1.Business Problem

Open-i (Open Access Biomedical Image Search Engine) service of the National Library of Medicine enables search and retrieval of abstracts and images (including charts, graphs, clinical images, etc.) from the open source literature, and biomedical image collections. Searching may be done using text queries as well as query images. Open-i provides access to over 3.7 million images from about 1.2 million PubMed Central® articles; 7,470 chest x-rays with 3,955 radiology reports; 67,517 images from NLM History of Medicine collection; and 2,064 orthopedic illustrations.

2. DATASET

This dataset have radiology reports for the corresponding chest x ray images from indiana university.

- In dataset images are available in png format
- In dataset reports are available in xml format.

- Each xml file have the report for the correspondings chest xray of patients.
- Important info is that more than one image is associated for one reports.

3. SOURCE OF DATA

- Main data source: https://openi.nlm.nih.gov/
- I also refer a source of kaggle to project the frontal and lateral image which you see later in eda

source of data:https://www.kaggle.com/raddar/chest-xrays-indiana-university

4. Problem statement

We have to generate the medical reports(impressions) for given chest xrays of patient.

5. Constraints:

1. Interpretability of model or result
2. no latency requirement.
3. Model prediction should be highly accurate. because its depend on the judement to save a person life.

6. Performance metrics

1. Bleu score for sentence generation evaluation

Import all imoprtant models

In [1]:

```
import numpy as np
import pickle
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import xml.etree.ElementTree as ET
import os
import cv2
import tensorflow as tf
from wordcloud import WordCloud
import re
from collections import defaultdict
import itertools
from collections import Counter
import warnings
warnings.filterwarnings("ignore")



from wordcloud import WordCloud, STOPWORDS
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
 from sklearn.manifold import TSNE
 import nltk
 import matplotlib.pyplot as plt
 nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[1]:

```
True
```

Mount the Drive

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Set the path of the directory

In [3]:

```
path='/content/drive/My Drive/ecgen-radiology'
```

Retreive all the findings and impression for images from xml files

In [4]:

```
columns = ["Path", "image_caption","findings", "impression"]
data_frame = pd.DataFrame(columns = columns)
#list files from Directory
for i in tqdm(os.listdir(path)): # for each xml file.
  if i.endswith('.xml'): # check file end with xml extension.
    file_n = path + '/' + i #give the name to file corresponding to path.
    tree= ET.parse(file_n) # here we parse the file.
    root= tree.getroot() # get the root of the xml file.
    img_list = set()# take the unique values in image_list
    cap_list = set()# take the unique values in caption_list
    for parent in tree.findall("parentImage"):# get the parent image
        img = parent.attrib['id']+".png"# here we get the images id
        cap_list.add('' if parent.find('caption').text is None else parent.find('caption').text)# g
et the captions of images
        img_list.add(img)# add images in image list
    findings = tree.find(".//AbstractText[@Label='FINDINGS']").text# get the findings from xml fil
es corresponding to images
    impression = tree.find(".//AbstractText[@Label='IMPRESSION']").text# get the impression from x
ml files corresponding to images.
        # add reports and image details to dataframe
    data_frame = data_frame.append(pd.Series([','.join(img_list), ','.join(cap_list),findings, impr
ession],# prepare the dataframe.
                                             index = columns), ignore_index = True)
```

```
100%|██████████| 3955/3955 [31:31<00:00,  2.09it/s]
```

In [5]:

```
data_frame
```

Out[5]:

	Path	image_caption	findings	impression
0	CXR3679_IM-1831-1001.png,CXR3679_IM-1831-2001.png	Chest radiograph, 2 images.	Normal heart. Clear lungs. No pneumothorax. No...	Normal chest exam.
	CXR3681_IM-1833-0001		There are lower lung volumes	

	Path	image_caption	findings	impression
1	CXR3681_IM-1833-0001-0002.png,CXR3681_IM-1833-...	Xray Chest PA and Lateral	There are lower lung volumes. There is central...	No acute abnormality identified.
2	CXR368_IM-1832-1001.png,CXR368_IM-1832-2001.png	Chest, 2 views, XXXX XXXX	Cardiomediastinal silhouette and pulmonary vas...	No acute cardiopulmonary findings.
3			Borderline heart size. Worsening central vascu...	Manifestations of decompensated congestive hea...
4	CXR3665_IM-1823-1001.png,CXR3665_IM-1823-2001.png	Chest radiographs, 2 XXXX and lateral	Heart size within normal limits. Negative for ...	1. No acute abnormality. 2. No evidence of pul...
...	...	...	...	...
3950	CXR187_IM-0563-2001.png,CXR187_IM-0563-1001.png	Frontal and lateral chest on XXXX XXXX.	Normal heart size. Stable tortuous aorta. No p...	Unchanged exam without acute abnormality.
3951	CXR1887_IM-0575-1001.png	Xray Chest PA and Lateral	The lungs are clear. There is no pleural effus...	Senescent changes no acute pulmonary disease.
3952	CXR185_IM-0551-2001.png,CXR185_IM-0551-1001.png	Xray Chest PA and Lateral	The heart is normal in size. The mediastinum i...	No acute disease.
3953	CXR1851_IM-0553-1001.png,CXR1851_IM-0553-2001.png	Xray Chest PA and Lateral	Heart size is normal. No focal airspace consol...	No acute cardiopulmonary findings. .
3954	CXR1863_IM-0558-1001.png,CXR1863_IM-0558-3001.png	Xray Chest PA and Lateral	Heart size is mildly enlarged. Tortuous aorta....	1. Low volume study without acute process. 2. ...

3955 rows × 4 columns

Fill the empty values in findings and impression

In [6]:

```
data_frame['findings'] = data_frame['findings'].fillna('No Findings')
data_frame['impression'] = data_frame['impression'].fillna('No Impression')
```

In [8]:

```
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3955 entries, 0 to 3954
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Path           3955 non-null   object
 1   image_caption  3955 non-null   object
 2   findings       3955 non-null   object
 3   impression     3955 non-null   object
dtypes: object(4)
memory usage: 123.7+ KB
```

In [9]:

```
data_frame.Path.describe()
```

Out[9]:

```
count     3955
unique    3852
top           
freq       104
Name: Path, dtype: object
```

Drop all the paths which have nan values

In [10]:

```
data_frame.replace("", float("NaN"), inplace=True)# replace all the blank spaces with NAN.
```

In [11]:

```
data_frame.dropna(subset = ["Path"], inplace=True)#and then drop the nan values.
```

In [12]:

```
data_frame.shape
```

Out[12]:

```
(3851, 4)
```

Unique values in data

In [13]:

```
print("Unique images are {} ".format(len(data_frame.Path.unique())))
print("Unique Caption are {} ".format(len(data_frame.image_caption.unique())))
print("Unique findings are {} ".format(len(data_frame.findings.unique())))
print("Unique impression are {} ".format(len(data_frame.impression.unique())))
```

```
Unique images are 3851
Unique Caption are 697
Unique findings are 2554
Unique impression are 1771
```

EDA ON CHEST X-RAY IMAGES

Plot the image findings and impression corresponding to chest x-ray image.

In [ ]:

```
def plot_image(x, number_of_images, response):
  cnt= 1 # intialize the count with 1.
  image= plt.figure(figsize=(10,20))# plot the figure.
  if response == 'REPORTS':# check response is finding

    for i in x['Path'].values[95:100]:# for each PNG.
      findings = list(x['findings'].loc[x['Path'] == i].values)# get the findings
      impression= list(x['impression'].loc[x['Path'] == i].values)# get the findings
      reports=findings+impression
      img = cv2.imread(i)# read the image
      axis_k= image.add_subplot(number_of_images, 2 , cnt , xticks=[], yticks=[])# here we add the
axis in the subplots.
      axis_k.imshow(img)# show the images
      cnt += 1# increase the count value
      axis_k= image.add_subplot(number_of_images, 2, cnt)# here we add the multiple images
      plt.axis('off')
      axis_k.plot()
      axis_k.set_xlim(0,1)
      axis_k.set_ylim(0, len(reports))
      for i, f in enumerate(reports):# here we try to append the labels infront of chest x-ray imag
es.
        axis_k.text(0,i,f,fontsize=20)
      cnt += 1
    plt.show()
  else:
    print('Enter a valid String')
```

In [ ]:

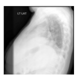
```
plot_image(data_frame, 5, 'REPORTS')
```

Possible area of pneumonitis right lower lobe.

There may be a subtle airspace opacity in the right base near the midclavicular line. There is no pleural effusion or pneumothorax. The heart and mediastinum are normal. The skeletal structures are normal.

No acute cardiopulmonary abnormality.

The lungs are clear, and without focal airspace opacity. The cardiomediastinal silhouette is stable from prior exam. There is no pneumothorax or large pleural effusion. Mediastinal surgical clips are again noted.



No acute cardiopulmonary abnormality.

The lungs are clear, and without focal airspace opacity. The cardiomediastinal silhouette is stable from prior exam. There is no pneumothorax or large pleural effusion. Mediastinal surgical clips are again noted.

No acute cardiopulmonary abnormalities. Specifically, no evidence of active tuberculosis.

The heart is normal in size and contour. There is no mediastinal widening. No focal airspace disease. Left upper lobe granuloma. No evidence of active tuberculosis. Stable chronic blunting of the right costophrenic XXXX. No pneumothorax. The XXXX are intact.

No acute cardiopulmonary abnormalities. Specifically, no evidence of active tuberculosis.

The heart is normal in size and contour. There is no mediastinal widening. No focal airspace disease. Left upper lobe granuloma. No evidence of active tuberculosis. Stable chronic blunting of the right costophrenic XXXX. No pneumothorax. The XXXX are intact.

check the height and weight of image because each image have different size.

In [ ]:

```
#Here we load the widths and heights of each image.
height= [] #Here we get the height of image.
widhts= [] #Here we get the widths of image.
for k in np.unique(data_frame['Path'].values): #Here we get the values of chest x-ray images.
  image= cv2.imread(k) #read the images.
  height.append(image.shape[0]) #Here we get the height of image.
  widhts.append(image.shape[1]) #Here we get the widths of image.
```

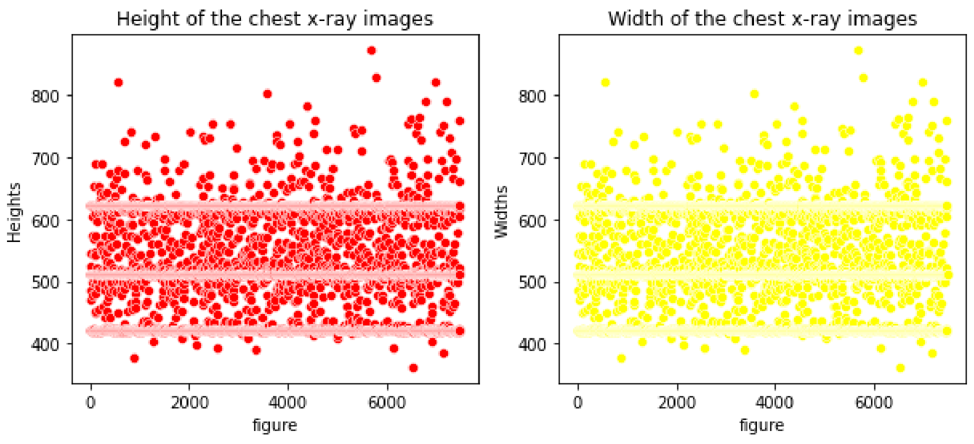
Plot the figure of heights and weights of chest x-ray images.

In [ ]:

