```python
# coding=utf-8
# Downloads all the MAUDE (including FOIDEV and MDRFOI) files from the FDA
 Website
# Searches for the records related to a specific device
# Merges the FIODEV and MDRFOI files into one Excel sheet
# Goes through MDR keys and opens up the MDR reports on FDA website to grab
 and add
# Patient outcome, Event Description and Narrative, and Number of Devices
 to the table

import bs4
import cookielib
import csv
import os
import re
import string
import time
import urllib
import urllib2
import xlrd
import xlsxwriter  # xlwt不支持超过256行的写入,需要采用xlsxwriter
import xlwt
from datetime import date
from dateutil import parser
from nltk.corpus import stopwords
from nltk.probability import ConditionalFreqDist
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from zipfile import ZipFile

cj = cookielib.CookieJar()
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
import requests
from tqdm import tqdm
import time
import lxml


# Extract the fields from each record
def field_extract(line, field_numbers):
    fields = line.split('|')
    # print('ljn: %d' % (len(fields))) # some is 28, lower than 45
    # print('ljn: %s' % line) # one record is not complete

    extracted = []  # [0 for x in range(0,len(line))]
    try:  # ljn changed
        for f in field_numbers:
            extracted.append(fields[f - 1].strip())
        # print extracted
        return extracted
    except Exception, e:
        print('Record  occurs errors, record is %s' % line)
        print(e.message)
        return None
```

```
# Download the data files from MAUDE database and save it
def maude_download(foidev_files, mdrfoi_files, data_dir):
    # maude_url =
     'https://www.fda
     .gov/medical-devices/mandatory-reporting-requirements-manufacturers-imp
     orters-and
    #
     -device-user-facilities/manufacturer-and-user-facility-device-experienc
     e-database-maude'
    maude_url = 'https://www.accessdata.fda.gov/MAUDE/ftparea/'  # ljn added
    os.chdir(data_dir)
    # Download All foidev files
    for filename in foidev_files + mdrfoi_files:
        # Download the Zip file
        with open(data_dir + filename + '.zip', 'wb') as zfile:
            print(maude_url + filename + '.zip')
            zfile.write(urllib2.urlopen(maude_url + filename +
             '.zip').read())
        # Extract the Zip file
        zip_data = ZipFile(data_dir + filename + '.zip',
         'r').extractall(data_dir)  # ljn changed
        # Clean up the folder by deleting the Zip file
        os.remove(data_dir + filename + '.zip')
        print(filename + ' downloaded.')


# Extract the FOIDEV records and append them to the file
def foidev_extract(foidev_files,
                   foidev_field_numbers,
                   device_name,
                   device_keywords,
                   data_dir):  # ljn changed
    """从foidevxxxx.txt文件中提取出相应的字段，"""
    foidev_count = 0
    os.chdir(data_dir)

    print('Starting to extract da Vinci related records..')
    # Extract those related to the device
    for filename in foidev_files:
        # If the first file, first get the titles
        # FOIDEV_Field_Numbers = get_numbers(filename) # ljn added
        if filename == foidev_files[0]:
            with open(filename + '.txt', "rb") as foidev_file:
                title = foidev_file.next()

                # Create the Hash Table of FOIDEV records
                foidev_titles = field_extract(title, foidev_field_numbers)
                device_MDR_Hash = {'mdr_key': foidev_titles}

                # Write the titles
                with open(device_name + '_foidev.txt', "w") as myfile:
                    myfile.write('|'.join('%s' % id for id in
                     foidev_titles) + '\n')
                myfile.close()

        foidev_file.close()

        # Extract only those FOIDEV records related to the Device
        # => Write in the Hash Table and in 'device_FOIDEV.txt'
        with open(filename + '.txt', "rb") as foidev_file:
            for line in foidev_file:
                for k in device_keywords:
                    if line.lower().find(k) > -1:
                        mdr_key = line.split('|')[0]
                        if not device_MDR_Hash.has_key(mdr_key):
                            # print mdr_key;
                            foidev_fields = field_extract(line,
                             foidev_field_numbers)
                            if foidev_fields is not None:  # ljn changed
                                device_MDR_Hash[mdr_key] = foidev_fields
                                foidev_count = foidev_count + 1
                                # print foidev_count
                                # Write FOIDEV Columns
                                with open(device_name + '_foidev.txt', "a")
                                 as myfile:
                                    myfile.write('|'.join('%s' % id for id
                                     in foidev_fields) + '\n')
                                myfile.close()
                                break
        foidev_file.close()
    # print (str(" ".join('%s' % id for id in foidev_count)+' FOIDEV
     records extracted and added to the table.'))
