, hyperlipidemia, blood, kidney, diabetes, exacerbation, respiratory, mellitus, myocardial, bilateral, bleed, history, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired.
- Heart Failure Symptoms (Yellow):** chronic, heart, failure, congestive, acute, coronary, atrial, aortic, diastolic, ventricular, post, aspiration, first, elevation, loss, chf, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired.
- Heart Failure Risk Factors (Purple):** chronic, heart, failure, congestive, acute, coronary, atrial, aortic, diastolic, ventricular, post, aspiration, first, elevation, loss, chf, systolic, type, insufficiency, stage, infection, obstructive, due, repair, name, lower, acquired.

Main Method, Implementation, & Results

Preprocessed the diagnosis Notes:

```
#preprocessing
diag_list = diagnosis_list(nodeList2)
string = ''.join(each for each in diag_list)
wordlist1 = string.lower()
wordlist1 = wordlist1.replace('/n', '')
wordlist1 = wordlist1.replace('diagnosis', '')
wordlist1 = wordlist1.replace('primary', '')
wordlist1 = wordlist1.replace('secondary', '')
tokenizer = RegexpTokenizer(r'[a-z]+')
tokens = tokenizer.tokenize(wordlist1)
tokens = [word for word in tokens if word not in stopwords.words('english')]

#find the top disease
fdist = FreqDist(tokens)
diag_common = fdist.most_common(50)
print(diag_common)
```


Main Method, Implementation, & Results

Word frequency on preprocessed diagnosis notes

```
In [8]: diag_common = fdist.most_common(50)
...: print(diag_common)
[('s', 4224), ('p', 3900), ('disease', 2572), ('failure', 2214), ('hypertension', 1899),
('artery', 1680), ('chronic', 1640), ('acute', 1611), ('coronary', 1438), ('atrial',
1386), ('with', 1382), ('of', 1321), ('left', 1301), ('fibrillation', 1237), ('and',
1210), ('renal', 1161), ('right', 1156), ('heart', 1126), ('pneumonia', 1080), ('aortic',
1029), ('on', 989), ('diabetes', 985), ('to', 946), ('anemia', 855), ('stenosis', 784),
('post', 739), ('hyperlipidemia', 728), ('pulmonary', 697), ('congestive', 621),
('mellitus', 616), ('htn', 608), ('history', 602), ('status', 591), ('bleed', 589),
('cad', 582), ('cancer', 565), ('infection', 553), ('in', 548), ('respiratory', 544),
('x', 517), ('cabg', 516), ('type', 512), ('urinary', 490), ('copd', 479), ('infarction',
460), ('valve', 452), ('fracture', 451), ('myocardial', 435), ('diagnoses', 423),
('hemorrhage', 415)]
```

- Hypertension
- Coronary Artery Disease
- Atrial Fibrillation

Main Method, Implementation, & Results

Filtering the notes with top diseases

```
#create the dataframe for each disease  
df_all = pd.DataFrame({'dia_med':allList})  
dfall_hyp = df_all[df_all['dia_med'].str.contains('hypertension')]  
dfall_cor = df_all[df_all['dia_med'].str.contains('coronary')]  
dfall_af = df_all[df_all['dia_med'].str.contains('atrial fibrillation')]
```

dfall_af	DataFrame	(389, 1)	Column names: dia_med
dfall_cor	DataFrame	(461, 1)	Column names: dia_med
dfall_hyp	DataFrame	(436, 1)	Column names: dia_med

- Coronary Artery Disease: 461 patients
- Hypertension: 436 patients
- Atrial Fibrillation: 389 patients

Main Method, Implementation, & Results

Word frequency on filtered medication notes

Coronary Artery Disease: Aspirin (184 word counts), Docusate Sodium (Docusate 136, Sodium 184), Metoprolol Tartrate (Metoprolol 143, Tartrate 120).

Atrial Fibrillation: the suggested medications are Metoprolol succinate (metoprolol has 124 word counts), Docusate sodium (docusate has 107 word counts, sodium has 150 word counts), and Metoprolol tartrate (tartrate has 105 word counts).