```
plt.figure(figsize=(10,4))# here we plot the figure.
plt.subplot(121)# then initialize the first plot
plt.title('Height of the chest x-ray images')# title
plt.ylabel('Heights')# y axis label
plt.xlabel('figure')# x axis label
sns.scatterplot(range(len(height)), height, color= 'red')# seaborn scatter plot for height.
plt.subplot(122)# then initialize the second plot.
plt.title('Width of the chest x-ray images')# title
plt.ylabel('Widths')# y axis label
plt.xlabel('figure')# x axis label
sns.scatterplot(range(len(widhts)), height, color= 'yellow')# seaborn scatter plot for widths.
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb655837f0>
```



Observation:

1. The data points in heights and widths shows us that the chest x-ray images have different heights and widths.

Check the shape of the dataframe

In [ ]:

```
print('The shape of data after removing the null values:', data_frame.shape)# here we print the sh
ape of the dataframe after removing all null values.
```

```
The shape of data after removing the null values: (3851, 4)
```

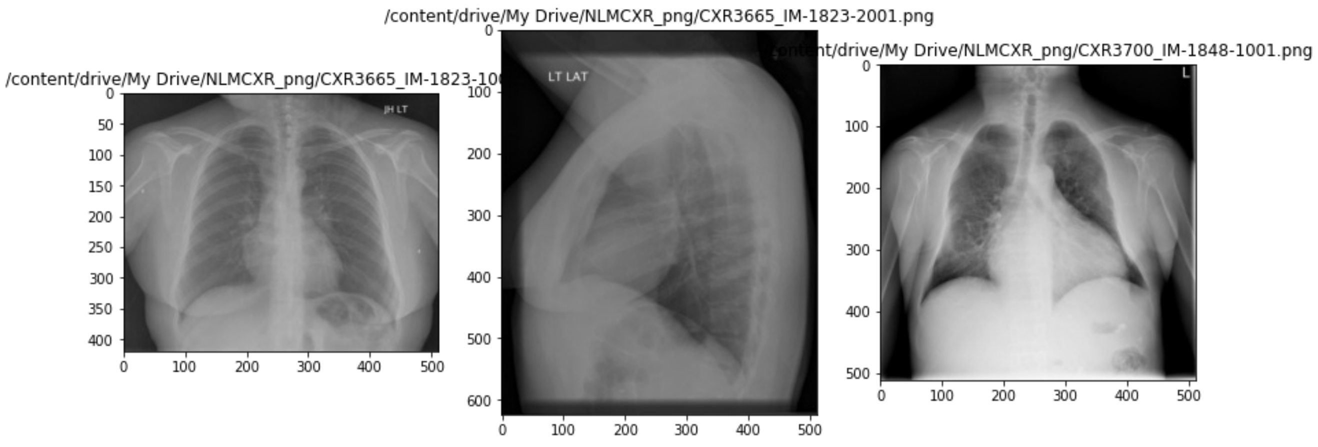
Here we plot the images and then check the findings and impressions for these corresponding images.

In [ ]:

```
plt.figure(figsize=(14,7))# here we plot the figure taking 14 and 7 as figure size
plt.subplot(131)# first subplot
img1 = cv2.imread(data_frame['Path'].values[6])# read the first image
plt.imshow(img1)# show the first image
plt.title(data_frame['Path'].values[6])# read the title of the image
plt.subplot(132)# second subplot.
img2 = cv2.imread(data_frame['Path'].values[7])# read the second image of the plot.
plt.title(data_frame['Path'].values[7])# title of second image of the plot
plt.imshow(img2)# show the second image.
plt.subplot(133)# Third subplot
img3 = cv2.imread(data_frame['Path'].values[8])# read the third image of the plot.
plt.title(data_frame['Path'].values[8])# read the title of third image
plt.imshow(img3)# show the third image.
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7fe2f77c7d30>
```


/content/drive/My Drive/NLMCXR_png/CXR3665_IM-1823-10
/content/drive/My Drive/NLMCXR_png/CXR3665_IM-1823-2001.png
nt/drive/My Drive/NLMCXR_png/CXR3700_IM-1848-1001.png
JH LT
LT LAT
L

In [ ]:

```
data_frame['findings'].values[6]# Finding for the first image.
```

Out[ ]:

```
'Heart size within normal limits. Negative for focal pulmonary consolidation, pleural effusion, or
pneumothorax. No upper lobe airspace disease or cavitary lesions identified.'
```

In [ ]:

```
data_frame['impression'].values[6]
```

Out[ ]:

```
'1. No acute abnormality. 2. No evidence of pulmonary tuberculosis.'
```

In [ ]:

```
data_frame['findings'].values[7]# Finding for the second image.
```

Out[ ]:

```
'Heart size within normal limits. Negative for focal pulmonary consolidation, pleural effusion, or
pneumothorax. No upper lobe airspace disease or cavitary lesions identified.'
```

In [ ]:

```
data_frame['impression'].values[7]# Finding for the third image.
```

Out[ ]:

```
'1. No acute abnormality. 2. No evidence of pulmonary tuberculosis.'
```

In [ ]:

```
data_frame['findings'].values[8]# Finding for the second image.
```

Out[ ]:

```
'Cardiomegaly is present. This is unchanged. There is mild prominence of the pulmonary vascularity
which is unchanged. No XXXX focal airspace disease is seen. No pleural effusion or pneumothorax is
identified.'
```

In [ ]:

```
data_frame['impression'].values[8]# Finding for the third image.
```

Out[ ]:

```
'1. Cardiomegaly with mild vascular prominence. No change.'
```

Observation:

1. Here we found out that above two images finding is same except for the third one. It means this tells us that above two images are same but the scanned type of images are different and third image is totally different from the above two.
2. Similarly impression from the above two is same and the third one is different.

Note: As far i observed path value of all the images have same name for all the files except the last four digits. So, we can consider them as person ID for that chest X-ray image which seems to be unique for all the images.

Now we try to create two dictionaries which have the keys as the person id and the number of images and findings and impressions for the person.

In [ ]:

```
IMAGES={}# create a dictionary of named image.
FINDINGS={}#create a dictionary of named findings.
IMPRESSION={}# create a dictionary of named Impression.

for image, fin, imp in data_frame.values: # here we have the image, finding ,and impression for
each value in data_frame.
  k=image.split('-')# split the image values with the _.
  k.pop(len(k)-1)# take the len of image and subtract it with one and pop that value.
  k= '-'.join(i for i in k)
  if k not in IMAGES.keys(): # here if images is not in keys of images.
    IMAGES[k]=1 # them put the value for that image is 1
    FINDINGS[k]=fin # and put the corresponding finding to their corresponding person id.
    IMPRESSION[k]= imp # here also put the impression.
  else:
    IMAGES[k] +=1# otherwise  make it +1
    FINDINGS[k] = fin# fill the findings
    IMPRESSION[k] = imp  # fill the impreesion
```

In [ ]:

```
IMAGES['/content/drive/My Drive/NLMCXR_png/CXR911_IM-2417']# check the person id.
```

Out[ ]:

```
2
```

In [ ]:

```
FINDINGS['/content/drive/My Drive/NLMCXR_png/CXR911_IM-2417']# check the findings corresponding to
their person id 2.
```

```
check person id 1.
```

Out[ ]:

```
'Heart size within normal limits and cardiomediastinal contours are normal. Lungs are clear bilate
rally. No focal consolidations. No pleural effusions or pneumothorax. Bony structures and soft
tissues are unremarkable.'
```

In [ ]:

```
IMPRESSION['/content/drive/My Drive/NLMCXR_png/CXR911_IM-2417']
```

Out[ ]:

```
'No active tuberculosis.'
```

In [ ]:

```
print('Here total number of unique_IDS are:', len(IMAGES.keys()))# check unique person id.
```

```
Here total number of unique_IDS are: 3867
```

Note: Here one information is realized that there are 3344 unique images which are of different peoples out of 7470 images.

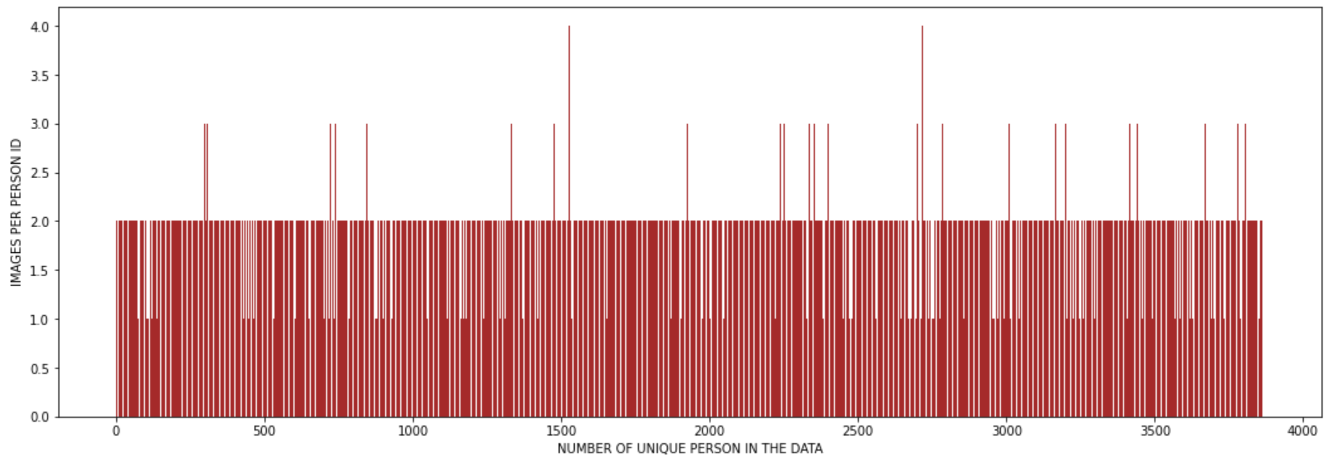
Let's check with the help of plot how many unique person ID has the multiple images on the same id.

In [ ]:

```
plt.figure(figsize=(18,6))# intialize the figure size.
plt.bar(range(len(IMAGES.keys())), IMAGES.values(),color='brown')# here we plot the bar graph with
the images keys.
plt.ylabel('IMAGES PER PERSON ID')
plt.xlabel('NUMBER OF UNIQUE PERSON IN THE DATA')
```

Out[ ]:

```
Text(0.5, 0, 'NUMBER OF UNIQUE PERSON IN THE DATA')
```



Observation:

1. From the above plot we can observe that there are less no of IDS which have four images corresponding to the ids.
2. Maximum ids have the two chest x-ray images corresponding to the person ids.

Let's check the count of the numbers of ids which have multiple chest x-ray images on them.

In [ ]:

```
One_images_corr_ids= 0 # intialize the one images per person id
Two_images_corr_ids= 0  # intialize the two images per person id
Three_images_corr_ids= 0 # intialize the three images per person id
```

```
Four_images_corr_ids= 0  # intialize the four images per person id

for count in IMAGES.values(): # for each values in IMAGES
  if count == 1:  # first check the count of the value =1
    One_images_corr_ids+=1# increase the count for the only one person_id.
  elif count == 2:# first check the count of the value =2
    Two_images_corr_ids+=1# increase the count for the only two person_id.
  elif count == 3:# first check the count of the value =3
    Three_images_corr_ids+=1# increase the count for the only three person_id.
  elif count == 4:# first check the count of the value =4
    Four_images_corr_ids+=1# increase the count for the only Four person_id.
  else:
    print('ERROR')
```

In [ ]:

```
print('count of those IDS which have only one image:',One_images_corr_ids)
```

```
count of those IDS which have only one image: 457
```

In [ ]:

```
print('count of those IDS which have two image:',Two_images_corr_ids)
```

```
count of those IDS which have two image: 3227
```

In [ ]:

```
print('count of those IDS which have three image:',Three_images_corr_ids)
```

```
count of those IDS which have three image: 173
```

In [ ]:

```
print('count of those IDS which have fourth image:',Four_images_corr_ids)
```

