# Filter 1 Function:

def nerfilter(fileName):

# Obtaining data

df = pd.read\_csv('../COde/Combined/Investigation2 (3).txt', sep='\t',encoding='utf-8')

df2 = pd.read\_csv('../COde/Combined/NER List (Combined).txt', sep='\t',encoding='utf-8')

#Data to lists

nutshellList = list(df['Nutshell'])

nerlist = list(df2['NER List'])

nerlabel = list(df2['NER Label'])

nertup = list(zip(nerlist,nerlabel))

#Comparison Logic

applist = []

for nut in nutshellList:

doc = nlp(nut)

applicablity = "Yes"

for ent in doc.ents:

for ner,label in nertup:

if ent.text.lower() == ner.lower() and ent.label\_.lower() == label.lower():

applicablity = "No"

applist.append(applicablity)

#Adding to the dataframe

df['AppFilter1'] = applist

#Righting to the file

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 2 Function

def ncfilter(fileName):

# Obtaining data

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

df2 = pd.read\_csv('../COde/Combined/Noun Chunk List (OR) v2.0 (Combined).txt', sep='\t',encoding='utf-8')

#Data to lists

nutshell = list(df['Nutshell'])

appfil1 = list(df['AppFilter1'])

filter1tup = list(zip(nutshell,appfil1))

nclist = list(df2['Noun Chunk List (OR)'])

#Comparison Logic

applist = []

for nut,fil1 in filter1tup:

applicablity = "Yes"

if fil1 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

for nounC in doc.noun\_chunks:

for nc in nclist:

if nounC.text.lower() == nc.lower():

applicablity = "No"

applist.append(applicablity)

#Adding to the Dataframe

df['AppFilter2'] = applist

#Save to file

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 3 Function

def nerncfilter(fileName):

# Obtaining data

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

df2 = pd.read\_csv('../COde/Combined/NER and NC list (combined).txt', sep='\t',encoding='utf-8')

#Data to lists

nutshell = list(df['Nutshell'])

appfil2 = list(df['AppFilter2'])

filter2tup = list(zip(nutshell,appfil2))

nerlist = list(df2['NER'])

nerLabelList = list(df2['NER Label'])

nclist = list(df2['NC'])

combtup = list(zip(nerlist,nerLabelList,nclist))

#Comparison Logic

applist = []

for nut,fil2 in filter2tup:

applicability = "Yes"

if fil2 == "No":

applicability = "No"

applist.append(applicability)

else:

doc = nlp(nut)

for ner,nerla,ncl in combtup:

for ent in doc.ents:

if ent.text.lower() == ner.lower() and ent.label\_.lower() == nerla.lower():

for nc in doc.noun\_chunks:

if ncl.lower() == nc.text.lower():

applicability = "No"

applist.append(applicability)

#Adding to the Dataframe

df['AppFilter3'] = applist

#Save to File

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 4 Function

def ncandfilter(fileName):

# Obtaining data

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

df2 = pd.read\_csv('../COde/Combined/Noun Chunk List (AND) (combined).txt', sep='\t',encoding='utf-8')

#Data to lists

nutshell = list(df['Nutshell'])

appfil3 = list(df['AppFilter3'])

filter3tup = list(zip(nutshell,appfil3))

nclist1 = list(df2['Noun Chunk and List'])

nclist2 = list(df2['Unnamed: 1'])

nctup = list(zip(nclist1,nclist2))

#Comparison Logic

applist = []

for nut,fil3 in filter3tup:

applicablity = "Yes"

if fil3 == "No":

applicablity = "No"

applist.append(applicablity)

else:

indicator1 = False

indicator2 = False

doc = nlp(nut)

for ncl1,ncl2 in nctup:

for nc1 in doc.noun\_chunks:

if nc1.text.lower() == ncl1.lower():

for nc2 in doc.noun\_chunks:

if nc2.text.lower() == ncl2.lower():

applicablity = "No"

applist.append(applicablity)