Hypertension: Aspirin (235), Metoprolol (173), HCl (136), Docusate (135)

Method 2: N-Gram

Some diseases name contains more than one word.

n-gram: a contiguous sequence of n items from a given sample of text or speech.

n-gram model: a type of probabilistic language model for predicting the next item in such a sequence in the form of a $(n - 1)$ -order Markov model.

Implementation: N-Gram

```
3 unigram = ngrams(tokens,1)
9 bigrams = ngrams(tokens,2)
0 trigrams = ngrams(tokens,3)
1
2 print("Most common unigram:\n")
3 for key, value in Counter(unigram).most_common():18]:
4     print(str(value) + "\t" + str(key))
5
6 print("Most common bigrams:\n")
7 for key, value in Counter(bigrams).most_common():10]:
8     print(str(value) + "\t" + str(key))
9 print()
0 print("Most common trigrams:\n")
1 for key, value in Counter(trigrams).most_common():5]:
2     print(str(value) + "\t" + str(key))
3 print()
```

Results: N-Gram

Top three diseases:

Coronary Artery Disease

Hypertension

Atrial Fibrillation

Most common unigram:

184	('chronic',)
182	('disease',)
161	('artery',)
161	('failure',)
156	('heart',)
142	('with',)
137	('renal',)
118	('acute',)
116	('and',)
115	('atrial',)
104	('left',)
94	('right',)
92	('coronary',)
91	('fibrillation',)
79	('aortic',)
77	('hypertension',)
73	('pulmonary',)
71	('congestive',)

Most common bigrams:

82	('coronary', 'artery')
80	('heart', 'failure')
71	('artery', 'disease')
68	('congestive', 'heart')
67	('atrial', 'fibrillation')
39	('acute', 'chronic')
35	('chronic', 'renal')
34	('urinary', 'tract')
34	('tract', 'infection')
33	('renal', 'failure')

Most common trigrams:

52	('coronary', 'artery', 'disease')
40	('congestive', 'heart', 'failure')
26	('urinary', 'tract', 'infection')
16	('with', 'rapid', 'ventricular')
15	('stage', 'renal', 'disease')

Method 3: TFIDF

- Bind the top diagnosis and common treatments found
- Calculate the TFIDF of each top medication for each diagnosis related papers.
- Example: if the TFIDF of certain medication is significantly higher than it for other medication for Hypertension therapeutic papers, the medication could be classified as treatments for Hypertension
- Additional corpus is pubmed for this method

$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$

$IDF(t) = \log_{10}(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$

*$TF * IDF = [(\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})] * \log_{10}(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$*

Implementation: TFIDF

- Extract treatment information from PubMed:

```
def searchPubMed(query, num_of_results):
    #PubMed needs your email to warn you for searching for too many articles at once
    Entrez.email = 'yuq2002@med.cornell.edu'
    if num_of_results > 200:
        raise ValueError('Search for less results! - Change the value in the num_of_results in searchPubMed function')
    else:
        handle = Entrez.esearch(db='pubmed',
                                sort='relevance',
                                retmax= num_of_results,
                                retmode='text', # 'xml' , 'text'
                                term=query)

        #print(handle.url)
        results = Entrez.read(handle)
        listOfPMID = results.get('IdList')
        abstractSearched = []
        for each in listOfPMID:
            handle = Entrez.efetch(db="pubmed", id=each ,
                                    rettype="xml", retmode="text", term = 'heart attack')
            records = Entrez.read(handle)
            abstractlist = records['PubMedArticle'][0]['MedLineCitation']['Article']['Abstract']['AbstractText']
            if len(abstractlist) > 1:
                abstract = ''
                for eachsection in abstractlist:
                    abstract += str(eachsection)
                abstractSearched.append(abstract)
            else:
                abstractSearched.append(abstractlist[0])
        return abstractSearched

def list_to_txt(listOfAbstracts, fileName):
    f = open(str(fileName) + '.txt', 'w', encoding = 'utf-8')
    for item in listOfAbstracts:
        f.write("%s\n" % item)
    f.close()
    print('list converted to .txt file')

## Below is where you would change the query
searchquery =searchPubMed("atrial fibrillation medication", 200)
list_to_txt(searchquery, 'AtrialFibrillation')
```