```
count of those IDS which have fourth image: 10
```

Observation:

1. Count of the Ids which have the two images is 3227 which is maximum in comparison to 457 ids which have only one image and 173 ids have 3 multiple images and minimum four images corresponding to 10 person ids.
2. With the help of this analysis we can conclude that there are multiple images corresponding to a single person. These are different chest scans at different views.

As we able to see that there are 3 and 4 images exist corresponding one xml files somewhere its create a problem for getting only frontal images as only limiting the images to two images in our data so we try to extract only frontal and lateral image if we have more than 2 images for a corresponding xml files.

we use this data to map that frontal and lateral image with our data frame images

In [17]:

```
# https://www.kaggle.com/raddar/chest-xrays-indiana-university
data_projecttions = pd.read_csv("/content/indiana_projections.csv")
```

for extracting frontal and lateral image

In [18]:

```
def find_Fr_la(li):
    list_of_images = [] # here we get the list of images
    image_last = "" # last image
    for i in li:# for each image in list of image
```

```
        projection = data_projecttions[data_projecttions['filename'].str.contains(re.search(r"\d.*\
_IM-\d.*\.", i).group())]['projection'].values# get the corresponding projection values to the dat
aframe.
        if "Lateral" == projection:# check projection is lateral
            image_last = i# make it last image
        else:
            list_of_images.append(i)# otherwise make them frontal images
    return list_of_images, image_last
```

data_frame having frontal and lateral image corresponding to findings and impression.

In [20]:

```
columns = ["image_1", "image_2", "impression","findings"]
df = pd.DataFrame(columns = columns)
no_lateral = 0
for item in tqdm(data_frame.iterrows()):
    lnegth_of_image = item[1]['Path'].split(',')# getting all the images corresponding to each path
    if len(lnegth_of_image) > 2:# check images are more than two in each xml file
        list_of_images, image_last = find_Fr_la(lnegth_of_image)# get the frontal and lateral image
        if image_last == "":# check if last image is none.
            no_lateral +=1# then increase the no lateral by one
            list_of_images, image_last = list_of_images[:-1], list_of_images[-1]# and get the last 
image an list of images
        for i in list_of_images:# for each image in list of images
            image_1 = i# get the frontal image as first image
            image_2 = image_last# and get the lateral image as last image
            df = df.append(pd.Series([image_1, image_2, item[1]['impression'],item[1]['findings']],
index = columns), ignore_index = True)# make the dataframe.
    elif len(lnegth_of_image) == 2:# check for only two images
        image_1 = lnegth_of_image[0] # get the first image
        image_2 = lnegth_of_image[1]# get the second image
        df = df.append(pd.Series([image_1, image_2, item[1]['impression'],item[1]['findings']], ind
ex = columns), ignore_index = True)# cmake the dataframe
    elif len(lnegth_of_image) == 1:# check if we have only one image, here we copy the only one 
image second time.
        df = df.append(pd.Series([lnegth_of_image[0],lnegth_of_image[0], item[1]['impression'],item
[1]['findings']], index = columns), ignore_index = True)#
print("Total Report without Lateral images {}".format(no_lateral))
```

```
3851it [00:12, 317.25it/s]
```

```
Total Report without Lateral images 1
```

In [21]:

```
df
```

Out[21]:

	image_1	image_2	impression	findings
0	CXR3679_IM-1831-1001.png	CXR3679_IM-1831-2001.png	Normal chest exam.	Normal heart. Clear lungs. No pneumothorax. No...
1	CXR3681_IM-1833-0001-0002.png	CXR3681_IM-1833-0001-0001.png	No acute abnormality identified.	There are lower lung volumes. There is central...
2	CXR368_IM-1832-1001.png	CXR368_IM-1832-2001.png	No acute cardiopulmonary findings.	Cardiomediastinal silhouette and pulmonary vas...
3	CXR3665_IM-1823-1001.png	CXR3665_IM-1823-2001.png	1. No acute abnormality. 2. No evidence of pul...	Heart size within normal limits. Negative for ...
4	CXR3700_IM-1848-1001.png	CXR3700_IM-1848-1001.png	1. Cardiomegaly with mild vascular prominence....	Cardiomegaly is present. This is unchanged. Th...
...	...	...	...	...
3973	CXR187_IM-0563-2001.png	CXR187_IM-0563-1001.png	Unchanged exam without acute abnormality.	Normal heart size. Stable tortuous aorta. No p...
3974	CXR1887_IM-0575-1001.png	CXR1887_IM-0575-1001.png	Senescent changes no acute pulmonary disease.	The lungs are clear. There is no pleural effus...
[illegible]	[illegible]	[illegible]	[illegible]	The heart is normal in size. The

	image_1	image_2	impression	findings
3975	CXR185_IM-0551-2001.png	CXR185_IM-0551-1001.png	No acute disease.	mediastinum i
3976	CXR1851_IM-0553-1001.png	CXR1851_IM-0553-2001.png	No acute cardiopulmonary findings. .	Heart size is normal. No focal airspace consol...
3977	CXR1863_IM-0558-1001.png	CXR1863_IM-0558-3001.png	1. Low volume study without acute process. 2. ...	Heart size is mildly enlarged. Tortuous aorta....

3978 rows × 4 columns

EDA ON TEXT DATA

Preprocessing of findings and impression

1. Decontractions of words like won't to will not.

In [22]:

```
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)


    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

Lowercase all the letters

In [23]:

```
def lowercase(text):
  text= text.lower()
  return text
```

Remove the punctuations from the text

In [24]:

```
def punctuations_remove(text):
  punct='''!()-[]{};:'"\,<>/?@#$%^&*~'''
  for char in text:
    if char in punct:
      text= text.replace(char,"")
    text= " ".join(k for k in text.split())
  return text
```

Remove the numbers from the text

In [25]:

```
def numbers(text):
  tem= re.sub(r'x*','',text)
  text= re.sub(r'\d','',tem)
  return text
```

Remove the words who has length then two except no and ct.

In [26]:

```
def filters(text):
  tem= text.split()
  tem2= []
  for word in tem:
    if len(word) <=2 and word !='no' and word!='ct':
      continue
    else:
      tem2.append(word)
    text=' '.join(k for k in tem2)
  return text
```

Remove the fullstops

In [27]:

```
def fullstops(text):
  text=re.sub(r'\.\.+','.',text)
  return text
```

In [28]:

```
def stops(text):
    text= re.sub('\.',' .', text)
    return text
```

Remove the spaces

In [29]:

```
def spaces(text):
  text= ' '.join(k for k in text.split())
  return text
```

In [30]:

```
def words(text):
  temp=[]
  words= text.split()
  for i in words:
    if i.startswith('.') == False:
      temp.append(i)
    else:
      k = i.replace('.','. ')
      temp.append(k)
    text= ' '.join(k for k in temp)
  return text
```

Remove the apostrophe from the text.

In [31]:

```
def remaining_apostrophe(text):
  text= re.sub("'",'',text)
  return text
```

In [32]:

```
import re
def preprocess(text):
  new_text = re.sub('XXXX','',str(text))#substituting XXXX string which occurs n most of the
impressions with space as it has no semantic meaning
  new_text=re.sub(r'[^A-Za-z]+',' ',new_text)#replacing anything other than words wtih space
  new_text = decontracted(new_text)#decontracting the words
  new_text=   lowercase(new_text)
  new_text= punctuations_remove(new_text)
  new_text= numbers(new_text)
  new_text= filters(new_text)
  new_text= fullstops(new_text)
  new_text= stops(new_text)
  new_text= spaces(new_text)
```

```
    new_text= words(new_text)
    new_text= remaining_apostrophe(new_text)

    # text = re.sub('\\r', ' ',text)#replacing new line with space
    # text = re.sub('\\n', ' ',text)#replacing tab with single space

    return new_text
```

Text Preprocessing

Preprocess the impression data

In [33]:

```
df['impression']=df['impression'].map(preprocess)
```

Preporcess the findings data

In [34]:

```
df['findings']=df['findings'].map(preprocess)
```

In [35]:

```
df
```

Out[35]:

	image_1	image_2	impression	findings
0	CXR3679_IM-1831-1001.png	CXR3679_IM-1831-2001.png	normal chest eam	normal heart clear lungs no pneumothora no ple...
1	CXR3681_IM-1833-0001-0002.png	CXR3681_IM-1833-0001-0001.png	no acute abnormality identified	there are lower lung volumes there central bro...
2	CXR368_IM-1832-1001.png	CXR368_IM-1832-2001.png	no acute cardiopulmonary findings	cardiomediastinal silhouette and pulmonary vas...
3	CXR3665_IM-1823-1001.png	CXR3665_IM-1823-2001.png	no acute abnormality no evidence pulmonary tub...	heart size within normal limits negative for f...
4	CXR3700_IM-1848-1001.png	CXR3700_IM-1848-1001.png	cardiomegaly with mild vascular prominence no ...	cardiomegaly present this unchanged there mild...
...	...	...	...	...
3973	CXR187_IM-0563-2001.png	CXR187_IM-0563-1001.png	unchanged eam without acute abnormality	normal heart size stable tortuous aorta no pne...
3974	CXR1887_IM-0575-1001.png	CXR1887_IM-0575-1001.png	senescent changes no acute pulmonary disease	the lungs are clear there no pleural effusion ...
3975	CXR185_IM-0551-2001.png	CXR185_IM-0551-1001.png	no acute disease	the heart normal size the mediastinum stable l...
3976	CXR1851_IM-0553-1001.png	CXR1851_IM-0553-2001.png	no acute cardiopulmonary findings	heart size normal no focal airspace consolidat...
3977	CXR1863_IM-0558-1001.png	CXR1863_IM-0558-3001.png	low volume study without acute process mild ca...	heart size mildly enlarged tortuous aorta lung...

3978 rows × 4 columns

Let's get the number of words in each findings

In [36]:

```
length_word_findings=[len(k.split()) for k in df['findings'].values]
```

Let's get the number of words in each impression

In [37]:

```
length_word_impression=[len(k.split()) for k in df['impression'].values]
```

In [38]:

```
print('maximum length of the sentence in findings:',max(length_word_findings))
print('maximum length of the sentence in impression:',max(length_word_impression))
```

```
maximum length of the sentence in findings: 138
maximum length of the sentence in impression: 104
```