#Adding to the Dataframe

df['AppFilter4'] = applist

#Save to File

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 5 Function

def entityruler():

from spacy.lang.en import English

from spacy.pipeline import EntityRuler

nlp = English()

#Creating an instance of Entity Rule

ruler = EntityRuler(nlp)

#List of the Entities that need to be a file upload

patterns = [{"label": "LAW", "pattern": [{"lower": "national"}, {"lower": "road"},{"lower":"traffic"},{"lower":"regulations"}]}]

pattern1 = [{"label": "ORG", "pattern": [{"lower": "cargo"}, {"lower": "reporters"}]}]

pattern2 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "national"},{"lower":"small"},{"lower":"enterprise"},{"lower":"act"}]}]

pattern3 = [{"label": "WORK\_OF\_ART", "pattern": [{"lower": "crime"}, {"lower": "statistics"}]}]

pattern4 = [{"label": "LAW", "pattern": [{"lower": "liquor"}, {"lower": "bill"}]}]

pattern5 = [{"label": "ORG", "pattern": [{"lower": "broadcasters"}]}]

pattern6 = [{"label": "LAW", "pattern": [{"lower": "political"}, {"lower": "party"},{"lower":"funding"},{"lower":"bill"}]}]

pattern7 = [{"label": "LAW", "pattern": [{"lower": "planning"}, {"lower": "profession"},{"lower":"act"}]}]

pattern8 = [{"label": "LAW", "pattern": [{"lower": "the"},{"lower": "end-user"}, {"lower": "subscriber"},{"lower":"service"},{"lower":"charter"},{"lower":"amendment"},{"lower":"regulations"}]}]

pattern9 = [{"label": "LAW", "pattern": [{"lower": "integrated"}, {"lower": "planning"},{"lower":"framework"},{"lower":"bill"}]}]

pattern10 = [{"label": "LAW", "pattern": [{"lower": "national"}, {"lower": "road"},{"lower":"traffic"},{"lower":"regulations"}]}]

pattern11 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "public"},{"lower":"finance"},{"lower":"management"},{"lower":"act"}]}]

pattern12 = [{"label": "ORG", "pattern": [{"lower": "the"}, {"lower": "communications"},{"lower":"regulatory"},{"lower":"authority"}]}]

pattern13 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "namibia"},{"lower":"revenue"},{"lower":"agency"},{"lower":"act"}]}]

pattern14 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "communications"},{"lower":"act"}]}]

pattern15 = [{"label": "LAW", "pattern": [{"lower": "investment"}, {"lower": "promotion"},{"lower":"act"}]}]

pattern16 = [{"label": "LAW", "pattern": [{"lower": "namibia"}, {"lower": "time"},{"lower":"act"}]}]

pattern17 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "namibia"},{"lower":"special"},{"lower":"risks"},{"lower":"insurance"},{"lower":"association"},{"lower":"act"}]}]

pattern18 = [{"label": "ORG", "pattern": [{"lower": "the"}, {"lower": "law"},{"lower":"society"}]}]

pattern19 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "namibian"},{"lower":"medical"},{"lower":"aid"},{"lower":"funds"},{"lower":"act"}]}]

pattern20 = [{"label": "ORG", "pattern": [{"lower": "bank"}, {"lower": "of"},{"lower":"mozambique"},{"lower":"advisory"},{"lower":"council"}]}]

pattern21 = [{"label": "LAW", "pattern": [{"lower": "customs"}, {"lower": "tariff"}]}]

pattern22 = [{"label": "ORG", "pattern": [{"lower": "the"}, {"lower": "lesotho"},{"lower":"citizenship"},{"lower":"(amendment"},{"lower":"of"},{"lower":"schedule)"},{"lower":"regulations"}]}]

pattern23 = [{"label": "LAW", "pattern": [{"lower": "land"}, {"lower": "commission"},{"lower":"act"}]}]

pattern24 = [{"label": "LAW", "pattern": [{"lower": "public"}, {"lower": "health"},{"lower":"bill"}]}]

pattern25 = [{"label": "LAW", "pattern": [{"lower": "public"}, {"lower": "health"},{"lower":"act"}]}]

