Text Vectorization of Medical Notes

Huang Jingfang

Task

The goal of this challenge is to

propose different approaches to

generate vector representations of texts (tokens) and

assess and compare the suitability of the approach

for this dataset.

Approach

- Evaluate the two datasets
- Apply different ways to tokenize the data and compare the results
- Apply different ways to generate vector representations of texts
- Assess and compare different vectorization methods

CONTENT

Data Visualization

/01

Count the frequency for each category

```
#Separate clinical notes into 3 categories
Cardio = Notes.loc[Notes['category']== 'Cardiovascular / Pulmonary']
Neuro = Notes.loc[Notes['category'] == 'Neurology']
Gastro = Notes.loc[Notes['category']== 'Gastroenterology']
```

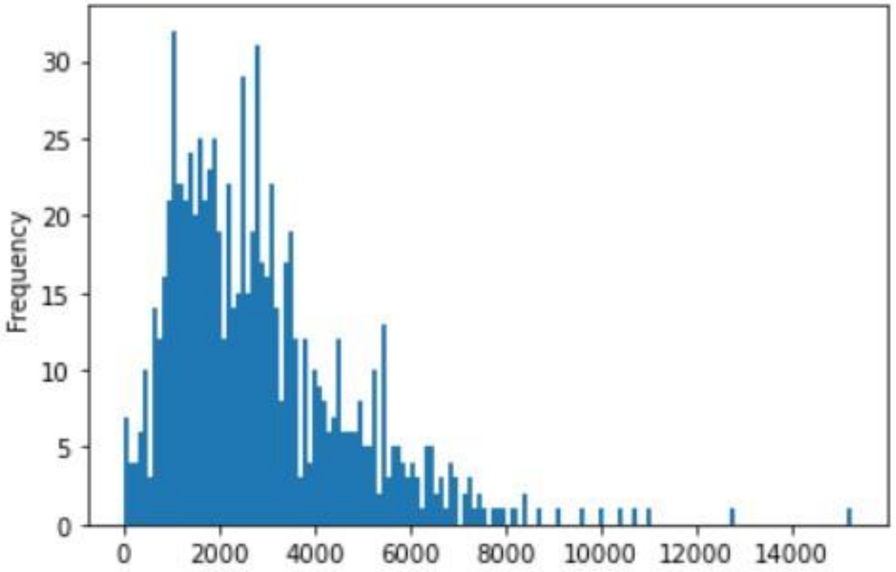
```
Notes.groupby('category').describe()
```

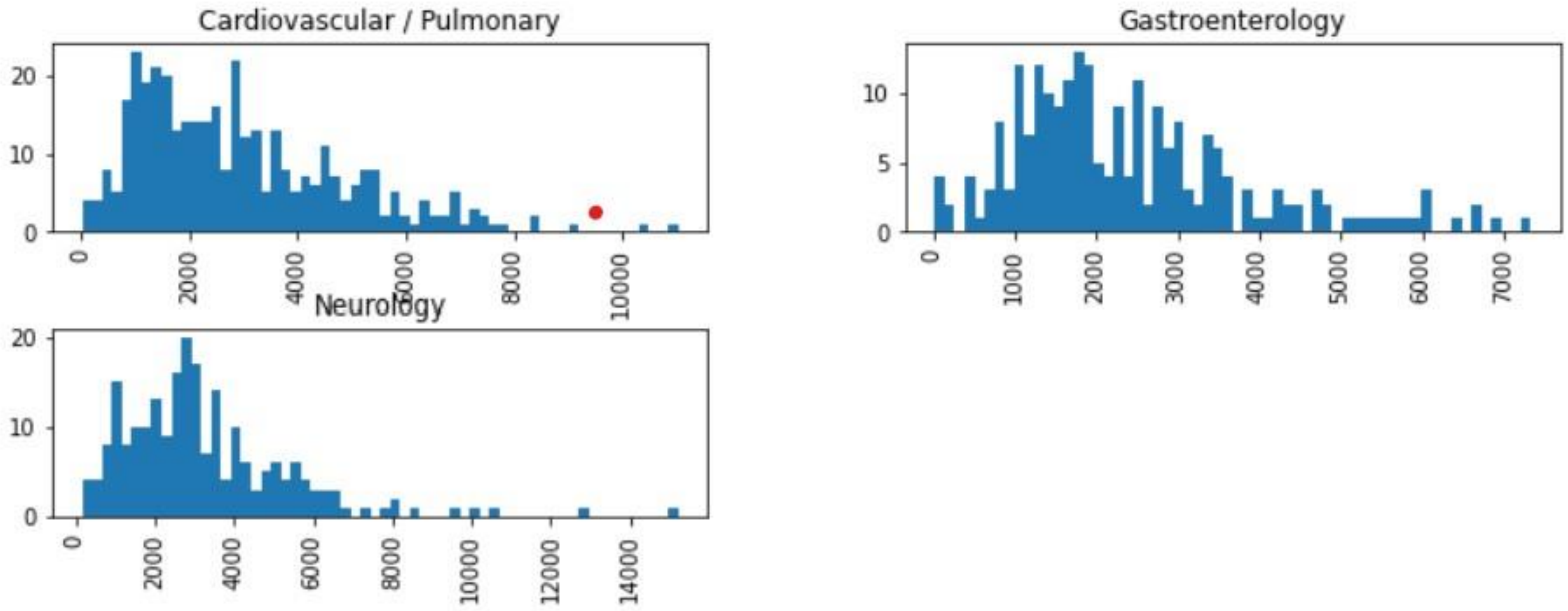
	notes			
	count	unique	top	freq
category				
Cardiovascular / Pulmonary	371	371	DIAGNOSES:,1. Bronchiolitis, respiratory sync...	1
Gastroenterology	224	224	The patient was placed in the left lateral dec...	1
Neurology	223	223	CHIEF COMPLAINT: , "A lot has been thrown at m...	1