In [39]:

```
range((df.shape[0]),1)
```

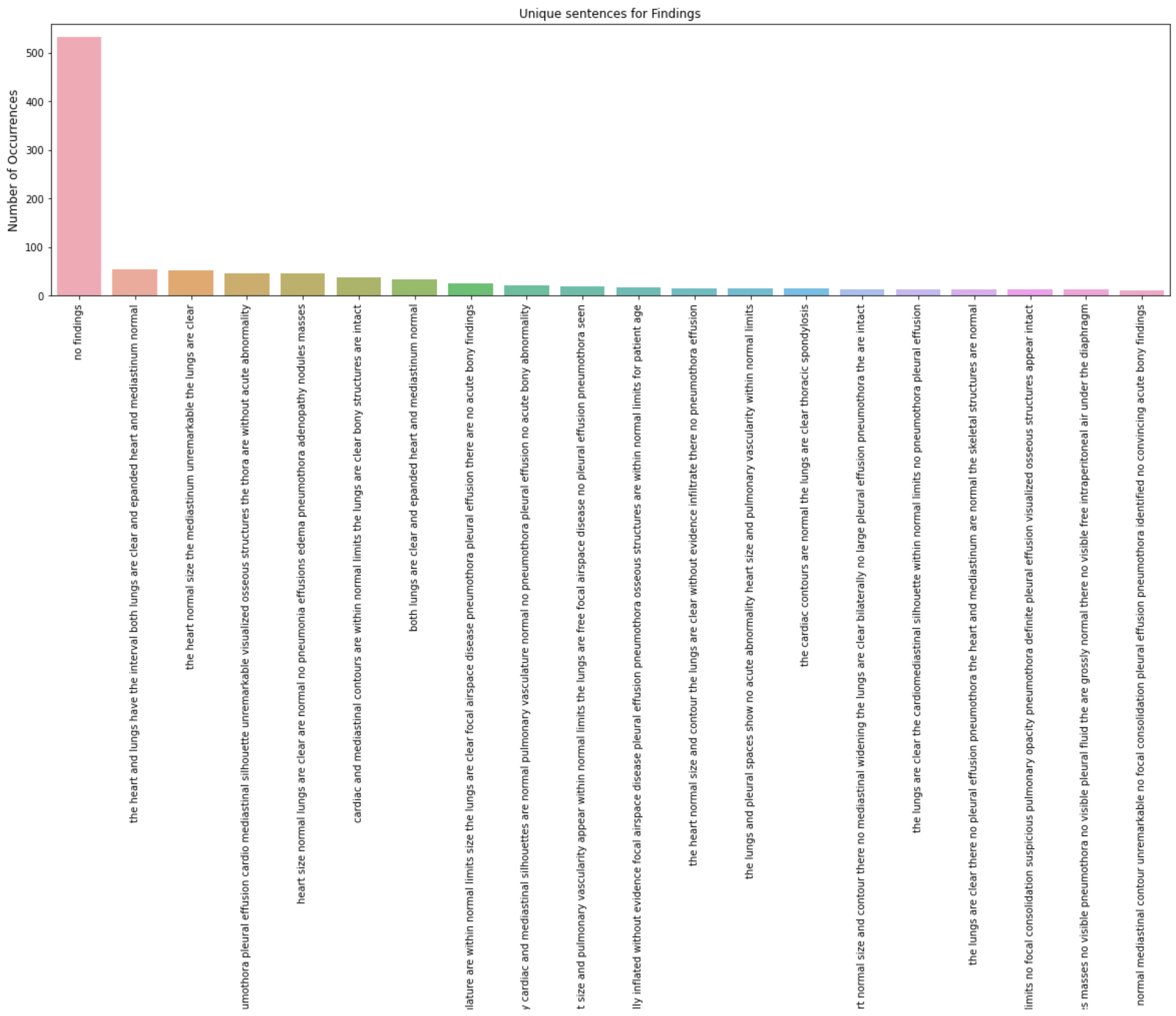
Out[39]:

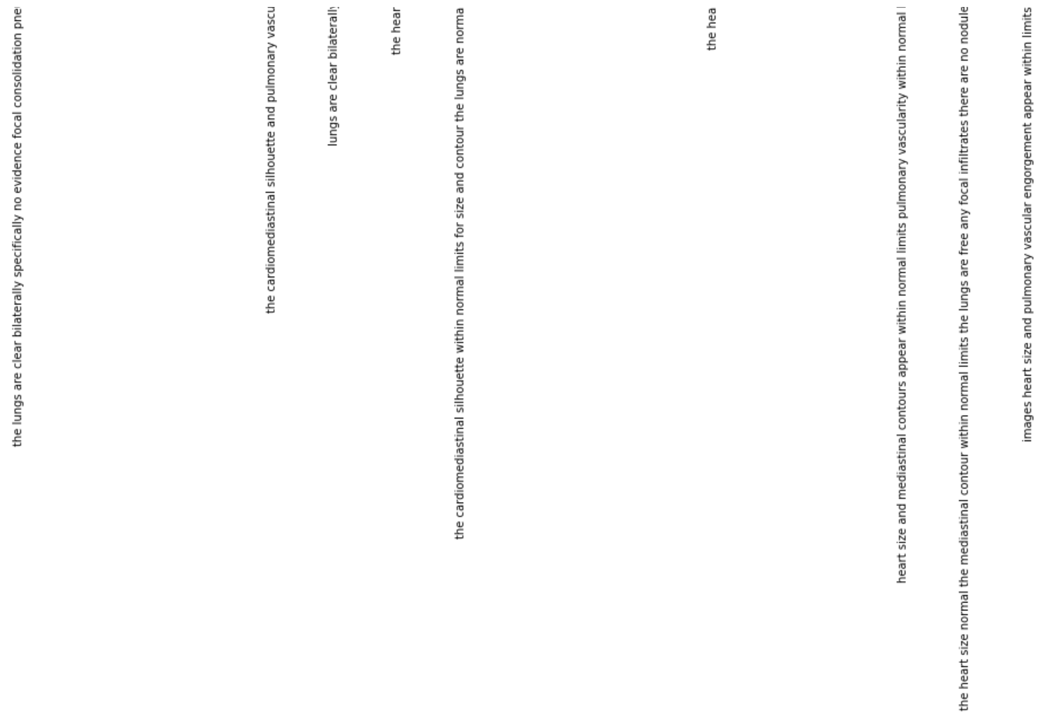
```
range(3978, 1)
```

Top 20 most occuring findings

In [41]:

```
findings = df.findings.value_counts()[:20]
plt.figure(figsize=(20,5))
sns.barplot(findings.index, findings.values, alpha=0.8)
plt.title("Unique sentences for Findings")
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xticks(rotation=90)
plt.show()
```

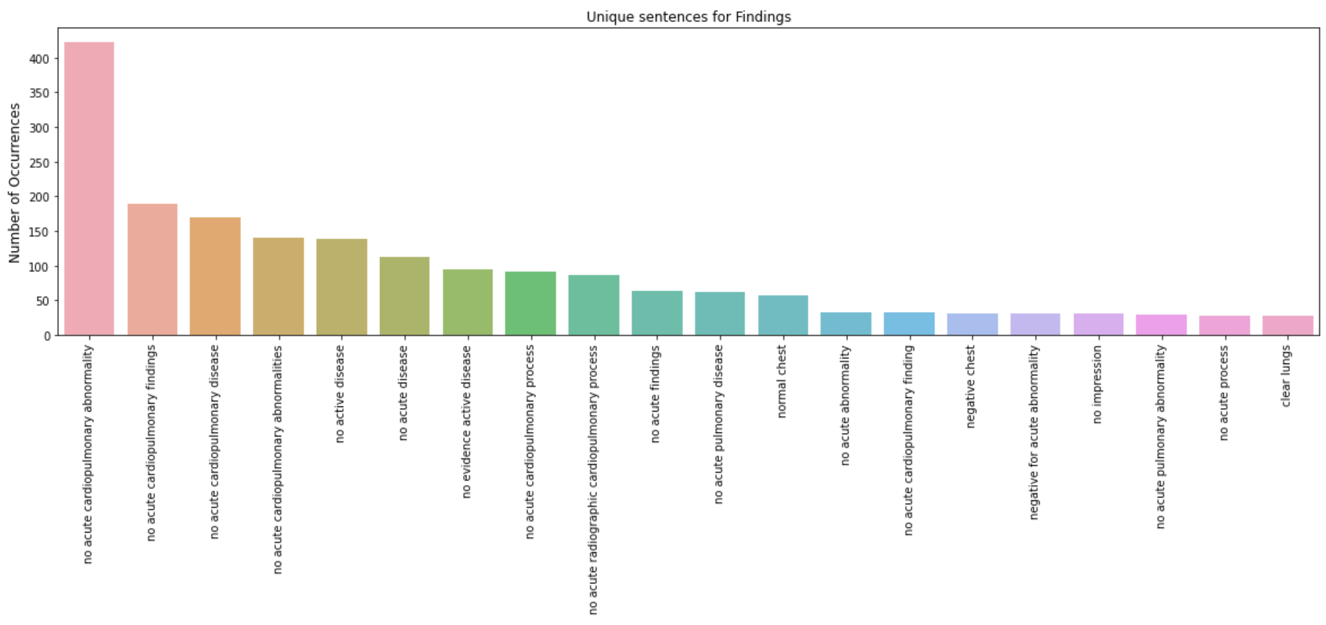




top 20 most occuring impression

In [42]:

```
impression = df.impression.value_counts()[:20]
plt.figure(figsize=(20,5))
sns.barplot(impression.index, impression.values, alpha=0.8)
plt.title("Unique sentences for Findings")
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xticks(rotation=90)
plt.show()
```



Observation:

1. First thing is that we have maximum number of sentences is no findings it means we have the maximum number of images whose findings are not available in the reports.
2. On the other hand if we look up in the impression sentences count we foound out that most of the reports have the no acute cardiopulmonary abnormality.It tells us about that it is the common impression which we found in the indiana chest x ray data.
3. Mostly longer sentences are occurred less than are equal to 10 times.

Plot the number of words in a findings

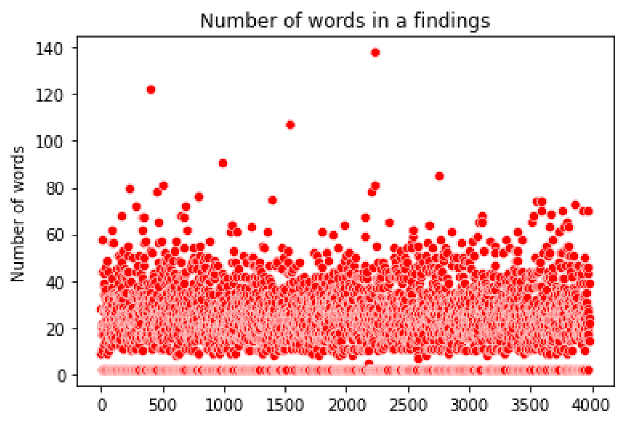
Plot the number of words in a findings

In [43]:

```
plt.title('Number of words in a findings')
sns.scatterplot(range(df.shape[0]),length_word_findings, color='red')
plt.ylabel('Number of words')
```

Out[43]:

```
Text(0, 0.5, 'Number of words')
```



Observation:

1. Above figure shows us that most of the word count is below 100.
2. Most of the points are dense lie in the range below 50 . It means most of the length of word count is below 50.
3. here the word range is more in comparison to the impression so we expect the larger sentences.

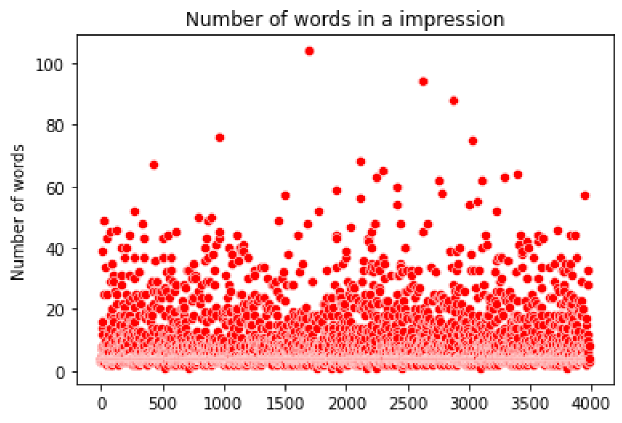
Plot the number of words in each impression

In [44]:

```
plt.title('Number of words in a impression')
sns.scatterplot(range(df.shape[0]),length_word_impression, color='red')
plt.ylabel('Number of words')
```

Out[44]:

```
Text(0, 0.5, 'Number of words')
```



Observation:

1. Above figure shows us that most of the word count is below 20.
2. Most of the points are dense lie in the range below 10 . It means most of the length of word count is below 10.
3. As the weords range is less it means we expect the shorter sentence in the impression.

for findings

In [46]:

```
length_findings= []
for i in df['findings'].values:
  length_findings.extend(i.split())
```

description of words in findings

In [49]:

```
word_length_findings = df.findings.str.split().apply(lambda x:len(x))
```

In [50]:

```
print('mean of words are in single impression are :',word_length_findings.mean())
print('median of words are in single impression are:',word_length_findings.median())
print('maximum number of words are in single impression are :',word_length_findings.max())
print('minimum number of words are in single impression are :',word_length_findings.min())
```

```
mean of words are in single impression are : 23.823529411764707
median of words are in single impression are: 23.0
maximum number of words are in single impression are : 138
minimum number of words are in single impression are : 2
```

for impression

In [ ]:

```
length_impression= []
for i in df['impression'].values:
  length_impression.extend(i.split())
```

description of words in impression

In [51]:

```
word_length_impression = df.impression.str.split().apply(lambda x:len(x))
```