pattern26 = [{"label": "LAW", "pattern": [{"lower": "labour"}, {"lower": "court"},{"lower":"rules"}]}]

pattern27 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "zimbabwe"},{"lower":"public"},{"lower":"procurement"},{"lower":"and"},{"lower":"disposal"},{"lower":"of"},{"lower":"public"},{"lower":"assets"}]}]

pattern28 = [{"label": "LAW", "pattern": [{"lower": "diamond"}, {"lower": "cutting"},{"lower":"act"}]}]

pattern29 = [{"label": "LAW", "pattern": [{"lower": "diamond"}, {"lower": "cutting"},{"lower":"bill"}]}]

pattern30 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "precious"},{"lower":"and"},{"lower":"semi-precious"},{"lower":"stones"},{"lower":"protection"},{"lower":"act"}]}]

pattern31 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "precious"},{"lower":"and"},{"lower":"semi-precious"},{"lower":"stones"},{"lower":"protection"},{"lower":"bill"}]}]

pattern32 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "national"},{"lower":"registration"},{"lower":"(amendment)"},{"lower":"bill"}]}]

pattern33 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "national"},{"lower":"registration"},{"lower":"(amendment)"},{"lower":"act"}]}]

pattern34 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "plant"},{"lower":"protection"},{"lower":"act"}]}]

pattern35 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "plant"},{"lower":"protection"},{"lower":"bill"}]}]

pattern36 = [{"label": "ORG", "pattern": [{"lower": "electricity"}, {"lower": "company"}]}]

pattern37 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "industrial"},{"lower":"property"},{"lower":"law"}]}]

pattern38 = [{"label": "ORG", "pattern": [{"lower": "Private"}, {"lower": "security"},{"lower":"companies"}]}]

pattern39 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "general"},{"lower":"law"},{"lower":"on"},{"lower":"archives"}]}]

pattern40 = [{"label": "LAW", "pattern": [{"lower": "national"}, {"lower": "flag"},{"lower":"act"}]}]

pattern41 = [{"label": "LAW", "pattern": [{"lower": "national"}, {"lower": "flag"},{"lower":"(amendment)"},{"lower":"act"}]}]

pattern42 = [{"label": "LAW", "pattern": [{"lower": "national"}, {"lower": "flag"},{"lower":"amendment"},{"lower":"act"}]}]

pattern43 = [{"label": "LAW", "pattern": [{"lower": "Copyright"}, {"lower": "(amendment)"},{"lower":"act"}]}]

pattern44 = [{"label": "LAW", "pattern": [{"lower": "Copyright"},{"lower":"act"}]}]

pattern45 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "mauritius"},{"lower":"allied"},{"lower":"health"},{"lower":"professionals"},{"lower":"council"},{"lower":"act"}]}]

pattern46 = [{"label": "ORG", "pattern": [{"lower": "the"}, {"lower": "development"},{"lower":"and"},{"lower":"classification"},{"lower":"of"},{"lower":"film"},{"lower":"act"}]}]

pattern47 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "tobacco"},{"lower":"control"},{"lower":"regulations"}]}]

pattern48 = [{"label": "LAW", "pattern": [{"lower": "namibian"}, {"lower": "time"},{"lower":"act"}]}]

pattern49 = [{"label": "LAW", "pattern": [{"lower": "diamond"}, {"lower": "cutting"}, {"lower": "(amendment)"},{"lower":"act"}]}]

pattern50 = [{"label": "LAW", "pattern": [{"lower": "diamond"}, {"lower": "cutting"}, {"lower": "(amendment)"},{"lower":"bill"}]}]

pattern51 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "precious"},{"lower":"and"},{"lower":"semi-precious"},{"lower":"stones"},{"lower":"protection"}, {"lower": "(amendment)"},{"lower":"act"}]}]

pattern52 = [{"label": "LAW", "pattern": [{"lower": "the"}, {"lower": "precious"},{"lower":"and"},{"lower":"semi-precious"},{"lower":"stones"},{"lower":"protection"}, {"lower": "(amendment)"},{"lower":"bill"}]}]