Three text files contain 200 abstracts for disease treatment related papers for top three diseases respectively are generated

Implementation: TFIDF

- Calculate the TFIDF for each word in the documents
- Output the TFIDF result to a txt file
- Search our medication terms to find corresponding TFIDF for each of three documents

```
path = "/Users/yuqing/Documents/Weill Cornell/Term 2/Natural Language Processing/NLP Project/"
disease_file_list = ['AtrialFibrillation.txt', 'Hypertension.txt', 'CoronaryHeartDisease.txt']
my_documents = []
for disease_file in disease_file_list:
    text = open(path + disease_file).read()
    my_documents.append(text)

tokenized_docs = [word_tokenize(doc.lower()) for doc in my_documents]

dictionary = Dictionary(tokenized_docs)
print("Mapping of tokens to indices:\n")
#print(dictionary.token2id)
print("-"*70)

corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
print("Corpus (shows you int representation of the word mapped to frequency):\n")
#print(corpus[10])
print("-"*70)

print("-"*70 + "\nTfidf:\n" + "-"*70)
tfidf = TfidfModel(corpus)

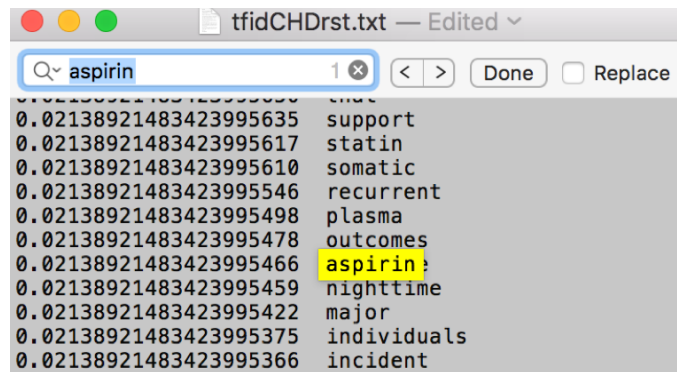
tfidf2=[]
for i in range(len(my_documents)):
    print(my_documents[i][:100] + "\n")
    # sorted in decreasing order (largest to smallest)
    sorted_by_tfidf = sorted(tfidf[corpus[i]], key = lambda x: x[1])[::-1]
    for i in range(len(sorted_by_tfidf)):
        tokenid = sorted_by_tfidf[i][0]
        score = sorted_by_tfidf[i][1]
        if i < 10000:
            #print(str(tokenid) + "\t" + str(dictionary[tokenid]) + "\t" + str(score))
            tfidf1=str(tokenid) + "\t" + str(dictionary[tokenid]) + "\t" + str(score)
            tfidf2.append(tfidf1)

tfidf2 = ''.join(tfidf2)
f = open('/Users/yuqing/Documents/Weill Cornell/Term 2/Natural Language Processing/NLP Project/tfidfCHDrst.txt','w')
f.write(tfidf2)
f.close()
```

Implementation: TFIDF

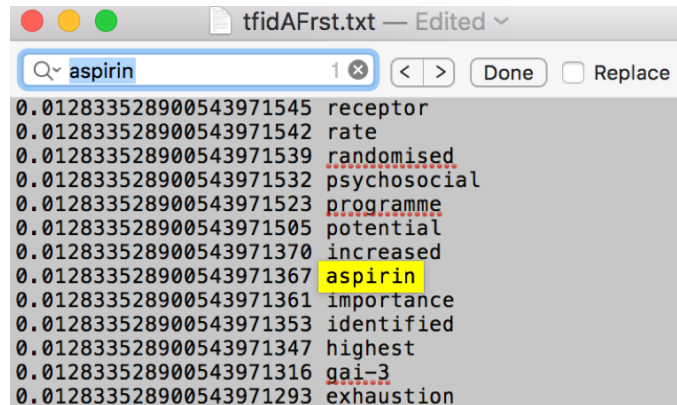
Example of comparason:

- To find which disease is "Aspirin" for
- TFIDF in CAD: 0.0214
- TFIDF in AF: 0.0128
- TFIDF in HPT: none
- Result: Aspirin is used for Coronary Artery Disease and Atrial Fibrillation, but not for Hypertension



A screenshot of a text editor window titled "tfidCHDrst.txt — Edited". The search bar at the top contains "aspirin" and shows "1" result. The search results are listed below, with "aspirin" highlighted in yellow in the 6th row.