In [53]:

```
print('mean of words are in single impression are :',word_length_impression.mean())
print('median of words are in single impression are:',word_length_impression.median())
print('maximum number of words are in single impression are :',word_length_impression.max())
print('minimum number of words are in single impression are :',word_length_impression.min())
```

```
mean of words are in single impression are : 9.024886877828054
median of words are in single impression are: 4.0
maximum number of words are in single impression are : 104
minimum number of words are in single impression are : 1
```

In [ ]:

```
counter_findings= Counter(length_findings)
```

In [ ]:

```
counter_impression= Counter(length_impression)
```

In [ ]:

```
words_finding= []
count_finding= []
for key,value in counter_findings.items():
  words_finding.append(key)
  words_finding.append(key)
  count_finding.append(value)
count_words_findings = list(zip(count_finding,words_finding))
```

In [ ]:

```
words_impression= []
count_impression= []
for key,value in counter_impression.items():
  words_impression.append(key)
  count_impression.append(value)
count_words_impression = list(zip(count_finding,words_finding))
```

In [ ]:

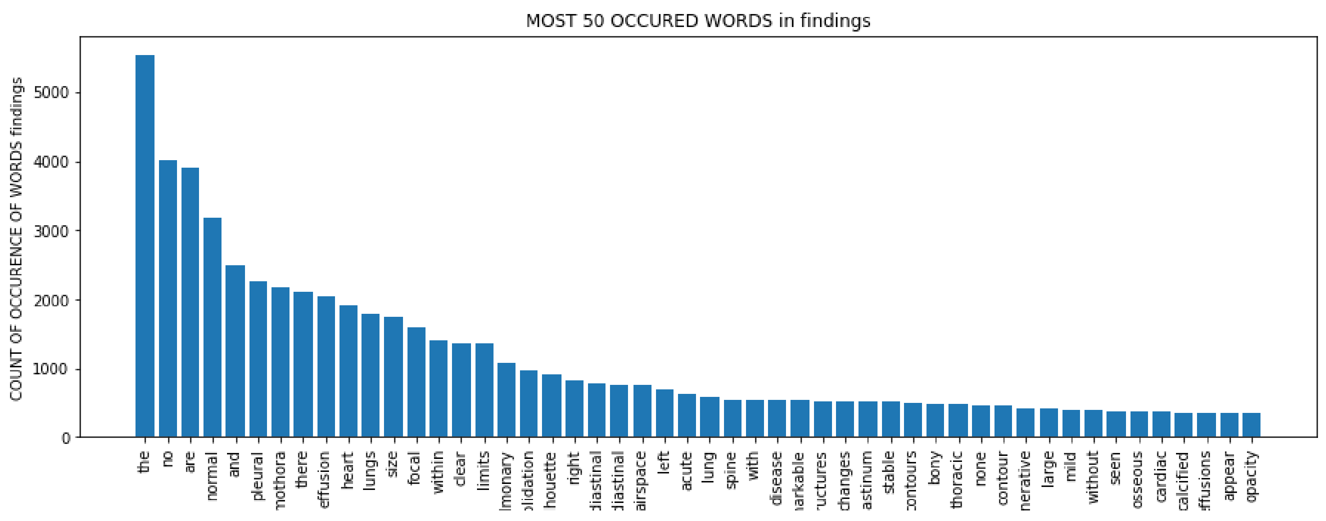
```
top_words_50_findings= sorted(count_words_findings)[::-1][:50]
top_words_50_impression= sorted(count_words_impression)[::-1][:50]
low_words_50_findings= sorted(count_words_findings)[:50]
low_words_50_impression= sorted(count_words_impression)[:50]
```

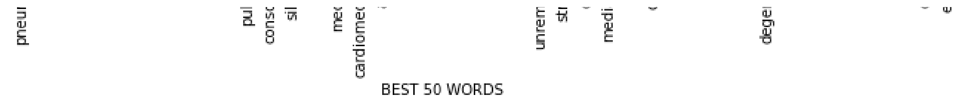
In [ ]:

```
plt.figure(figsize=(15,5))
plt.bar(range(50),[i for i,w in top_words_50_findings])
plt.title('MOST 50 OCCURED WORDS in findings')
plt.xlabel(' BEST 50 WORDS')
plt.ylabel('COUNT OF OCCURENCE OF WORDS findings')
plt.xticks(ticks=range(50), labels=[word for i,word in top_words_50_findings],rotation=90)
```

Out[ ]:

```
([<matplotlib.axis.XTick at 0x7fe2f15d47b8>,
  <matplotlib.axis.XTick at 0x7fe2f15d4780>,
  <matplotlib.axis.XTick at 0x7fe2f15d44a8>,
  <matplotlib.axis.XTick at 0x7fe2f15bc780>,
  <matplotlib.axis.XTick at 0x7fe2f15bcc18>,
  <matplotlib.axis.XTick at 0x7fe2f15c2160>,
  <matplotlib.axis.XTick at 0x7fe2f15c2588>,
  <matplotlib.axis.XTick at 0x7fe2f15c2a20>,
  <matplotlib.axis.XTick at 0x7fe2f15c2eb8>,
  <matplotlib.axis.XTick at 0x7fe2f15c2b00>,
  <matplotlib.axis.XTick at 0x7fe2f15bcd68>,
  <matplotlib.axis.XTick at 0x7fe2f15c4128>,
  <matplotlib.axis.XTick at 0x7fe2f15c4978>,
  <matplotlib.axis.XTick at 0x7fe2f15c4e10>,
  <matplotlib.axis.XTick at 0x7fe2f15c82e8>,
  <matplotlib.axis.XTick at 0x7fe2f15c8780>,
  <matplotlib.axis.XTick at 0x7fe2f15c8c18>,
  <matplotlib.axis.XTick at 0x7fe2f15ca160>,
  <matplotlib.axis.XTick at 0x7fe2f15ca588>,
  <matplotlib.axis.XTick at 0x7fe2f15c8860>,
  <matplotlib.axis.XTick at 0x7fe2f15c20b8>,
  <matplotlib.axis.XTick at 0x7fe2f15c4470>,
  <matplotlib.axis.XTick at 0x7fe2f15ca630>,
  <matplotlib.axis.XTick at 0x7fe2f15caf60>,
  <matplotlib.axis.XTick at 0x7fe2f15cf438>,
  <matplotlib.axis.XTick at 0x7fe2f15cf8d0>,
  <matplotlib.axis.XTick at 0x7fe2f15cfd68>,
  <matplotlib.axis.XTick at 0x7fe2f1553240>,
  <matplotlib.axis.XTick at 0x7fe2f15536d8>,
  <matplotlib.axis.XTick at 0x7fe2f15cf898>,
  <matplotlib.axis.XTick at 0x7fe2f15ca550>,
  <matplotlib.axis.XTick at 0x7fe2f1553a20>,
  <matplotlib.axis.XTick at 0x7fe2f1553d68>,
  <matplotlib.axis.XTick at 0x7fe2f1558240>,
  <matplotlib.axis.XTick at 0x7fe2f15586d8>,
  <matplotlib.axis.XTick at 0x7fe2f1558b70>,
  <matplotlib.axis.XTick at 0x7fe2f15600b8>,
  <matplotlib.axis.XTick at 0x7fe2f15604e0>,
  <matplotlib.axis.XTick at 0x7fe2f1560978>,
  <matplotlib.axis.XTick at 0x7fe2f15587b8>,
  <matplotlib.axis.XTick at 0x7fe2f1553cf8>,
  <matplotlib.axis.XTick at 0x7fe2f1560ba8>,
  <matplotlib.axis.XTick at 0x7fe2f156a0b8>,
 <matplotlib.axis.XTick at 0x7fe2f156a4e0>,
 <matplotlib.axis.XTick at 0x7fe2f156a978>,
 <matplotlib.axis.XTick at 0x7fe2f156ae10>,
 <matplotlib.axis.XTick at 0x7fe2f15712e8>,
 <matplotlib.axis.XTick at 0x7fe2f1571780>,
 <matplotlib.axis.XTick at 0x7fe2f1571c18>,
 <matplotlib.axis.XTick at 0x7fe2f1571cf8>],
[Text(0, 0, 'the'),
 Text(0, 0, 'no'),
 Text(0, 0, 'are'),
 Text(0, 0, 'normal'),
 Text(0, 0, 'and'),
 Text(0, 0, 'pleural'),
 Text(0, 0, 'pneumothora'),
 Text(0, 0, 'there'),
 Text(0, 0, 'effusion'),
 Text(0, 0, 'heart'),
 Text(0, 0, 'lungs'),
 Text(0, 0, 'size'),
 Text(0, 0, 'focal'),
 Text(0, 0, 'within'),
 Text(0, 0, 'clear'),
 Text(0, 0, 'limits'),
 Text(0, 0, 'pulmonary'),
 Text(0, 0, 'consolidation'),
 Text(0, 0, 'silhouette'),
 Text(0, 0, 'right'),
 Text(0, 0, 'mediastinal'),
 Text(0, 0, 'cardiomediastinal'),
 Text(0, 0, 'airspace'),
 Text(0, 0, 'left'),
 Text(0, 0, 'acute'),
 Text(0, 0, 'lung'),
 Text(0, 0, 'spine'),
 Text(0, 0, 'with'),
 Text(0, 0, 'disease'),
 Text(0, 0, 'unremarkable'),
 Text(0, 0, 'structures'),
 Text(0, 0, 'changes'),
 Text(0, 0, 'mediastinum'),
 Text(0, 0, 'stable'),
 Text(0, 0, 'contours'),
 Text(0, 0, 'bony'),
 Text(0, 0, 'thoracic'),
 Text(0, 0, 'none'),
 Text(0, 0, 'contour'),
 Text(0, 0, 'degenerative'),
 Text(0, 0, 'large'),
 Text(0, 0, 'mild'),
 Text(0, 0, 'without'),
 Text(0, 0, 'seen'),
 Text(0, 0, 'osseous'),
 Text(0, 0, 'cardiac'),
 Text(0, 0, 'calcified'),
 Text(0, 0, 'effusions'),
 Text(0, 0, 'appear'),
 Text(0, 0, 'opacity')])
```





Observation:

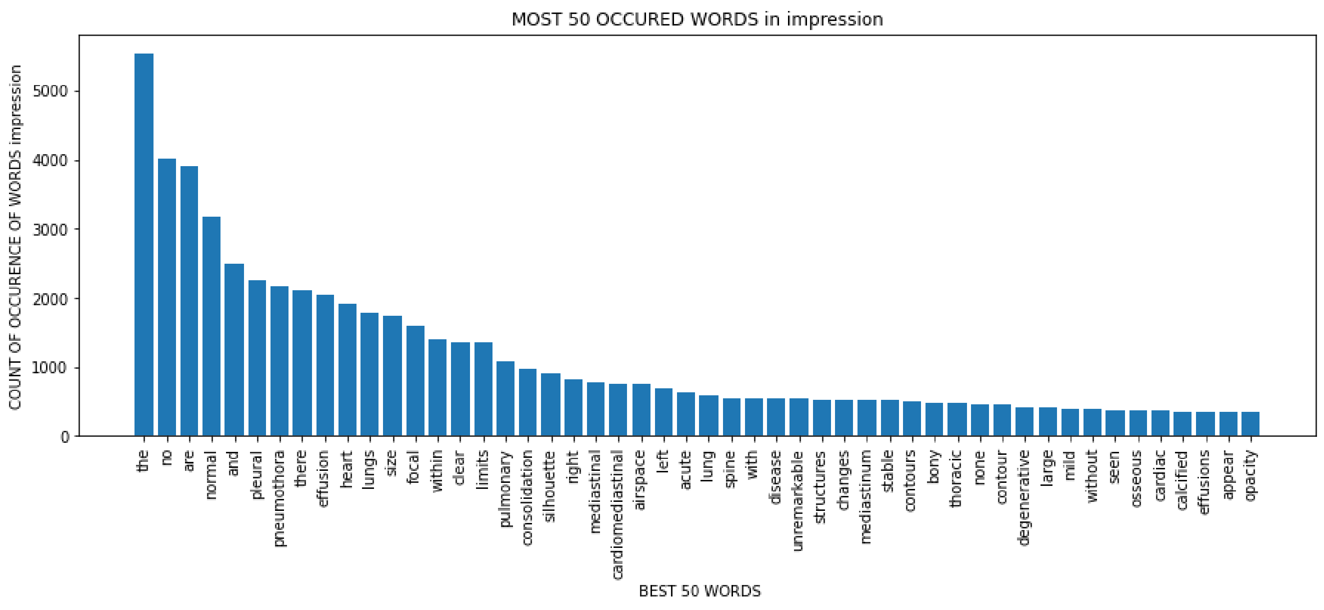
1. Here 'no' is the most occuiring words in findings it means somewhere most of the reports are neutral.
2. After this pneumothora is the most occured word it means this is the common disease which is found in having lungs diseases carrying patients.