pattern53 = [{"label": "LAW", "pattern": "Public Finance Management Act"}]

pattern54 = [{"label": "ORG", "pattern": "The Lesotho Citizenship (Amendment of Schedule) Regulations"}]

pattern55 = [{"label": "LAW", "pattern": "Diamond Cutting (Amendment) Bill"}]

pattern56 = [{"label": "LAW", "pattern": "The Precious and Semi-Precious Stones Protection (Amendment) Bill"}]

pattern57 = [{"label": "LAW", "pattern": "The National Registration (Amendment) Bill"}]

pattern58 = [{"label": "LAW", "pattern": "National Flag (Amendment) Bill"}]

pattern59 = [{"label": "LAW", "pattern": "National Flag (Amendment) Act"}]

pattern60 = [{"label": "LAW", "pattern": "Copyright (Amendment) Bill"}]

pattern61 = [{"label": "ORG", "pattern": "Moody's Investors Service"}]

pattern62 = [{"label": "LAW", "pattern": "the Industrial Property Law"}]

pattern63 = [{"label": "ORG", "pattern": 'Private Sercurity Companies'}]

#Adding the Patterns to the entity ruler

ruler.add\_patterns(patterns)

ruler.add\_patterns(pattern1)

ruler.add\_patterns(pattern2)

ruler.add\_patterns(pattern3)

ruler.add\_patterns(pattern4)

ruler.add\_patterns(pattern5)

ruler.add\_patterns(pattern6)

ruler.add\_patterns(pattern7)

ruler.add\_patterns(pattern8)

ruler.add\_patterns(pattern9)

ruler.add\_patterns(pattern10)

ruler.add\_patterns(pattern11)

ruler.add\_patterns(pattern12)

ruler.add\_patterns(pattern13)

ruler.add\_patterns(pattern14)

ruler.add\_patterns(pattern15)

ruler.add\_patterns(pattern16)

ruler.add\_patterns(pattern17)

ruler.add\_patterns(pattern18)

ruler.add\_patterns(pattern19)

ruler.add\_patterns(pattern20)

ruler.add\_patterns(pattern21)

ruler.add\_patterns(pattern22)

ruler.add\_patterns(pattern23)

ruler.add\_patterns(pattern24)

ruler.add\_patterns(pattern25)

ruler.add\_patterns(pattern26)

ruler.add\_patterns(pattern27)

ruler.add\_patterns(pattern28)

ruler.add\_patterns(pattern29)

ruler.add\_patterns(pattern30)

ruler.add\_patterns(pattern31)

ruler.add\_patterns(pattern32)

ruler.add\_patterns(pattern33)

ruler.add\_patterns(pattern34)

ruler.add\_patterns(pattern35)

ruler.add\_patterns(pattern36)

ruler.add\_patterns(pattern37)

ruler.add\_patterns(pattern38)

ruler.add\_patterns(pattern39)

ruler.add\_patterns(pattern40)

ruler.add\_patterns(pattern41)

ruler.add\_patterns(pattern42)

ruler.add\_patterns(pattern43)

ruler.add\_patterns(pattern44)

ruler.add\_patterns(pattern45)

ruler.add\_patterns(pattern46)

ruler.add\_patterns(pattern47)

ruler.add\_patterns(pattern48)

ruler.add\_patterns(pattern49)

ruler.add\_patterns(pattern50)

ruler.add\_patterns(pattern51)

ruler.add\_patterns(pattern52)

ruler.add\_patterns(pattern53)

ruler.add\_patterns(pattern54)

ruler.add\_patterns(pattern55)

ruler.add\_patterns(pattern56)

ruler.add\_patterns(pattern57)

ruler.add\_patterns(pattern58)

ruler.add\_patterns(pattern59)

ruler.add\_patterns(pattern60)

ruler.add\_patterns(pattern61)

ruler.add\_patterns(pattern62)

ruler.add\_patterns(pattern63)

#Adding the instance of the Entity Ruler into the Pipeline

nlp.add\_pipe(ruler)