    # print (str(" ".join('%d' % id for id in range(1, foidev_count+1)) + '
     FOIDEV records extracted and added to the
    # table.'))
    print (str(" ".join('%d' % foidev_count) + ' foidev records extracted
     and added to the table.'))

    return device_MDR_Hash


### Extract the FOIDEV records and append them to the file
def FOIDEVExtract2(FOIDEV_files, FOIDEV_Field_Numbers, device_name,
 device_codes, data_dir):
    foidev_count = 0
    os.chdir(data_dir)

    # Extract those related to the device
    for filename in FOIDEV_files:
        # If the first file, first get the titles
        if (filename == FOIDEV_files[0]):
            with open(data_dir + filename + '.txt', "rb") as foidev_file:
                title = foidev_file.next()

                # Create the Hash Table of FOIDEV records
                FOIDEV_Titles = field_extract(title, FOIDEV_Field_Numbers)
                device_MDR_Hash = {'MDR_Key': FOIDEV_Titles}

                # Write the titles
                with open(device_name + '_FOIDEV.txt', "w") as myfile:
                    myfile.write('|'.join(FOIDEV_Titles) + '\n')
                myfile.close()

        foidev_file.close()

        # Extract only those FOIDEV records related to the Device
        # => Write in the Hash Table and in 'device_FOIDEV.txt'
        with open(data_dir + filename + '.txt', "rb") as foidev_file:
            for line in foidev_file:
                for k in device_codes:
                    if (line.find(k) > -1):
                        MDR_Key = line.split('|')[0]
                        if (not device_MDR_Hash.has_key(MDR_Key)):
                            # print MDR_Key;
                            FOIDEV_Fields = field_extract(line,
                             FOIDEV_Field_Numbers)
                            device_MDR_Hash[MDR_Key] = FOIDEV_Fields
                            foidev_count = foidev_count + 1
                            # print foidev_count
                            # Write FOIDEV Columns
                            with open(device_name + '_FOIDEV.txt', "a") as
                             myfile:
                                myfile.write('|'.join(FOIDEV_Fields) + '\n')
                            myfile.close()
                            break
        foidev_file.close()

    print (str(foidev_count) + ' FOIDEV records extracted and added to the
     table.')
    return device_MDR_Hash


regex = re.compile(r'\s*[\n\r\t]')


def Get_Other_Fields(MDR_Link):
    # Open each MDR Link
    # time.sleep(0.25)

    # result = urllib2.urlopen(MDR_Link)
    r = requests.get(MDR_Link)
    r.raise_for_status()  # 有错误就直接抛出
    # soup = bs4.BeautifulSoup(r.text, features='html.parser')
    soup = bs4.BeautifulSoup(r.text, 'lxml')  # ljn changed, html.parser is
     slow

    ##### Patient Outcome, Event Description, and Manufacturer Narrative
    # regex = re.compile(r'\s*[\n\r\t]')
    Patient_Outcome = 'N/A'
    Event = ''
    Narrative = ''
    for st in soup.findAll('strong'):
        # Patient Outcome
        if (st.string.count('Patient Outcome') > 0):
            if (st.next.next != ''):
                Raw_Outcome = st.next.next
                Patient_Outcome = regex.sub('',
                 Raw_Outcome).strip().encode('ascii',
                 'ignore').replace("&nbsp;", "")

        # Event Description
        if (st.string.find('Event Description') > 0):
            if (st.findNext('p').contents != []):
                Raw_Event = st.findNext('p').contents[0]
                Event = Event + regex.sub('',
                 Raw_Event).strip().encode('ascii', 'ignore') + ' '

        # Manufacturer Narrative
        if (st.string.find('Manufacturer Narrative') > 0):
            if (st.findNext('p').contents != []):
                Raw_Narrative = st.findNext('p').contents[0]
                Narrative = Narrative + regex.sub('',
                 Raw_Narrative).strip().encode('ascii', 'ignore') + ' '