Check the Length of the Documents

```
# Attach a new column 'length'
Notes['length'] = Notes['notes'].apply(len)
Notes.head(10)
```

	category	notes	length
0	Cardiovascular / Pulmonary	2-D M-MODE: , ,1. Left atrial enlargement wit...	495
1	Cardiovascular / Pulmonary	1. The left ventricular cavity size and wall ...	1618
2	Cardiovascular / Pulmonary	2-D ECHOCARDIOGRAM,Multiple views of the heart...	759
3	Cardiovascular / Pulmonary	DESCRIPTION:,1. Normal cardiac chambers size....	495
4	Cardiovascular / Pulmonary	2-D STUDY,1. Mild aortic stenosis, widely calc...	662
5	Neurology	CC:, Confusion and slurred speech.,HX , (prima...	3509
6	Cardiovascular / Pulmonary	PREOPERATIVE DIAGNOSES,Airway obstruction seco...	5749
7	Neurology	PROCEDURE: , EEG during wakefulness demonstrat...	1127
8	Neurology	Doctor's Address,Dear Doctor:,This letter is a...	2516
9	Neurology	TIME SEEN: , 0734 hours and 1034 hours.,TOTAL ...	1088




Cardiovascular / Pulmonary
Gastroenterology
Neurology
0
10
20
2000
4000
6000
8000
10000
12000
14000
5
1000
3000
5000
7000

Tokenization

/02

a. Sentence tokenization with split ('.')

```
for records in Notes['notes']:
    tokens = records.split('.')
    print(tokens)
```

```
['2-D M-MODE: , ,1', '  Left atrial enlargement with left atrial diameter of 4', '7 cm', ',2', '  Normal size right and left ventricl
e', ',3', '  Normal LV systolic function with left ventricular ejection fraction of 51%', ',4', '  Normal LV diastolic function', ',
5', '  No pericardial effusion', ',6', '  Normal morphology of aortic valve, mitral valve, tricuspid valve, and pulmonary valve', ',
7', '  PA systolic pressure is 36 mmHg', ',DOPPLER: , ,1', '  Mild mitral and tricuspid regurgitation', ',2', '  Trace aortic and pulm
onary regurgitation', '']
['1', '  The left ventricular cavity size and wall thickness appear normal', '  The wall motion and left ventricular systolic function
appears hyperdynamic with estimated ejection fraction of 70% to 75%', '  There is near-cavity obliteration seen', '  There also appear
s to be increased left ventricular outflow tract gradient at the mid cavity level consistent with hyperdynamic left ventricular systol
ic function', '  There is abnormal left ventricular relaxation pattern seen as well as elevated left atrial pressures seen by Doppler
examination', ',2', '  The left atrium appears mildly dilated', ',3', '  The right atrium and right ventricle appear normal', ',4', '
The aortic root appears normal', ',5', '  The aortic valve appears calcified with mild aortic valve stenosis, calculated aortic valve
area is 1', '3 cm square with a maximum instantaneous gradient of 34 and a mean gradient of 19 mm', ',6', '  There is mitral annular c
alcification extending to leaflets and supportive structures with thickening of mitral valve leaflets with mild mitral regurgitation',
',7', '  The tricuspid valve appears normal with trace tricuspid regurgitation with moderate pulmonary artery hypertension', '  Estima
ted pulmonary artery systolic pressure is 49 mmHg', '  Estimated right atrial pressure of 10 mmHg', ',8', '  The pulmonary valve appea
rs normal with trace pulmonary insufficiency', ',9', '  There is no pericardial effusion or intracardiac mass seen', ',10', '  There i
s a color Doppler suggestive of a patent foramen ovale with lipomatous hypertrophy of the interatrial septum', ',11', '  The study was
somewhat technically limited and hence subtle abnormalities could be missed from the study', ',']
['2-D ECHOCARDIOGRAM,Multiple views of the heart and great vessels reveal normal intracardiac and great vessel relationships', '  Card
```

The output is not satisfactory because '.' is used in the notes for multiple functions, such as after each bullet point.