#Reading from the List

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil4 = list(df["AppFilter4"])

fil4tup = list(zip(nutshellList,appfil4))

df2 = pd.read\_csv('../COde/Combined/NER Phrase Matcher (combined).txt', sep='\t',encoding='utf-8')

phrase = list(df2['Phrase Matcher'])

nerlabelList = list(df2['NER Label'])

matchingtup = list(zip(phrase,nerlabelList))

hitlist = []

#Logic

for nut,app in fil4tup:

applicability = 'Yes'

if app == "No":

applicability = 'No'

hitlist.append(applicability)

else:

doc = nlp(nut)

for ent in doc.ents:

for ner,nerL in matchingtup:

if ent.text.lower() == ner.lower() and ent.label\_.lower() == nerL.lower():

applicability = 'No'

hitlist.append(applicability)

#Saving to the Dataframe

df["Hit List"] = hitlist

#Save to File

return df.to\_csv('../COde/Combined/OutputFIle.txt', sep='\t',encoding='utf-8')

# Filter6 Function

def phrasematch():

from spacy.matcher import PhraseMatcher

from spacy.lang.en import English

nlp = English()

#Creating an instance of the Phrase

matcher = PhraseMatcher(nlp.vocab, attr = "LOWER")

#Importing the lists

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil5 = list(df["Hit List"])

fil5tup = list(zip(nutshellList,appfil5))

df2 = pd.read\_csv('../COde/Combined/singleWord(PhraseMatcher) (combined).txt', sep='\t',encoding='utf-8')

patternlist = list(df2['Word'])

#Creating a pattern list from dataframe2

patterns = [nlp.make\_doc(text) for text in patternlist]

#Adding the pattern list to the instance of the phraematcher

matcher.add("NounList",None,\*patterns)

#Logic

hitlist = []

for nut,app5 in fil5tup:

applicability = "Yes"

if app5 == "No":

applicability = 'No'

hitlist.append(applicability)

else:

doc = nlp(nut)

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

for pat in patternlist:

if span.text.lower() == pat.lower():

applicability = "No"

hitlist.append(applicability)

#Adding to the Dataframe

df['AppFilter6'] = hitlist

#Save to File

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 7 Function

def nerncphrasematcher():

#Special Filter #1 - NER and NC PhraseMatcher

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil6 = list(df["AppFilter6"])

fil6tup = list(zip(nutshellList,appfil6))

#This will have to be in a list later

df2 = pd.read\_csv('../COde/Combined/Special#1 - NER and NC\_PM.txt', sep='\t',encoding='utf-8')

entCheck = list(df2['NER '])

entCheckLabel = list(df2['Label'])

patternlist = list(df2['NC'])

entityList = list(zip(entCheck,entCheckLabel,patternlist))

from spacy.matcher import PhraseMatcher

#from spacy.lang.en import English

matcher = PhraseMatcher(nlp.vocab, attr = "LOWER")

patterns = [nlp.make\_doc(text) for text in patternlist]

matcher.add("NounList",None,\*patterns)

applist = []

for nut,fil6 in fil6tup:

applicablity = "Yes"

if fil6 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

for entityName, entityLabel, patC in entityList:

for ent in doc.ents:

if entityName.lower() == ent.text.lower() and entityLabel.lower() == ent.label\_.lower():

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

if span.text.lower() == patC.lower():

applicablity = "No"

applist.append(applicablity)

df['AppFilter7'] = applist

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 8 Function

def nerpm\_ncpm():

#Special Filter #2 - NER Entity Ruler and NC Phrase Matcher

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil7 = list(df["AppFilter7"])

fil7tup = list(zip(nutshellList,appfil7))

from spacy.lang.en import English

from spacy.pipeline import EntityRuler

nlp2 = English()

#Creating an instance of the entity ruler

ruler = EntityRuler(nlp2)

#the below needs to be fed into a list

patternEnt = [{"label": "ORG", "pattern": [{"lower": "bargaining"}, {"lower": "council"}]}]

ruler.add\_patterns(patternEnt)

nlp2.add\_pipe(ruler)