In [ ]:

```
plt.figure(figsize=(15,5))
plt.bar(range(50),[i for i,w in top_words_50_impression])
plt.title('MOST 50 OCCURED WORDS in impression')
plt.xlabel(' BEST 50 WORDS')
plt.ylabel('COUNT OF OCCURENCE OF WORDS impression')
plt.xticks(ticks=range(50), labels=[word for i,word in top_words_50_impression],rotation=90)
```

Out[ ]:

```
([<matplotlib.axis.XTick at 0x7fe2f153b940>,
  <matplotlib.axis.XTick at 0x7fe2f1531b38>,
  <matplotlib.axis.XTick at 0x7fe2f1531d30>,
  <matplotlib.axis.XTick at 0x7fe2f14786a0>,
  <matplotlib.axis.XTick at 0x7fe2f1478b38>,
  <matplotlib.axis.XTick at 0x7fe2f1478d68>,
  <matplotlib.axis.XTick at 0x7fe2f1478828>,
  <matplotlib.axis.XTick at 0x7fe2f14802b0>,
  <matplotlib.axis.XTick at 0x7fe2f14809e8>,
  <matplotlib.axis.XTick at 0x7fe2f1480e80>,
  <matplotlib.axis.XTick at 0x7fe2f1489358>,
  <matplotlib.axis.XTick at 0x7fe2f14897f0>,
  <matplotlib.axis.XTick at 0x7fe2f1489c88>,
  <matplotlib.axis.XTick at 0x7fe2f148f198>,
  <matplotlib.axis.XTick at 0x7fe2f148f5f8>,
  <matplotlib.axis.XTick at 0x7fe2f148fa90>,
  <matplotlib.axis.XTick at 0x7fe2f1489f98>,
  <matplotlib.axis.XTick at 0x7fe2f1480940>,
  <matplotlib.axis.XTick at 0x7fe2f148f748>,
  <matplotlib.axis.XTick at 0x7fe2f141a198>,
  <matplotlib.axis.XTick at 0x7fe2f141a5f8>,
  <matplotlib.axis.XTick at 0x7fe2f141aa90>,
  <matplotlib.axis.XTick at 0x7fe2f141af28>,
  <matplotlib.axis.XTick at 0x7fe2f1422400>,
  <matplotlib.axis.XTick at 0x7fe2f1422898>,
  <matplotlib.axis.XTick at 0x7fe2f1422d30>,
  <matplotlib.axis.XTick at 0x7fe2f1422940>,
  <matplotlib.axis.XTick at 0x7fe2f141aa58>,
  <matplotlib.axis.XTick at 0x7fe2f14784e0>,
  <matplotlib.axis.XTick at 0x7fe2f142c390>,
  <matplotlib.axis.XTick at 0x7fe2f142c828>,
  <matplotlib.axis.XTick at 0x7fe2f142ccc0>,
  <matplotlib.axis.XTick at 0x7fe2f1434198>,
  <matplotlib.axis.XTick at 0x7fe2f1434630>,
  <matplotlib.axis.XTick at 0x7fe2f1434ac8>,
  <matplotlib.axis.XTick at 0x7fe2f1434f60>,
  <matplotlib.axis.XTick at 0x7fe2f1434278>,
  <matplotlib.axis.XTick at 0x7fe2f142c128>,
  <matplotlib.axis.XTick at 0x7fe2f143d0f0>,
  <matplotlib.axis.XTick at 0x7fe2f143d6a0>,
  <matplotlib.axis.XTick at 0x7fe2f143db38>,
  <matplotlib.axis.XTick at 0x7fe2f143dd68>,
  <matplotlib.axis.XTick at 0x7fe2f14464a8>,
  <matplotlib.axis.XTick at 0x7fe2f1446940>,
  <matplotlib.axis.XTick at 0x7fe2f1446dd8>,
  <matplotlib.axis.XTick at 0x7fe2f1446cc0>,
  <matplotlib.axis.XTick at 0x7fe2f143dc18>,
  <matplotlib.axis.XTick at 0x7fe2f1434240>,
  <matplotlib.axis.XTick at 0x7fe2f1480ac8>,
  <matplotlib.axis.XTick at 0x7fe2f144e550>],
 [Text(0, 0, 'the'),
  Text(0, 0, 'no'),
  Text(0, 0, 'are'),
 Text(0, 0, 'are'),
 Text(0, 0, 'normal'),
 Text(0, 0, 'and'),
 Text(0, 0, 'pleural'),
 Text(0, 0, 'pneumothora'),
 Text(0, 0, 'there'),
 Text(0, 0, 'effusion'),
 Text(0, 0, 'heart'),
 Text(0, 0, 'lungs'),
 Text(0, 0, 'size'),
 Text(0, 0, 'focal'),
 Text(0, 0, 'within'),
 Text(0, 0, 'clear'),
 Text(0, 0, 'limits'),
 Text(0, 0, 'pulmonary'),
 Text(0, 0, 'consolidation'),
 Text(0, 0, 'silhouette'),
 Text(0, 0, 'right'),
 Text(0, 0, 'mediastinal'),
 Text(0, 0, 'cardiomediastinal'),
 Text(0, 0, 'airspace'),
 Text(0, 0, 'left'),
 Text(0, 0, 'acute'),
 Text(0, 0, 'lung'),
 Text(0, 0, 'spine'),
 Text(0, 0, 'with'),
 Text(0, 0, 'disease'),
 Text(0, 0, 'unremarkable'),
 Text(0, 0, 'structures'),
 Text(0, 0, 'changes'),
 Text(0, 0, 'mediastinum'),
 Text(0, 0, 'stable'),
 Text(0, 0, 'contours'),
 Text(0, 0, 'bony'),
 Text(0, 0, 'thoracic'),
 Text(0, 0, 'none'),
 Text(0, 0, 'contour'),
 Text(0, 0, 'degenerative'),
 Text(0, 0, 'large'),
 Text(0, 0, 'mild'),
 Text(0, 0, 'without'),
 Text(0, 0, 'seen'),
 Text(0, 0, 'osseous'),
 Text(0, 0, 'cardiac'),
 Text(0, 0, 'calcified'),
 Text(0, 0, 'effusions'),
 Text(0, 0, 'appear'),
 Text(0, 0, 'opacity')])
```



observation:

1. Here important thing is that the distribution of words in impression and findings are mostly same in geeting of top 50 words in

findings and impression.

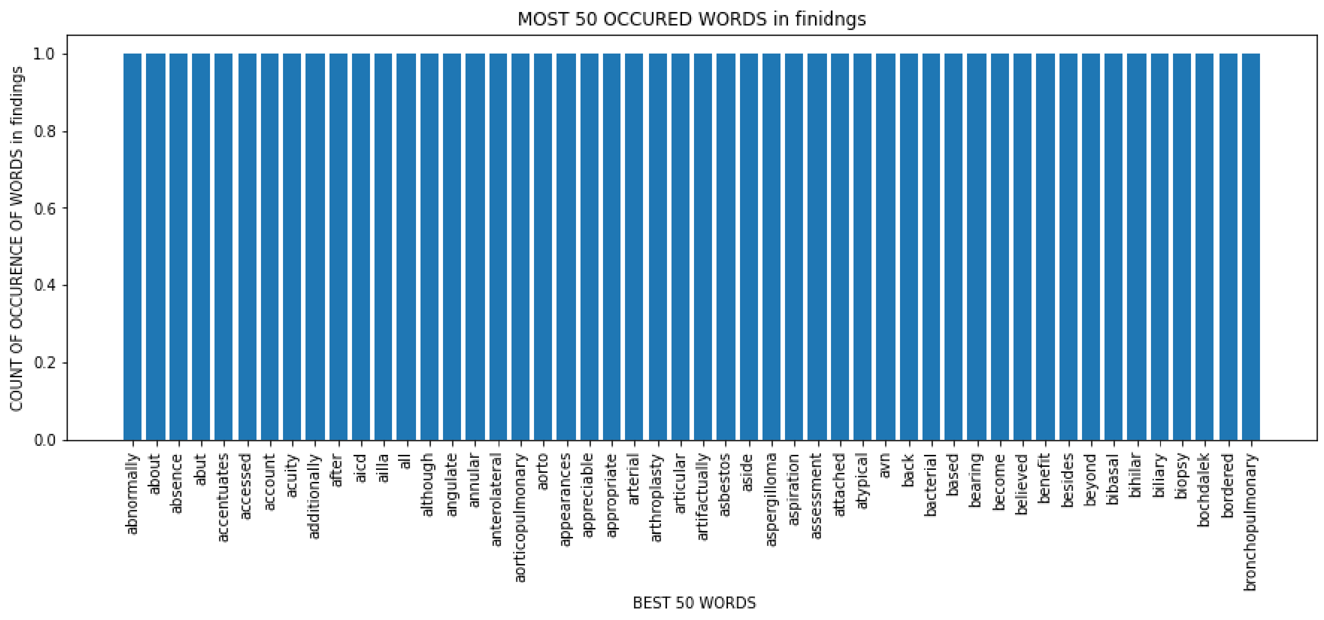
2. it means the words which we get in the findings and impression gives us the insight that in indiana chest x ray most of the patient have no disease and after that its common to have a common disease as pneumothora.

In [ ]:

```
plt.figure(figsize=(15,5))
plt.bar(range(50),[i for i,w in low_words_50_findings])
plt.title('MOST 50 OCCURED WORDS in finidngs')
plt.xlabel(' BEST 50 WORDS')
plt.ylabel('COUNT OF OCCURENCE OF WORDS in findings')
plt.xticks(ticks=range(50), labels=[word for i,word in low_words_50_findings],rotation=90)
```

Out[ ]:

```
([<matplotlib.axis.XTick at 0x7fe2f6acd8d0>,
  <matplotlib.axis.XTick at 0x7fe2f6acd588>,
  <matplotlib.axis.XTick at 0x7fe2f6aff828>,
  <matplotlib.axis.XTick at 0x7fe2f540d8d0>,
  <matplotlib.axis.XTick at 0x7fe2f539b1d0>,
  <matplotlib.axis.XTick at 0x7fe2f76ed978>,
  <matplotlib.axis.XTick at 0x7fe2f76ee3c8>,
  <matplotlib.axis.XTick at 0x7fe2f5081128>,
  <matplotlib.axis.XTick at 0x7fe2f5081710>,
  <matplotlib.axis.XTick at 0x7fe2f5081dd8>,
  <matplotlib.axis.XTick at 0x7fe2f7640748>,
  <matplotlib.axis.XTick at 0x7fe2f884a240>,
  <matplotlib.axis.XTick at 0x7fe2f1326198>,
  <matplotlib.axis.XTick at 0x7fe2f76407f0>,
  <matplotlib.axis.XTick at 0x7fe2f5081278>,
  <matplotlib.axis.XTick at 0x7fe2f539bc88>,
  <matplotlib.axis.XTick at 0x7fe2f13260f0>,
  <matplotlib.axis.XTick at 0x7fe2f1326d30>,
  <matplotlib.axis.XTick at 0x7fe2f132d208>,
  <matplotlib.axis.XTick at 0x7fe2f132d6a0>,
  <matplotlib.axis.XTick at 0x7fe2f132db38>,
  <matplotlib.axis.XTick at 0x7fe2f132dd68>,
  <matplotlib.axis.XTick at 0x7fe2f13384a8>,
  <matplotlib.axis.XTick at 0x7fe2f132d780>,
  <matplotlib.axis.XTick at 0x7fe2f1326a20>,
  <matplotlib.axis.XTick at 0x7fe2f1338860>,
  <matplotlib.axis.XTick at 0x7fe2f1338b38>,
  <matplotlib.axis.XTick at 0x7fe2f1338d68>,
  <matplotlib.axis.XTick at 0x7fe2f133e4a8>,
  <matplotlib.axis.XTick at 0x7fe2f133e940>,
  <matplotlib.axis.XTick at 0x7fe2f133edd8>,
  <matplotlib.axis.XTick at 0x7fe2f13472b0>,
  <matplotlib.axis.XTick at 0x7fe2f1347748>,
  <matplotlib.axis.XTick at 0x7fe2f133eeb8>,
  <matplotlib.axis.XTick at 0x7fe2f1338748>,
  <matplotlib.axis.XTick at 0x7fe2f13477f0>,
  <matplotlib.axis.XTick at 0x7fe2f1347dd8>,
  <matplotlib.axis.XTick at 0x7fe2f12d12b0>,
  <matplotlib.axis.XTick at 0x7fe2f12d1748>,
  <matplotlib.axis.XTick at 0x7fe2f12d1be0>,
  <matplotlib.axis.XTick at 0x7fe2f12db128>,
  <matplotlib.axis.XTick at 0x7fe2f12db550>,
  <matplotlib.axis.XTick at 0x7fe2f12db9e8>,
  <matplotlib.axis.XTick at 0x7fe2f12d1898>,
  <matplotlib.axis.XTick at 0x7fe2f1347438>,
  <matplotlib.axis.XTick at 0x7fe2f1326588>,
  <matplotlib.axis.XTick at 0x7fe2f12db240>,
  <matplotlib.axis.XTick at 0x7fe2f12e7160>,
  <matplotlib.axis.XTick at 0x7fe2f12e7588>,
  <matplotlib.axis.XTick at 0x7fe2f12e7a20>],
 [Text(0, 0, 'abnormally'),
  Text(0, 0, 'about'),
  Text(0, 0, 'absence'),
  Text(0, 0, 'abut'),
  Text(0, 0, 'accentuates'),
  Text(0, 0, 'accessed'),
  Text(0, 0, 'account'),
  Text(0, 0, 'acuity'),
  Text(0, 0, 'additionally'),
  Text(0, 0, 'after'),
 Text(0, 0, 'ailla'),
 Text(0, 0, 'all'),
 Text(0, 0, 'although'),
 Text(0, 0, 'angulate'),
 Text(0, 0, 'annular'),
 Text(0, 0, 'anterolateral'),
 Text(0, 0, 'aorticopulmonary'),
 Text(0, 0, 'aorto'),
 Text(0, 0, 'appearances'),
 Text(0, 0, 'appreciable'),
 Text(0, 0, 'appropriate'),
 Text(0, 0, 'arterial'),
 Text(0, 0, 'arthroplasty'),
 Text(0, 0, 'articular'),
 Text(0, 0, 'artifactually'),
 Text(0, 0, 'asbestos'),
 Text(0, 0, 'aside'),
 Text(0, 0, 'aspergilloma'),
 Text(0, 0, 'aspiration'),
 Text(0, 0, 'assessment'),
 Text(0, 0, 'attached'),
 Text(0, 0, 'atypical'),
 Text(0, 0, 'avn'),
 Text(0, 0, 'back'),
 Text(0, 0, 'bacterial'),
 Text(0, 0, 'based'),
 Text(0, 0, 'bearing'),
 Text(0, 0, 'become'),
 Text(0, 0, 'believed'),
 Text(0, 0, 'benefit'),
 Text(0, 0, 'besides'),
 Text(0, 0, 'beyond'),
 Text(0, 0, 'bibasal'),
 Text(0, 0, 'bihilar'),
 Text(0, 0, 'biliary'),
 Text(0, 0, 'biopsy'),
 Text(0, 0, 'bochdalek'),
 Text(0, 0, 'bordered'),
 Text(0, 0, 'bronchopulmonary')])
```



Observation:

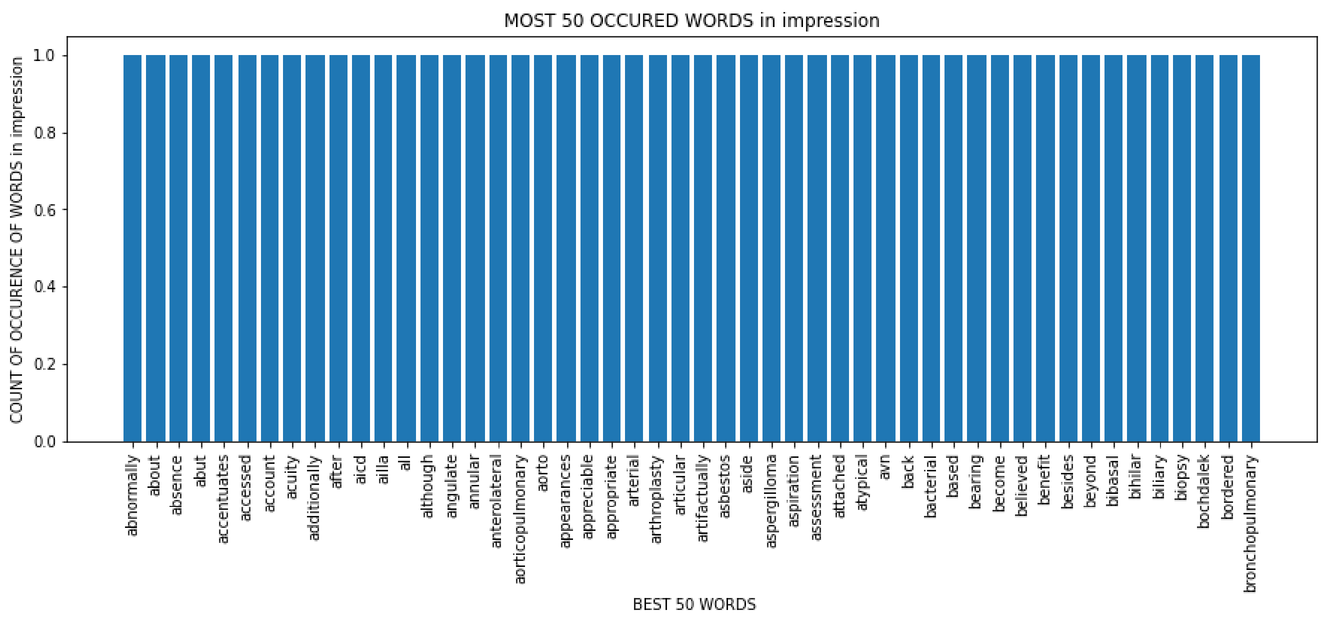
1. least 50 words give us the idea about that abnormal word is the least used in the findings. it means abnormality in chest x-ray data is less. It means most of the chest x-ray reports are normal.
2. the rare diseases words are very rare in the findings like bronchopulmonary, biopsy and many more. it means special medical terms are very less in the reports.

```
In [ ]:
```

```
plt.figure(figsize=(15,5))
plt.bar(range(50),[i for i,w in low_words_50_impression])
plt.title('MOST 50 OCCURED WORDS in impression')
plt.xlabel(' BEST 50 WORDS')
plt.ylabel('COUNT OF OCCURENCE OF WORDS in impression')
plt.xticks(ticks=range(50), labels=[word for i,word in low_words_50_impression],rotation=90)
```

Out[ ]:

```
([<matplotlib.axis.XTick at 0x7fe2f12cec88>,
  <matplotlib.axis.XTick at 0x7fe2f12cec50>,
  <matplotlib.axis.XTick at 0x7fe2f12ce898>,
  <matplotlib.axis.XTick at 0x7fe2f11f0898>,
  <matplotlib.axis.XTick at 0x7fe2f11f0d30>,
  <matplotlib.axis.XTick at 0x7fe2f11f0dd8>,
  <matplotlib.axis.XTick at 0x7fe2f11f9400>,
  <matplotlib.axis.XTick at 0x7fe2f11f9898>,
  <matplotlib.axis.XTick at 0x7fe2f11f9d30>,
  <matplotlib.axis.XTick at 0x7fe2f1201208>,
  <matplotlib.axis.XTick at 0x7fe2f12016a0>,
  <matplotlib.axis.XTick at 0x7fe2f1201b38>,
  <matplotlib.axis.XTick at 0x7fe2f1201d68>,
  <matplotlib.axis.XTick at 0x7fe2f12094a8>,
  <matplotlib.axis.XTick at 0x7fe2f12017f0>,
  <matplotlib.axis.XTick at 0x7fe2f11f9cf8>,
  <matplotlib.axis.XTick at 0x7fe2f1209828>,
  <matplotlib.axis.XTick at 0x7fe2f12090b8>,
  <matplotlib.axis.XTick at 0x7fe2f1194160>,
  <matplotlib.axis.XTick at 0x7fe2f1194588>,
  <matplotlib.axis.XTick at 0x7fe2f1194a20>,
  <matplotlib.axis.XTick at 0x7fe2f1194eb8>,
  <matplotlib.axis.XTick at 0x7fe2f119b390>,
  <matplotlib.axis.XTick at 0x7fe2f119b828>,
  <matplotlib.axis.XTick at 0x7fe2f1194898>,
  <matplotlib.axis.XTick at 0x7fe2f1209cf8>,
  <matplotlib.axis.XTick at 0x7fe2f119b9e8>,
  <matplotlib.axis.XTick at 0x7fe2f119beb8>,
  <matplotlib.axis.XTick at 0x7fe2f11a5390>,
  <matplotlib.axis.XTick at 0x7fe2f11a5828>,
  <matplotlib.axis.XTick at 0x7fe2f11a5cc0>,
  <matplotlib.axis.XTick at 0x7fe2f11ae198>,
  <matplotlib.axis.XTick at 0x7fe2f11ae630>,
  <matplotlib.axis.XTick at 0x7fe2f11aeac8>,
  <matplotlib.axis.XTick at 0x7fe2f11a5fd0>,
  <matplotlib.axis.XTick at 0x7fe2f119bba8>,
  <matplotlib.axis.XTick at 0x7fe2f11ae780>,
  <matplotlib.axis.XTick at 0x7fe2f11b8198>,
  <matplotlib.axis.XTick at 0x7fe2f11b8630>,
  <matplotlib.axis.XTick at 0x7fe2f11b8ac8>,
  <matplotlib.axis.XTick at 0x7fe2f11b8f60>,
  <matplotlib.axis.XTick at 0x7fe2f11c0438>,
  <matplotlib.axis.XTick at 0x7fe2f11c08d0>,
  <matplotlib.axis.XTick at 0x7fe2f11c0d68>,
  <matplotlib.axis.XTick at 0x7fe2f11c09b0>,
  <matplotlib.axis.XTick at 0x7fe2f11b8780>,
  <matplotlib.axis.XTick at 0x7fe2f11aed68>,
  <matplotlib.axis.XTick at 0x7fe2f12095f8>,
  <matplotlib.axis.XTick at 0x7fe2f11c8400>,
  <matplotlib.axis.XTick at 0x7fe2f11c8898>],
 [Text(0, 0, 'abnormally'),
  Text(0, 0, 'about'),
  Text(0, 0, 'absence'),
  Text(0, 0, 'abut'),
  Text(0, 0, 'accentuates'),
  Text(0, 0, 'accessed'),
  Text(0, 0, 'account'),
  Text(0, 0, 'acuity'),
  Text(0, 0, 'additionally'),
  Text(0, 0, 'after'),
  Text(0, 0, 'aicd'),
  Text(0, 0, 'ailla'),
  Text(0, 0, 'all'),
  Text(0, 0, 'although'),
  Text(0, 0, 'angulate'),
  Text(0, 0, 'annular'),
  Text(0, 0, 'anterolateral'),
  Text(0, 0, 'aorticopulmonary'),
 Text(0, 0, 'aorto'),
 Text(0, 0, 'appearances'),
 Text(0, 0, 'appreciable'),
 Text(0, 0, 'appropriate'),
 Text(0, 0, 'arterial'),
 Text(0, 0, 'arthroplasty'),
 Text(0, 0, 'articular'),
 Text(0, 0, 'artifactually'),
 Text(0, 0, 'asbestos'),
 Text(0, 0, 'aside'),
 Text(0, 0, 'aspergilloma'),
 Text(0, 0, 'aspiration'),
 Text(0, 0, 'assessment'),
 Text(0, 0, 'attached'),
 Text(0, 0, 'atypical'),
 Text(0, 0, 'avn'),
 Text(0, 0, 'back'),
 Text(0, 0, 'bacterial'),
 Text(0, 0, 'based'),
 Text(0, 0, 'bearing'),
 Text(0, 0, 'become'),
 Text(0, 0, 'believed'),
 Text(0, 0, 'benefit'),
 Text(0, 0, 'besides'),
 Text(0, 0, 'beyond'),
 Text(0, 0, 'bibasal'),
 Text(0, 0, 'bihilar'),
 Text(0, 0, 'biliary'),
 Text(0, 0, 'biopsy'),
 Text(0, 0, 'bochdalek'),
 Text(0, 0, 'bordered'),
 Text(0, 0, 'bronchopulmonary')])
```