b. NLTK

```
#word_tokenizer
from nltk.tokenize import word_tokenize
for records in Notes['notes']:
    nltktokens = word_tokenize(records)
    print(nltktokens)
```

```
['2-D', 'M-MODE', ':', ',', ',1', '.', 'Left', 'atrial', 'enlargement', 'with', 'left', 'atrial', 'diameter', 'of', '4.7', 'cm.,2',
'.', 'Normal', 'size', 'right', 'and', 'left', 'ventricle.,3', '.', 'Normal', 'LV', 'systolic', 'function', 'with', 'left', 'ventricul
ar', 'ejection', 'fraction', 'of', '51', '%', '.,4', '.', 'Normal', 'LV', 'diastolic', 'function.,5', '.', 'No', 'pericardial', 'effus
ion.,6', '.', 'Normal', 'morphology', 'of', 'aortic', 'valve', ',', 'mitral', 'valve', ',', 'tricuspid', 'valve', ',', 'and', 'pulmona
ry', 'valve.,7', '.', 'PA', 'systolic', 'pressure', 'is', '36', 'mmHg.', ',', 'DOPPLER', ':', ',', ',1', '.', 'Mild', 'mitral', 'and',
'tricuspid', 'regurgitation.,2', '.', 'Trace', 'aortic', 'and', 'pulmonary', 'regurgitation', '.']
['1', '.', 'The', 'left', 'ventricular', 'cavity', 'size', 'and', 'wall', 'thickness', 'appear', 'normal', '.', 'The', 'wall', 'motio
n', 'and', 'left', 'ventricular', 'systolic', 'function', 'appears', 'hyperdynamic', 'with', 'estimated', 'ejection', 'fraction', 'o
f', '70', '%', 'to', '75', '%', '.', 'There', 'i
e', 'increased', 'left', 'ventricular', 'outflow
'hyperdynamic', 'left', 'ventricular', 'systolic
attern', 'seen', 'as', 'well', 'as', 'elevated',
'left', 'atrium', 'appears', 'mildly', 'dilated.
4', '.', 'The', 'aortic', 'root', 'appears', 'no
ic', 'valve', 'stenosis', ',', 'calculated', 'ao
neous', 'gradient', 'of', '34', 'and', 'a', 'mea
cation', 'extending', 'to', 'leaflets', 'and', '
'with', 'mild', 'mitral', 'regurgitation.,7', '.
gurgitation', 'with', 'moderate', 'pulmonary', '
```

```
#sentence_tokenizer
from nltk.tokenize import sent_tokenize
for records in Notes['notes']:
    sentences = sent_tokenize(records)
    print(sentences)
```

```
['2-D M-MODE: , ,1.', 'Left atrial enlargement with left atrial diameter of 4.7 cm.,2.', 'Normal size right and left ventricle.,3.',
'Normal LV systolic function with left ventricular ejection fraction of 51%.,4.', 'Normal LV diastolic function.,5.', 'No pericardial
effusion.,6.', 'Normal morphology of aortic valve, mitral valve, tricuspid valve, and pulmonary valve.,7.', 'PA systolic pressure is 3
6 mmHg.,DOPPLER: , ,1.', 'Mild mitral and tricuspid regurgitation.,2.', 'Trace aortic and pulmonary regurgitation.']
['1.', 'The left ventricular cavity size and wall thickness appear normal.', 'The wall motion and left ventricular systolic function a
ppears hyperdynamic with estimated ejection fraction of 70% to 75%.', 'There is near-cavity obliteration seen.', 'There also appears t
o be increased left ventricular outflow tract gradient at the mid cavity level consistent with hyperdynamic left ventricular systolic
function.', 'There is abnormal left ventricular relaxation pattern seen as well as elevated left atrial pressures seen by Doppler exam
ination.,2.', 'The left atrium appears mildly dilated.,3.', 'The right atrium and right ventricle appear normal.,4.', 'The aortic root
appears normal.,5.', 'The aortic valve appears calcified with mild aortic valve stenosis, calculated aortic valve area is 1.3 cm squar
e with a maximum instantaneous gradient of 34 and a mean gradient of 19 mm.,6.', 'There is mitral annular calcification extending to l
eaflets and supportive structures with thickening of mitral valve leaflets with mild mitral regurgitation.,7.', 'The tricuspid valve a
ppears normal with trace tricuspid regurgitation with moderate pulmonary artery hypertension.', 'Estimated pulmonary artery systolic p
ressure is 49 mmHg.', 'Estimated right atrial pressure of 10 mmHg.,8.', 'The pulmonary valve appears normal with trace pulmonary insuf
ficiency.,9.', 'There is no pericardial effusion or intracardiac mass seen.,10.', 'There is a color Doppler suggestive of a patent for
amen ovale with lipomatous hypertrophy of the interatrial septum.,11.', 'The study was somewhat technically limited and hence subtle a
bnormalities could be missed from the study.,']
['2-D ECHOCARDIOGRAM,Multiple views of the heart and great vessels reveal normal intracardiac and great vessel relationships.', 'Cardi
ac function is normal.', 'There is no significant chamber enlargement or hypertrophy.', 'There is no pericardial effusion or vegetatio
```