#this will need to be a list going forward

entCheck = ['Bargaining Council']

entCheckLabel = ['ORG']

patternlist = ['furniture']

entityTup = list(zip(entCheck,entCheckLabel,patternlist))

from spacy.matcher import PhraseMatcher

#Creating an instance of the phrasematcher

matcher = PhraseMatcher(nlp.vocab, attr = "LOWER")

patterns = [nlp.make\_doc(text) for text in patternlist]

matcher.add("NounList",None,\*patterns)

#Logic

applist = []

for nut,fil7 in fil7tup:

applicablity = "Yes"

if fil7 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

doc2 = nlp2(nut)

for entC,entCL,patC in entityTup:

for ent in doc2.ents:

if entC.lower() == ent.text.lower() and entCL.lower() == ent.label\_.lower():

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

if span.text.lower() == patC.lower():

applicablity = "No"

applist.append(applicablity)

#Saving to the Dataframe

df['AppFilter8'] = applist

#Saving the File

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 9 Function

def nc\_ncpm():

#Special Filter #3 - NC and NC\_PM

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil8 = list(df["AppFilter8"])

fil8tup = list(zip(nutshellList,appfil8))

#Reading the list of the Noun Chunks for exact matching and the phrase matcher noun chunks

df2 = pd.read\_csv('../COde/Combined/Special#3 - NC and NC\_PM.txt', sep='\t',encoding='utf-8')

nounCList = list(df2['NC'])

nounCPMList = list(df2['NC PM'])

nounTup = list(zip(nounCList,nounCPMList))

from spacy.matcher import PhraseMatcher

#Creating an instance of the phrase matcher

matcher = PhraseMatcher(nlp.vocab, attr = "LOWER")

#Creating a list of the patterns to be fed into the phrase matcher

patterns = [nlp.make\_doc(text) for text in nounCPMList]

#Adding the phrase matcher list to the instance of the Phrasematcher

matcher.add("NounList",None,\*patterns)

#Logic

applist = []

for nut,fil8 in fil8tup:

applicablity = "Yes"

if fil8 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

for nc1,nc2 in nounTup:

for noun in doc.noun\_chunks:

if noun.text.lower() == nc1.lower():

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

if span.text.lower() == nc2.lower():

applicablity = "No"

applist.append(applicablity)

#Saving to the Dataframe

df['AppFilter9'] = applist

#Saving to file

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 10 Function

def nc\_and\_nc\_and\_nc():

# Special Filter #4 - NC and NC and NC

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil9 = list(df["AppFilter9"])

fil9tup = list(zip(nutshellList,appfil9))

#This needs to be fed into lists

nounCList = ['report','a report']

nounCList2 = ['comparison','a comparison']

nounCList3 = ['banking fees','banking fees']

nounCTup = list(zip(nounCList,nounCList2,nounCList3))

#Comparison Logic

applist = []

for nut,fil9 in fil9tup:

applicablity = "Yes"

if fil9 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

#Checking indicator1

for ncl1,ncl2,ncl3 in nounCTup:

for nc1 in doc.noun\_chunks:

if nc1.text.lower() == ncl1.lower():

for nc2 in doc.noun\_chunks:

if nc2.text.lower() == ncl2.lower():

for nc3 in doc.noun\_chunks:

if nc3.text.lower() == ncl3.lower():

applicablity = "No"

applist.append(applicablity)

#Saving into the dataframe

df['AppFilter10'] = applist

#Saving to file - this will be replaced with saving to the database

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 11 Function

def nc\_and\_nc\_and\_nc\_and\_nc():

# Special Filter #5 - NC and NC and NC and NC

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil10 = list(df["AppFilter10"])

fil10tup = list(zip(nutshellList,appfil10))

#This will need to be put into the database

nounCList = ['the government']

nounCList2 = ['the exchange']

nounCList3 = ['information']

nounCList4 = ['tax matters']

nounCTup = list(zip(nounCList,nounCList2,nounCList3,nounCList4))