Index	Term
0.02138921483423995635	support
0.02138921483423995617	statin
0.02138921483423995610	somatic
0.02138921483423995546	recurrent
0.02138921483423995498	plasma
0.02138921483423995478	outcomes
0.02138921483423995466	aspirin
0.02138921483423995459	nighttime
0.02138921483423995422	major
0.02138921483423995375	individuals
0.02138921483423995366	incident



A screenshot of a text editor window titled "tfidAFrst.txt — Edited". The search bar at the top contains "aspirin" and shows "1" result. The search results are listed below, with "aspirin" highlighted in yellow in the 10th row.

Index	Term
0.012833528900543971545	receptor
0.012833528900543971542	rate
0.012833528900543971539	randomised
0.012833528900543971532	psychosocial
0.012833528900543971523	programme
0.012833528900543971505	potential
0.012833528900543971370	increased
0.012833528900543971367	aspirin
0.012833528900543971361	importance
0.012833528900543971353	identified
0.012833528900543971347	highest
0.012833528900543971316	gai-3
0.012833528900543971293	exhaustion

Results: TFIDF

Common medications are linked to three diseases as:

Coronary Heart Disease:

Aspirin, Docusate Sodium, Metoprolol Tartrate, Acetaminophen

Atrial Fibrillation:

Aspirin, Docusate Sodium, Metoprolol Tartrate, Warfarin

Hypertension:

Docusate Sodium, Metoprolol Tartrate, Furosemide

Method selection & Combination

The final results are selected based on the precisions of results from two methods compared with annotation, literature research, and consultation with medical specialist.

Coronary Artery Disease:

Aspirin, Docusate Sodium, Metoprolol Tartrate, Acetaminophen

Atrial Fibrillation:

Aspirin, Docusate Sodium, Metoprolol Tartrate, Warfarin, Metoprolol succinate

Hypertension:

Aspirin, Docusate Sodium, Metoprolol Tartrate, Furosemide

Conclusion

- Found that that Coronary Artery Disease, Hypertension, and Atrial Fibrillation are most common diseases
- **Coronary Artery Disease:** Aspirin, Docusate Sodium, Metoprolol Tartrate, Acetaminophen
- **Atrial Fibrillation:** Aspirin, Docusate Sodium, Metoprolol Tartrate, Warfarin, Metoprolol succinate
- **Hypertension:** Aspirin, Docusate Sodium, Metoprolol Tartrate, Furosemide
- Studies show lowering helps prevents stroke, heart attacks and kidney problems
- Very Similar word counts for each medication
 - Diagnoses are closely related, which means high chance to erupt simultaneously

We gave up on automated data cleaning
and went with this instead.



References

- Johnson, A. E. W. et al. MIMIC-III, a freely accessible critical care database. Sci. Data 3:160035 doi: 10.1038/sdata.2016.35 (2016)
- Kaplan, D., Berkman, B. and Fitzdale, H. (2016). The Elderly Living Alone. [online] Merck Manual. Available at: <https://www.merckmanuals.com/professional/geriatrics/social-issues-in-the-elderly/the-elderly-living-alone> [Accessed Aug. 2016].
- Ghassemi, M., Richter, S., Eche, I., Chen, T., Danziger, J. and Celi, L. (2018). A data-driven approach to optimized medication dosing: a focus on heparin.
- Physionet.org. (2016). The MIMIC-III Clinical Database. [online] Available at: <https://physionet.org/physiobank/database/mimic3cdb/>