c. Sentence Tokenization using Regular Expressions

```python
import re
for records in Notes['notes']:
    sentences = re.compile('[:.!?] ').split(records)
sentences
```

```
['REASON FOR CONSULTATION',
 ',Abnormal echocardiogram findings and followup',
 'Shortness of breath, congestive heart failure, and valvular insufficiency.,HISTORY OF PRESENT ILLNESS',
 ',The patient is an 86-year-old female admitted for evaluation of abdominal pain and bloody stools',
 'The patient has colitis and also diverticulitis, undergoing treatment',
 'During the hospitalization, the patient complains of shortness of breath, which is worsening',
 'The patient underwent an echocardiogram, which shows severe mitral regurgitation and also large pleural effusion',
 'This consultation is for further evaluation in this regard',
 'As per the patient, she is an 86-year-old female, has limited activity level',
 'She has been having shortness of breath for many years',
 'She also was told that she has a heart murmur, which was not followed through on a regular basis.,CORONARY RISK FACTORS',
 ', History of hypertension, no history of diabetes mellitus, nonsmoker, cholesterol status unclear, no prior history of coronary artery disease, and family history noncontributory.,FAMILY HISTORY',
 ',Nonsignificant.,PAST SURGICAL HISTORY',
 ', No major surgery.,MEDICATIONS',
 ', Presently on Lasix, potassium supplementation, Levaquin, hydralazine 10 mg b.i.d., antibiotic treatments, and thyroid supplementation.,ALLERGIES',
 ',AMBIEN, CARDIZEM, AND IBUPROFEN.,PERSONAL HISTORY:,  She is a nonsmoker',
 'Does not consume alcohol',
 'No history of recreational drug use.,PAST MEDICAL HISTORY',
 ',Basically GI pathology with diverticulitis, colitis, hypothyroidism, arthritis, questionable hypertension, no prior history of coronary artery disease, and heart murmur.,REVIEW OF SYSTEMS,CONSTITUTIONAL',
 'Weakness, fatigue, and tiredness.,HEENT',
```

Thought Process 1:

- need to treat the different types of the tokens separately, e.g. key words are different from punctuations, *stopwords etc.*
- need to remove distractions like upper/lower-case, punctuations, etc.
- need to consider different forms of the same word, e.g. eat, ate, eaten.

c. Sentence Tokenization using Regular Expressions

```
import re
for records in Notes['notes']:
    sentences = re.compile('[:.!?] ').split(records)
sentences
```

```
['REASON FOR CONSULTATION',
 ' ,Abnormal echocardiogram findings and followup',
 ' Shortness of breath, congestive heart failure, and valvular insufficiency.,HISTORY OF PRESENT ILLNESS',
 ' ,The patient is an 86-year-old female admitted for evaluation of abdominal pain and bloody stools',
 ' The patient has colitis and also diverticulitis, undergoing treatment',
 ' During the hospitalization, the patient complains of shortness of breath, which is worsening',
 ' The patient underwent an echocardiogram, which shows severe mitral regurgitation and also large pleural effusion',
 ' This consultation is for further evaluation in this regard',
 ' As per the patient, she is an 86-year-old female, has limited activity level',
 ' She has been having shortness of breath for many years',
 ' She also was told that she has a heart murmur, which was not followed through on a regular basis.,CORONARY RISK FACTORS',
 ', History of hypertension, no history of diabetes mellitus, nonsmoker, cholesterol status unclear, no prior history of coronary artery
disease, and family history noncontributory.,FAMILY HISTORY',
 ' ,Nonsignificant.,PAST SURGICAL HISTORY',
 ', No major surgery.,MEDICATIONS',
 ', Presently on Lasix, potassium supplementation, Levaquin, hydralazine 10 mg b.i.d., antibiotic treatments, and thyroid supplementatio
n.,ALLERGIES',
 ' ,AMBIEN, CARDIZEM, AND IBUPROFEN.,PERSONAL HISTORY:,  She is a nonsmoker',
 ' Does not consume alcohol',
 ' No history of recreational drug use.,PAST MEDICAL HISTORY',
 ' ,Basically GI pathology with diverticulitis, colitis, hypothyroidism, arthritis, questionable hypertension, no prior history of coron
ary artery disease, and heart murmur.,REVIEW OF SYSTEMS,CONSTITUTIONAL',
 ' Weakness, fatigue, and tiredness.,HEENT',
```

Simple Text Processing

By using the stopwords package from NLTK.

```
from nltk.corpus import stopwords
```

```
def text_process(mess):
    """
    1. remove punctuation
    2. remove stop words
    3. return list of clean text words
    """

    nopunc = [char for char in mess if char not in string.punctuation]

    nopunc = ''.join(nopunc)

    return [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
```

```
Notes['notes'].head(5).apply(text_process)
```

```
0    [2D, MMODE, 1, Left, atrial, enlargement, left...
1    [1, left, ventricular, cavity, size, wall, thi...
2    [2D, ECHOCARDIOGRAMMultiple, views, heart, gre...
3    [DESCRIPTION1, Normal, cardiac, chambers, size...
4    [2D, STUDY1, Mild, aortic, stenosis, widely, c...
Name: notes, dtype: object
```

Vectorization

/03

Bag-of-words Model

- To vectorize a corpus with a bag-of-words (BOW) approach, I represent every document from the corpus as a vector whose length is equal to the vocabulary of the corpus, and arrive at a vector mapping of the corpus that enables unique representation of every document.
- Steps:
  - use CountVectorizer to convert a collection of text documents to a matrix of token counts
  - Count number of vocabulary
  - Use BOW to transform the data
  - Calculate sparsity

```python
from sklearn.feature_extraction.text import CountVect
```

```python
#use CountVectorizer to convert a collection of text
bow_transformer = CountVectorizer(analyzer = text_pro
```

```python
#Number of Vocabulary
print(len(bow_transformer.vocabulary_))
```

```
25683
```

```python
bow2 = bow_transformer.transform([note2])
bow3 = bow_transformer.transform([note3])
```

```python
#visualize the vector - bow2
print(bow2)
```

```
(0, 256)	1
(0, 257)	1
(0, 705)	1
(0, 1023)	1
(0, 1607)	1
(0, 1841)	1
(0, 2154)	1
(0, 2215)	1
(0, 4109)	2
(0, 4381)	2
(0, 8696)	1
(0, 8699)	1
(0, 9203)	1
(0, 9422)	1
(0, 9551)	4
(0, 9593)	2
(0, 9601)	7
(0, 9700)	1
(0, 9791)	2
```

Thought Process 2:

```
1 bow_transformer.get_feature_names()[18878] #freq=7
```

'normal'

As "normal" appears many times, we might be able to imply that the patient have multiple indicators that are normal

```
1 bow_transformer.get_feature_names()[9601] #freq=7
```

'appears'

```
1 bow_transformer.get_feature_names()[24994] #freq=6
```

'valve'

These two word are not very meaningful in helping us find out the actual condition of the patient.

```python
notes_bow = bow_transformer.transform(Notes['notes'])
```

```python
print('Shape of Matrix: ', notes_bow.shape)
```

```
Shape of Matrix:  (818, 25683)
```

```python
#check the number of non-zero occurance
notes_bow.nnz
```

```
149202
```

```python
sparsity = (100.0 * notes_bow.nnz/ (notes_bow.shape[0] * notes_bow.shape[1]))
sparsity
```

```
0.7101916949240158
```

The sparsity shows that there are many unique words in the document.

Normalization

Normalization with TF-IDF

```
In [35]: from sklearn.feature_extraction.text import TfidfTransformer
```

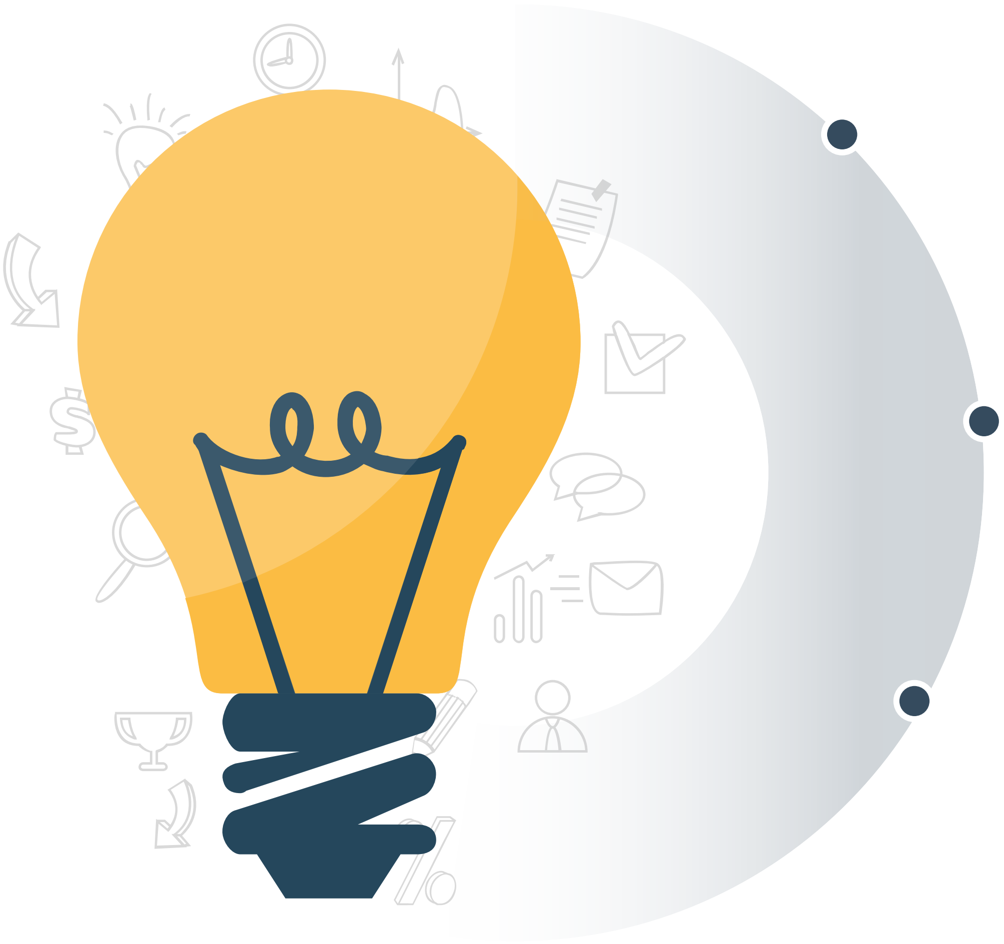
```
In [36]: tfidf_transformer = TfidfTransformer().fit(notes_bow)
```

```
In [37]: tfidf2 = tfidf_transformer.transform(bow2)
```

```
In [38]: print(tfidf2)
```

```
  (0, 25397)	0.024176928565806725
  (0, 25292)	0.08014153513607941
  (0, 25113)	0.20357199948992044
  (0, 25101)	0.04853946535359795
  (0, 24994)	0.2675714660566556
  (0, 24512)	0.11776938228529918
  (0, 24366)	0.06510583686328861
  (0, 24337)	0.13848083796142804
  (0, 24005)	0.0763183173397755
  (0, 24001)	0.06571750591586736
  (0, 23825)	0.08281427168981095
  (0, 23727)	0.1444984328250693
```

Thought Process 3:



Other ways of vectorization?

- Frequency
- One-hot
- TF-IDF

Different ways of implementing them?
Available models? Libraries?

- Scikit-Learn
- Gensim
- NLTK

Note

- **NLTK** offers many methods that are especially well-suited to text data, but is a big dependency.

- **Scikit-Learn** was not designed with text in mind, but does offer a robust API and many other conveniences (which we'll explore later in this chapter) particularly useful in an applied context.

- **Gensim** can serialize dictionaries and references in matrix market format, making it more flexible for multiple platforms. However, unlike Scikit-Learn, Gensim doesn't do any work on behalf of your documents for tokenization or stemming.

Resources: O'REILLY

New Tokenization Function

```
# Tokenization function
"""
1. lightweight normalization
2. strip punctuation
3. set text to lowercase
4. feature reduction using snowball stemmer
"""
def tokenize(text):
    stem = nltk.stem.SnowballStemmer('english')
    text = text.lower()

    for token in nltk.word_tokenize(text):
        if token in string.punctuation: continue
        yield stem.stem(token)
```

Frequency vectors

using NLTK ¶

```
import nltk
```

```
from collections import defaultdict

def fvectorize_nltk(doc):
    features = defaultdict(int) #specify that a 0 should be returned, create a simple counting dictionary
    for token in tokenize(doc):
        features[token] += 1
    return features

fvectors_nltk = map(fvectorize_nltk, Notes['notes'])
```

```
print(fvectors_nltk)
```

```
<map object at 0x00000266AF0469A0>
```

```
print(list(fvectors_nltk))
```

```
[defaultdict(<class 'int'>, {'2-d': 1, 'echocardiogram': 1, 'multipl': 1, 'view': 1, 'of': 1, 'the': 8, 'heart': 1, 'and':
2, 'vessel': 2, 'reveal': 2, 'normal': 7, 'intracardiac': 1, 'relationship': 1, 'cardiac': 1, 'function': 1, 'is': 8, 'the
o': 2, 'signific': 1, 'chamber': 1, 'enlarg': 1, 'or': 2, 'hypertrophi': 1, 'pericardi': 1, 'effus': 1, 'veget': 1, 'seen'
r': 1, 'interrog': 1, 'includ': 1, 'color': 1, 'flow': 2, 'imag': 1, 'system': 1, 'venous': 2, 'return': 2, 'to': 3, 'righ
um': 2, 'with': 2, 'tricuspid': 1, 'inflow': 2, 'pulmonari': 2, 'outflow': 1, 'at': 1, 'valv': 2, 'left': 1, 'interatri':
1, 'intact': 1, 'mitral': 1, 'ascend': 1, 'aorta': 2, 'are': 1, 'aortic': 2, 'trileaflet': 1, 'coronari': 1, 'arteri': 1,
'be': 1, 'in': 1, 'their': 1, 'origin': 1, 'arch': 1, 'left-sid': 1, 'patent': 1, 'descend': 1, 'pulsatil': 1}), defaultdi
nt'>, {'descript': 1, ',1.': 3, 'normal': 4, 'cardiac': 1, 'chamber': 1, 'size.,2': 1, 'left': 1, 'ventricular': 1, 'size.
v': 2, 'systol': 2, 'function': 1, 'eject': 2, 'fraction': 2, 'estim': 2, 'around': 2, '60': 2, '.,4': 1, 'aortic': 1, 'va
n': 3, 'with': 3, 'good': 3, 'motion.,5': 1, 'mitral': 2, 'motion.,6': 1, 'tricuspid': 2, 'motion.,7': 1, 'no': 1, 'perica
fus': 1, 'or': 1, 'intracardiac': 1, 'masses.': 1, 'doppler': 1, 'trace': 2, 'regurgitation.,2': 1, 'regurgitation.': 1, '
'function.,2': 1}), defaultdict(<class 'int'>, {'2-d': 1, 'study,1': 1, 'mild': 6, 'aortic': 3, 'stenosi': 1, 'wide': 1, '
'minim': 1, 'restricted.,2': 1, 'left': 4, 'ventricular': 2, 'hypertrophi': 1, 'but': 3, 'normal': 5, 'systol': 2, 'functi
'moder': 2, 'biatrial': 2, 'enlargement.,4': 1, 'right': 3, 'ventricle.,5': 1, 'appear': 1, 'of': 1, 'the': 1, 'tricuspid'
```