#Comparison Logic

applist = []

for nut,fil10 in fil10tup:

applicablity = "Yes"

if fil10 == "No":

applicablity = "No"

applist.append(applicablity)

else:

indicator1 = False

indicator2 = False

indicator3 = False

indicator4 = False

doc = nlp(nut)

for ncl1,ncl2,ncl3,ncl4 in nounCTup:

for nc1 in doc.noun\_chunks:

if nc1.text.lower() == ncl1.lower():

for nc2 in doc.noun\_chunks:

if nc2.text.lower() == ncl2.lower():

for nc3 in doc.noun\_chunks:

if nc3.text.lower() == ncl3.lower():

for nc4 in doc.noun\_chunks:

if nc4.text.lower() == ncl4.lower():

applicablity = "No"

applist.append(applicablity)

#Saving to dataframe

df['AppFilter11'] = applist

#Saving to file - This will be replaced with the database

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 12 Function

def ner\_ncpm\_ncpm():

#Special Filter #6 - NER and NC PM and NC PM

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil11 = list(df["AppFilter11"])

fil11tup = list(zip(nutshellList,appfil11))

#This will have to be in a list later

entCheck = ['the Bank of Mozambique']

entCheckLabel = ['ORG']

patternlist = ['treasury bills']

patternlist2 = ['auction']

entityList = list(zip(entCheck,entCheckLabel,patternlist,patternlist2))

from spacy.matcher import PhraseMatcher

matcher = PhraseMatcher(nlp.vocab, attr = "LOWER")

patterns = [nlp.make\_doc(text) for text in patternlist]

matcher.add("NounList",None,\*patterns)

patterns2 = [nlp.make\_doc(text) for text in patternlist2]

matcher.add("NounList",None,\*patterns2)

applist = []

for nut,fil11 in fil11tup:

applicablity = "Yes"

if fil11 == "No":

applicablity = "No"

applist.append(applicablity)

else:

indicator1 = False

indicator2 = False

indicator3 = False

doc = nlp(nut)

#Checking indicator1

for entityName, entityLabel, patC1, patC2 in entityList:

for ent in doc.ents:

if entityName.lower() == ent.text.lower() and entityLabel.lower() == ent.label\_.lower():

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

if span.text.lower() == patC1.lower():

indicator1 = True

if span.text.lower() == patC2.lower():

indicator2 = True

if indicator1 == True and indicator2 == True:

applicablity = "No"

applist.append(applicablity)

df['AppFilter12'] = applist

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Filter 13 Function

def nerCapncpm():

#Special Filter #7 - NER and Capital NC PM

df = pd.read\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

nutshellList = list(df['Nutshell'])

appfil12 = list(df["AppFilter12"])

fil12tup = list(zip(nutshellList,appfil12))

#This will have to be in a list later on and loop through the list in indicator 1

#df2 = pd.read\_csv('../COde/Combined/Special#1 - NER and NC\_PM.txt', sep='\t',encoding='utf-8')

entCheck = ['the Mozambique Cabinet']

entCheckLabel = ['ORG']

patternlist = ['Policy']

entityList = list(zip(entCheck,entCheckLabel,patternlist))

from spacy.matcher import PhraseMatcher

#from spacy.lang.en import English

matcher = PhraseMatcher(nlp.vocab)

patterns = [nlp.make\_doc(text) for text in patternlist]

matcher.add("NounList",None,\*patterns)

applist = []

for nut,fil12 in fil12tup:

applicablity = "Yes"

if fil12 == "No":

applicablity = "No"

applist.append(applicablity)

else:

doc = nlp(nut)

for entityName, entityLabel, patC in entityList:

for ent in doc.ents:

if entityName.lower() == ent.text.lower() and entityLabel.lower() == ent.label\_.lower():

matches = matcher(doc)

for mid,mstart,mend in matches:

span = doc[mstart:mend]

if span.text == patC:

applicablity = "No"

applist.append(applicablity)

df['AppFilter13'] = applist

return df.to\_csv('../COde/Combined/OutputFile.txt', sep='\t',encoding='utf-8')

# Documents

Filters:

