```
import pandas as pd
import os
import spacy
from spacy.matcher import Matcher
from spacy.tokens import Doc

import numpy as np
from numpy import array
from sklearn.feature_extraction.text import CountVectorizer
from collections import Counter
from matplotlib import pyplot as plt
import seaborn as sb
import re

# Objective isto get medical care mentioned in the documents by using intelligent SpaCy
rule-based pattern matching.
# Sentences that contain patterns given below are retreived.
# The assumption is that medical cares are in the  sentences that contain the pattterns
# The output is csv file containing sha, keyword ans the sentences that contain the
keyword.

#from timer import  Timer
import time
#Use bag of to get gather words that are useful for medical care

nlp = spacy.load("en_core_web_sm")

pattern1 = [{"LOWER": "medical"}]
pattern2 = [{"LOWER": "care"}]
pattern3 = [{"LOWER": "diagnosis"}]
pattern4 = [{"LOWER": "treatment"}]
pattern5 = [{"LOWER": "therapy"}]
pattern5 = [{"LOWER": "effective"}]
pattern6 = [{"LOWER": "medical care"}]
pattern7 = [{"LOWER": "vaccine"}]
pattern8 = [{"LOWER": "vaccines"}]
pattern9 = [{"LOWER": "drug"}]
pattern10 = [{"LOWER": "vitamin"}]
pattern11 = [{"LOWER": "nursing"}]
pattern12 = [{"LOWER": "medical staff"}]
pattern13 = [{"LOWER": "Acute Respiratory Distress Syndrome"}] # not found in the text
pattern14 = [{"LOWER": "Extracorporeal membrane oxygenation"}]
pattern15 = [{"LOWER": "ventilation"}]
pattern16 = [{"LOWER": "manifestations"}]
pattern17 = [{"LOWER": "EUA"}]
pattern18 = [{"LOWER": "CLIA"}]
pattern19 = [{"LOWER": "elastomeric"}]
pattern20 = [{"LOWER": "N95"}]
pattern21 = [{"LOWER": "telemedicine"}]
pattern22 = [{"LOWER": "outcomes"}]




allWords = []

medical_care = [['', '', '']]

#customize_stop_words = [
#    'From','from', 'To', 'to', 'Hospital', 'hospital', '-', ')', '(', ',', ':', 'of',
'for', 'the', 'The', 'is',
#      '[', ']', ';', "\xa0", '/', 'virus', 'studies', '1', 'BACKGROUND', 'population',
'previously', 'countries', 'dogs', 'data',
#      'infection', '%', 'viral'
#]
```

```
customize_stop_words = [
     "\xa0", ','
]
for w in customize_stop_words:
    nlp.vocab[w].is_stop = True

os.chdir("/home/saul/corona/CORD-19-research-challenge/2020-03-13") # change this to
your local directory

filename = 'all_sources_metadata_2020-03-13.csv'



def readfile():

       textcount = 0
       print("reading the file")
       coronafile = pd.read_csv(filename, sep=',')

       #check dup rows
       duprows = coronafile[coronafile.duplicated(keep = False)]
       print(len(duprows))

       #save duplicate rows to csv
       duprows.to_csv("duplicates.csv")

       #remove duplicated rows
       nodupcorona = coronafile.drop_duplicates(subset=None, keep='first', inplace=False)
       print(len(nodupcorona))




       drop_list = ["WHO #Covidence"]

       nodupcorona = nodupcorona.drop(drop_list, axis=1)



       analyseAbstract(nodupcorona.sha, nodupcorona.abstract, textcount)
       print ("Text Count", textcount)


def analyseAbstract(sha, abstract, textcount):

       abstractList = [['', '']]
       abstract.dropna()
       #print('Abstract :', abstract)

       for sha, abst in zip(sha, abstract):
              abstractList.append([sha, abst]) #allocate each abstract into list

       cleanabstracts = [word for word in abstractList if str(word[1]) != 'nan']

       nlpWork(cleanabstracts, textcount)

def nlpWork(abstract, textcount):

       print(len(abstract))

       wordcount = 0
       for words in abstract:
              wordcount +=1

              coronaAnalysis(words[0], nlp(words[1]), wordcount, textcount)
```

```
def coronaAnalysis(sha, abstract, count, textcount):
      #doc = nlp(text)
      textcount = 0



      cleantext = [t.text for t in abstract if  not t.is_stop  and t.ent_type_ != 'GPE'
] # remove stop words. Exclude Geographic location

      # convert list to nlp doc
      cleandoc = Doc(nlp.vocab, words=cleantext)

      matcher = Matcher(nlp.vocab)



      matcher.add("medicalcare", None, pattern1, pattern2, pattern3, pattern4, pattern5,
pattern6,
                        pattern7, pattern8, pattern9, pattern10, pattern11, pattern12,
pattern13, pattern14, pattern15,
                        pattern16, pattern17, pattern18, pattern19, pattern20,
pattern21, pattern22)
      matches = matcher(cleandoc)

      for match_id, start, end in matches:

            moveleft = 0
            moveright = 0

            leftwords = []
            rightwords = []

            string_id = nlp.vocab.strings[match_id]  # Get string representation
            span = cleandoc[start:end]  # The matched span

            print(start, end, span.text)

            while ((len(cleandoc) >  start + moveleft) and (str(cleandoc[start -
moveleft]) != ".") ):


                  leftwords.append(cleandoc[start - moveleft])
                  moveleft= moveleft +1



                  if len(cleandoc) ==  start + moveleft:
                        break
            leftwords.reverse()



            while ((len(cleandoc) >  end + moveright) and (str(cleandoc[end +
moveright]) != ".") ):



                  moveright = moveright + 1



                  if len(cleandoc) ==  end + moveright:
                        break
                  rightwords.append(cleandoc[end + moveright])
```

```
            combinedList = leftwords + rightwords
            sentence = ' '.join(map(str, combinedList))
            sentence.replace(".","")
            #print("Combined Words ", combinedList, 'SHA ', sha, 'Keyword ', span.text)
            print("Sentence ", sentence, 'SHA ', sha, 'Keyword ', span.text)

            medical_care.append([sha, span.text, sentence])



if __name__ == '__main__':
      print("Process Started!!!")
      start = time.time()

      readfile()

      print("word freq is being written into csv")
      #bow.to_csv('wordfreq.csv', sep=',', index=False)
      df = pd.DataFrame(medical_care, columns=['sha', 'keyword', 'medical_care'])

      df.to_csv('medical_care.csv', sep=',', index=False)
      print("Medicare Data has been written into csv")

      print("Process Ended!!!")
      end = time.time()
      print("Time taken to run the code ", (end - start) // 60, " minutes")
```