Observation:

1. The obseration which we found in the findinsg are similar for the impression for the top 50 low words.
2. It means if we try to extract the information on the basis of impression and findings we have the similarities between them.

In [ ]:

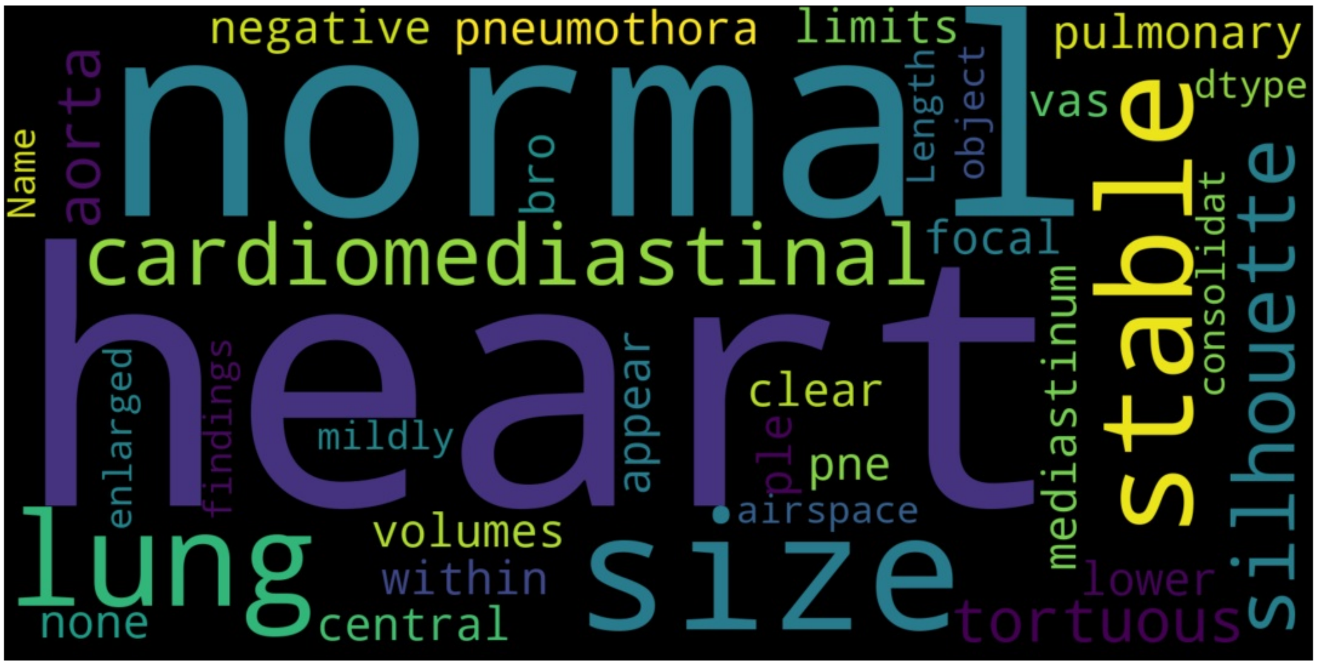
```
stopwords1 = set(STOPWORDS)#setting stopwords
```

In [ ]:

```
# generating word cloud of most occuring words ignoring stop words
wc = WordCloud(background_color="black", max_words=len(df),width=1600,
                          height=800, stopwords=stopwords1)
wc.generate(str(df["findings"]))
print("Word Cloud for findings ")
```

```
fig = plt.figure(figsize=(15,10))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

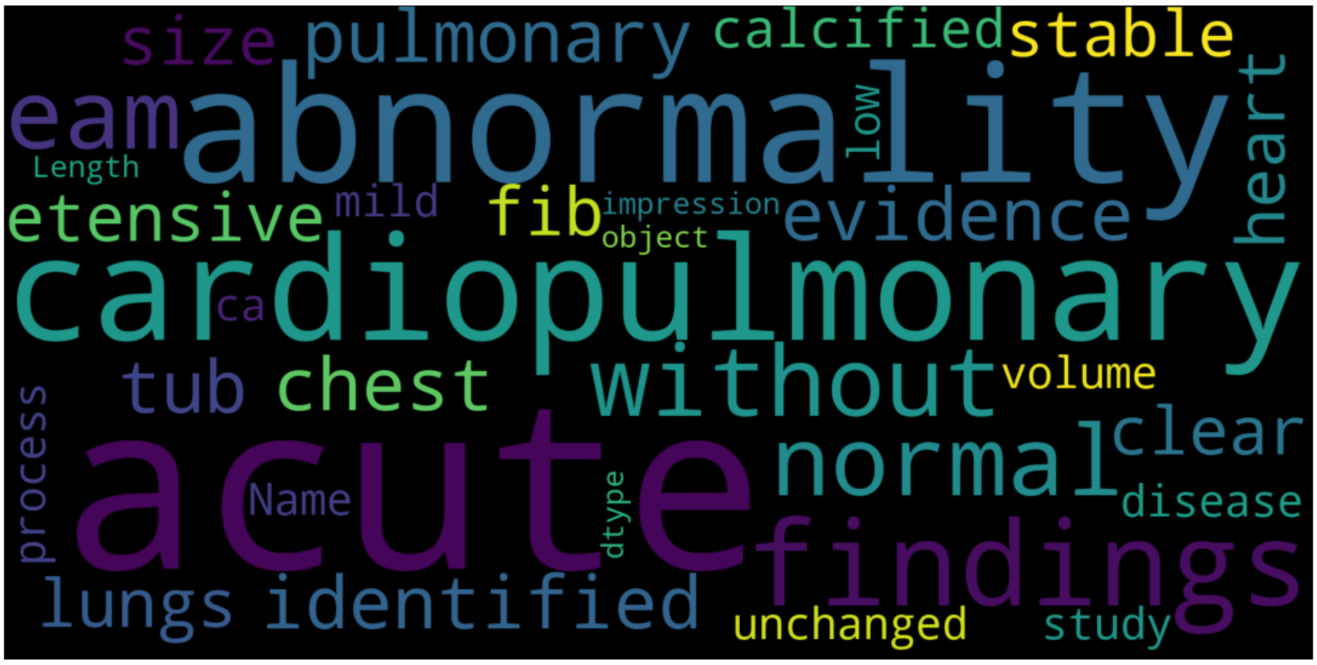
```
Word Cloud for findings
```



In [ ]:

```
# generating word cloud of most occuring words ignoring stop words
wc = WordCloud(background_color="black", max_words=len(df),width=1600,
                           height=800, stopwords=stopwords1)
wc.generate(str(df["impression"]))
print("Word Cloud for IMPRESSION ")
fig = plt.figure(figsize=(15,10))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

```
Word Cloud for IMPRESSION
```



In [ ]:

```
def remodelling_of_data(x):
  return 'start'+ ' ' + x + ' ' + 'end'
```

In [ ]:

```
df['impression']=df['impression'].apply(lambda x : remodelling_of_data(x))
```

In [ ]:

```
df['findings']=df['findings'].apply(lambda x : remodelling_of_data(x))
```

In [ ]:

```
df
```

Out[ ]:

	image_1	image_2	impression	findings
0	CXR3679_IM-1831-1001.png	CXR3679_IM-1831-2001.png	start normal chest eam end	start normal heart clear lungs no pneumothora ...
1	CXR3681_IM-1833-0001-0001.png	CXR3681_IM-1833-0001-0002.png	start no acute abnormality identified end	start there are lower lung volumes there centr...
2	CXR368_IM-1832-2001.png	CXR368_IM-1832-1001.png	start no acute cardiopulmonary findings end	start cardiomediastinal silhouette and pulmona...
3	CXR3665_IM-1823-1001.png	CXR3665_IM-1823-2001.png	start no acute abnormality no evidence pulmona...	start heart size within normal limits negative...
4	CXR3697_IM-1846-2001.png	CXR3697_IM-1846-1001.png	start no acute cardiopulmonary findings etensi...	start the cardiomediastinal silhouette stable ...
...	...	...	...	...
3527	CXR1842_IM-0545-1002.png	CXR1842_IM-0545-1001.png	start heart size normal lungs clear stable cal...	start none end
3528	CXR187_IM-0563-1001.png	CXR187_IM-0563-2001.png	start unchanged eam without acute abnormality end	start normal heart size stable tortuous aorta ...
3529	CXR185_IM-0551-1001.png	CXR185_IM-0551-2001.png	start no acute disease end	start the heart normal size the mediastinum st...
3530	CXR1851_IM-0553-2001.png	CXR1851_IM-0553-1001.png	start no acute cardiopulmonary findings end	start heart size normal no focal airspace cons...
3531	CXR1863_IM-0558-3001.png	CXR1863_IM-0558-1001.png	start low volume study without acute process m...	start heart size mildly enlarged tortuous aort...

3532 rows × 4 columns

Main observation of eda on findings and impression.

1. Impresion are the short explaination of chest x-ray dataset.
2. They both have the similarities if we try to extract the nformation with the help of words.
3. findings sentence length are much taller in comparison to impression sentence length. it means findings give the detailed explainantion of chest x-rays.
4. Based on the having count of the words special medical terms are less used in the findinsg and impression.
5. Findings and impression dictates that most of the reports are normal corresponding to the chest x ray. so it creates a imbalance when we try to train a model on this data. it mostly gives us the biased results because most of the reports are normal. so it is highly probable that my model after training gives the normal reports for the test chest x-ray data.To remove this problem we have to make a balance dataset by doing upsampling or downsampling.

In [ ]:

```
df.to_csv('preprocessed_data_with_only_frontal_lateral.csv',index=False)
```