Frequency vectors

using CountVectorizer from sklearn

The CountVectorizer transformer from the sklearn.feature_extraction model has its own internal tokenization and normalization methods.
Each individual document is transformed into a sparse array whose index tuple is the row (the document ID) and the token ID from the dictionary, and whose value is the count.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
fvectors_cv = vectorizer.fit_transform(Notes['notes'])
```

```
print(fvectors_cv)
```

```
(0, 7938)	1
(0, 7195)	4
(0, 1721)	2
(0, 4630)	1
(0, 13431)	2
(0, 3904)	1
(0, 8592)	3
(0, 2906)	1
(0, 8436)	4
(0, 11265)	1
(0, 10661)	1
(0, 1312)	4
(0, 13099)	1
(0, 7461)	2
(0, 12038)	2
```

* Vectors can become extremely sparse, particularly as vocabularies get larger, which can have a significant impact on the speed and performance of machine learning models.

Study notes: For very large corpora, it is recommended to use the Scikit-Learn *HashingVectorizer*, which uses a hashing trick to find the token string name to feature index mapping. This means it uses very low memory and scales to large datasets as it does not need to store the entire vocabulary and it is faster to pickle and fit since there is no state. However, there is no inverse transform (from vector to text), there can be collisions, and there is no inverse document frequency weighting.

One-Hot Encoding

using NLTK

A dictionary whose keys are tokens and whose value is True.

```
def ohvectorize_nltk(doc):
    return {
        token: True
        for token in tokenize(doc) #remember
    }

ohvectors_nltk = map(ohvectorize_nltk, Notes[
```

```
#visualize the vectors
print(list(ohvectors_nltk))
```

```
[{'2-d': True, 'm-mode': True, ',1.': True, 'left
'4.7': True, 'cm.,2': True, 'normal': True, 'size
e, 'function': True, 'ventricular': True, 'eject'
ue, 'no': True, 'pericardi': True, 'effusion.,6':
True, 'pulmonari': True, 'valve.,7': True, 'pa':
d': True, 'regurgitation.,2': True, 'trace': True
viti': True, 'size': True, 'and': True, 'wall': T
'function': True, 'hyperdynam': True, 'with': Tru
e, '75': True, 'there': True, 'is': True, 'near-c
'outflow': True, 'tract': True, 'gradient': True,
ue, 'pattern': True, 'as': True, 'well': True, 'e
n.,2': True, 'atrium': True, 'mild': True, 'dilat
ot': True, 'normal.,5': True, 'valv': True, 'calc
'squar': True, 'a': True, 'maximum': True, 'insta
nular': True, 'calcif': True, 'extend': True, 'le
True, 'tricuspid': True, 'trace': True, 'regurgit
rue, 'mmhg': True, '10': True, 'mmhg.,8': True, '
ntracardiac': True, 'mass': True, 'seen.,10': Tru
pomat': True, 'hypertrophi': True, 'interatri': T
```

using Scikit-Learn

One-hot encoding is implemented with the Binarizer transformer in the preprocessing module. The Binarizer takes only numeric data, so the text data must be transformed into a numeric space using the CountVectorizer ahead of one-hot encoding. The Binarizer class uses a threshold value (0 by default) such that all values of the vector that are less than or equal to the threshold are set to zero, while those that are greater than the threshold are set to 1. Therefore, by default, the Binarizer converts all frequency values to 1 while maintaining the zero-valued frequencies.

```
from sklearn.preprocessing import Binarizer

freq = CountVectorizer()
notes = freq.fit_transform(Notes['notes'])

onehot = Binarizer()
notes = onehot.fit_transform(notes.toarray()) #converts the sparse matrix representation to a dense one
```

```
notes
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
#check the size of array
print("Number of rows: ", len(notes))
print("Number of columns: ", len(notes[0]))
```

```
Number of rows:  818
Number of columns:  13590
```

Thought Process 4:

- One-hot encoding represents similarity and difference at the document level, but because all words are rendered **equidistant**, it is not able to encode per-word similarity.

- Moreover, because all words are equally distant, word form becomes incredibly important(solved by stemmer); the tokens “try” and “attempt” will be equally distant from unrelated tokens like “red” or “bicycle”.

TF-IDF

Justification:

- The bag-of-words representations does not take into account the **context** of the corpus. A better approach would be to consider the **relative frequency** or rareness of tokens in the document against their frequency in other documents.

- TF–IDF normalizes the frequency of tokens in a document with respect to the rest of the corpus. This encoding approach accentuates terms that are very relevant to a specific instance.

TF-IDF

Scikit-Learn

A default tokenization and preprocessing method is applied

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf  = TfidfVectorizer()
tfidfcorpus = tfidf.fit_transform(list(Notes['notes']))
```

```python
tfidfcorpus
```

```
<818x13590 sparse matrix of type '<class 'numpy.float64'>'
	with 170131 stored elements in Compressed Sparse Row format>
```

The vectorizer returns a sparse matrix representation where each key is a document and term pair and the value is the TF-IDF score.

