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

import time
#nlp = spacy.load("en_core_web_sm")
nlp = spacy.load("en_core_web_lg")

# The objective is to get obtain non-pharmaceutical interventions mentioned in the documents by using intelligent SpaCy rule-based pattern matching.
# Sentences that contain below patterns are retrieved.
# The assumption is that non-pharmaceutical interventions are in the  sentences that contain the patterns
# The output is csv file containing sha, keyword and the sentences that contain the keyword.
#The patterns below generated by getting the simentically similar words with non-pharmaceutical (e.g non-pharmacological, therapeutical, non-chemical)  in en_core_web_lg dictionary.
# Download data to your local from kaggle and change the folders to your locations.
allWords = []

non_pharmaceutical = [['', '', '']]

pattern1= [{"LOWER":"non-pharmaceutical"}]
pattern2= [{"LOWER":"non-pharmacological"}]
pattern3= [{"LOWER":"non-drug"}]
pattern4= [{"LOWER":"nonpharmaceutical"}]
pattern5= [{"LOWER":"non-chemical"}]
pattern6= [{"LOWER":"alterantive"}]
pattern7= [{"LOWER":"psychopharmacological"}]
pattern8= [{"LOWER":"less-toxic"}]
pattern9= [{"LOWER":"LESS-TOXIC"}]
pattern10= [{"LOWER":"LOWER-TECH"}]
pattern11= [{"LOWER":"Lower-Tech"}]
pattern12= [{"LOWER":"lower-tech"}]
pattern13= [{"LOWER":"better/cheaper"}]
pattern14= [{"LOWER":"altenative"}]
pattern15= [{"LOWER":"less-effective"}]
pattern16= [{"LOWER":"Low-Technology"}]
pattern17= [{"LOWER":"low-technology"}]
pattern18= [{"LOWER":"LOW-TECHNOLOGY"}]
pattern19= [{"LOWER":"nonhormonal"}]
pattern20= [{"LOWER":"LESS-INTRUSIVE"}]
pattern21= [{"LOWER":"less-intrusive"}]
pattern22= [{"LOWER":"Less-Intrusive"}]
pattern23= [{"LOWER":"non-coercive"}]
pattern24= [{"LOWER":"nonsurgical"}]
pattern25= [{"LOWER":"homepathic"}]
pattern26= [{"LOWER":"NONADDICTIVE"}]
pattern27= [{"LOWER":"non-surgical"}]
pattern28= [{"LOWER":"anti-depression"}]
pattern29= [{"LOWER":"nutraceutical"}]
pattern30= [{"LOWER":"pharmacuetical"}]
pattern31= [{"LOWER":"preventions"}]
pattern32= [{"LOWER":"less-traditional"}]
pattern33= [{"LOWER":"therapeutical"}]
pattern34= [{"LOWER":"pain-relief"}]
pattern35= [{"LOWER":"biotechnical"}]
pattern36= [{"LOWER":"marine-based"}]
pattern37= [{"LOWER":"self-treatment"}]
```

```
pattern38= [{"LOWER":"immune-based"}]
pattern39= [{"LOWER":"treatements"}]
pattern40= [{"LOWER":"NPI"}]
pattern41= [{"LOWER":"DHS"}]

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'


#get words similar to non-pharmaceutical
def most_similar(word):
    by_similarity = sorted(word.vocab, key=lambda w: word.similarity(w), reverse=True)
    return [w.orth_ for w in by_similarity[:100]]

results = most_similar(nlp.vocab[u'non-pharmaceutical'])

print(results)

for key in range(len(results)):
    print(results[key])


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("non-pharmaceutical", None, pattern1, pattern2, pattern3, pattern4, pattern5,
pattern6,
                pattern7, pattern8, pattern9, pattern10, pattern11, pattern12, pattern13, pattern14,
pattern15,
                pattern16, pattern17, pattern18, pattern19, pattern20, pattern21, pattern22,
pattern23, pattern24,
                pattern25, pattern26, pattern27, pattern28, pattern29, pattern30, pattern31,
pattern32, pattern33,
                pattern34, pattern35, pattern36, pattern37, pattern38, pattern39, pattern40,
pattern41)
    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)

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


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

    readfile()

    df = pd.DataFrame(non_pharmaceutical, columns=['sha', 'keyword', 'non-pharmaceutical'])

    df.to_csv('non-pharmaceutical.csv', sep=',', index=False)
    print("non-pharmaceutical interventions has been written into csv")

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