One benefit of TF–IDF is that it naturally addresses the problem of stopwords as well. This biases the TF–IDF model toward moderately rare words.

Problem encountered:

using NLTK

use the TextCollection class.

```
from nltk.text import TextCollection

def tfidfvectorize_nltk(corpus):
    corpus = [tokenize(doc) for doc in corpus]
    texts  = TextCollection(corpus)

    for doc in corpus:
        yield {
            term: texts.tf_idf(term, doc)
            for term in doc
        }
```

```
tfidfvectors = map(tfidfvectorize_nltk, list(Notes['notes']))
```

Question : how to interpret the output here?

```
print(list(tfidfvectors))
```

```
e_nltk at 0x00000266AF43C5F0>, <generator object tfidfvectorize_nltk at 0x00000266AF43C660>, <generator object tfidfvectorize_nltk at
0x00000266AF43C6D0>, <generator object tfidfvectorize_nltk at 0x00000266AF43C740>, <generator object tfidfvectorize_nltk at 0x00000266
AF43C7B0>, <generator object tfidfvectorize_nltk at 0x00000266AF43C820>, <generator object tfidfvectorize_nltk at 0x00000266AF43C890>,
<generator object tfidfvectorize_nltk at 0x00000266AF43C900>, <generator object tfidfvectorize_nltk at 0x00000266AF43C970>, <generator
object tfidfvectorize_nltk at 0x00000266AF43C9E0>, <generator object tfidfvectorize_nltk at 0x00000266AF43CA50>, <generator object tfi
dfvectorize_nltk at 0x00000266AF43CAC0>, <generator object tfidfvectorize_nltk at 0x00000266AF43CB30>, <generator object tfidfvectoriz
e_nltk at 0x00000266AF43CBA0>, <generator object tfidfvectorize_nltk at 0x00000266AF43CC10>, <generator object tfidfvectorize_nltk at
0x00000266AF43CC80>, <generator object tfidfvectorize_nltk at 0x00000266AF43CCF0>, <generator object tfidfvectorize_nltk at 0x00000266
```

Solve the problem

```python
from nltk.text import TextCollection

def tfidfvectorize_nltk(corpus):
    print(corpus)
    corpus = [tokenize(doc) for doc in corpus]
    print(corpus)

    texts  = TextCollection(corpus)

    return_list = []
    for doc in corpus:
        a = {
            term: texts.tf_idf(term, doc)
            for term in doc
        }
        return_list.append(a)
    return return_list
```

```python
import sklearn
vectorizer = sklearn.feature_extraction.text.TfidfVectorizer()
X = vectorizer.fit_transform(Notes['notes'].values)
print(X)
vectorizer.vocabulary_
```

```
(0, 12511)    0.13604710624600735
(0, 10348)    0.231544102918857l
(0, 7853)     0.07198140887308489
(0, 4174)     0.14180632907289684
(0, 7921)     0.11261058800304297
(0, 471)      0.12186554482834694
(0, 6865)     0.041417102002462534
(0, 9684)     0.06675779065337811
(0, 8842)     0.16012179309170874
(0, 9922)     0.1917569334654704
```

Study Notes:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

TF-IDF Score: Term Frequency — Inverse Document Frequency

Limitations

/04

Summary and Comparisons

Vectorization Method	Functions	Considerations	
Frequency Vectors	Counts term frequencies	Most frequent words may not always be informative	Key information of patients might be removed
One-Hot Encoding	Binarizes term occurrence (0,1)	All words being equidistant	Can be used in neural networks
TF-IDF	Normalizes term frequencies across documents	Moderately frequent terms may not be representative of document topics	Some semantic meanings are included

Thought Process 5:

- However, all the vectorization I have learnt and tried so far either simply count the frequency of words or based on the relative frequency in the document.

- It will be more meaningful if I apply a model that is able to contextualize word representation. And this is especially needed in this task, as medical notes is very domain specific.

- And I think this is also the beauty of NLP! After some research, I found out some possible models that might help me with this task.

Possible Models

Try to apply ELMo but the package is huge

Contextualized Word Representation - ELMo

```
import tensorflow_hub as hub
import tensorflow as tf
```

```
x = ["Roasted ants are a popular snack in Columbia","it is fine"]

# Extract ELMo features
embeddings = elmo(x, signature="default", as_dict=True)["elmo"]

embeddings.shape
```

THANKS