    # If not found any narrative or event description
    if (Event == ''):
        Event = 'N/A'
    if (Narrative == ''):
        Narrative = 'N/A'

        ##### Number of Devices
    for st in soup.findAll('th'):
        if (len(st.contents) > 1):
            if ((st.contents[1].string.strip().encode('ascii',
             'ignore').count(
                    'Device Was Involved in the Event') > 0) or
                    (st.contents[1].string.strip().encode('ascii',
                     'ignore').count(
                        'DeviceS WERE Involved in the Event') > 0)):
                # Number_Devices =
                 st.contents[0].contents
                 [0].string.strip().encode('ascii','ignore') # ljn changed
                break
    return [Patient_Outcome, Event, Narrative]


def MAUDE_Merge_Tables(end_year, foidev_files, mdrfoi_files,
 foidev_field_numbers,
                       mdrfoi_field_numbers, device_name, data_dir):  # ljn
                        changed
    os.chdir('./')
    MAUDE_Keys = []
    AllCounts = [0, 0, 0]

    # Optimized MAUDE Data Output
    # newbook = xlwt.Workbook("iso-8859-2")
    #  newbook = xlsxwriter.Workbook("iso-8859-2")
    newbook = xlsxwriter.Workbook('full.xlsx')
    # newsheet = newbook.add_sheet('Maude_Data', cell_overwrite_ok = True)
    newsheet = newbook.add_worksheet('Maude_Data')  # ljn added

    f1 = open('./' + device_name + '_MAUDE_Data_' + str(end_year) +
     'plus_plus.csv', 'wb')
    csv_wr = csv.writer(f1, dialect='excel', delimiter=',')

    # Extract the Titles of Fields of Interest
    # FOIDEV_Titles
    with open(data_dir + foidev_files[0] + '.txt', "rb") as foidev_file:
        # FOIDEV_Field_Numbers = get_numbers(FOIDEV_files[0]) # LJN CHANGED
        title = foidev_file.next()
        FOIDEV_titles = field_extract(title, foidev_field_numbers)

    # MDRFOI_titles
    with open(data_dir + mdrfoi_files[0] + '.txt', "rb") as mdrfoi_file:
        # MDRFOI_Field_Numbers = get_numbers(MDRFOI_files[0])
        title = mdrfoi_file.next()
        MDRFOI_titles = field_extract(title, mdrfoi_field_numbers)

    # Create device_MDR_Hash
    device_MDR_Hash = {'MDR_Key': FOIDEV_titles}
    with open(device_name + '_foidev.txt', "r") as foidev_file:
        # Skip the title
        title = foidev_file.next()
        for line in foidev_file:
            FOIDEV_Fields = line.split('|')
            MDR_Key = FOIDEV_Fields[0].strip()
            device_MDR_Hash[MDR_Key] = FOIDEV_Fields
            # print FOIDEV_Fields;
    print('Number of records = ' + str(len(device_MDR_Hash)) + '\n')

    # Cross-match MDRFOI files to FOIDEV file
    curr_row = 0
    for filename in mdrfoi_files:
        with open(data_dir + filename + '.txt', 'rb') as mdrfoi_file:
            # Skip the title
            title = mdrfoi_file.next()

            # If first time, write the titles
            if filename == mdrfoi_files[0]:
                newsheet.write(curr_row, 0, 'MDR_Link')
                newsheet.write(curr_row, 1, 'Patient_Outcome')
                newsheet.write(curr_row, 2, 'Event')
                newsheet.write(curr_row, 3, 'Narrative')
                newsheet.write(curr_row, 4, 'Manufacture Year')
                newsheet.write(curr_row, 5, 'Event Year')
                newsheet.write(curr_row, 6, 'Report Year')
                newsheet.write(curr_row, 7, 'Time to Event')
                newsheet.write(curr_row, 8, 'Time to Report')
                curr_col = 9
                # Write MDRFOI Titles
                for i in range(0, len(MDRFOI_titles)):
```

```
            newsheet.write(curr_row, curr_col + i,
             MDRFOI_titles[i])  #
             xlwt不支持超过256行的写入们需要采用xlsxwriter
        # Write FOIDEBV Titles
        for i in range(0, len(FOIDEV_titles)):
            newsheet.write(curr_row, curr_col + len(MDRFOI_titles)
             + i, FOIDEV_titles[i])
            # Goto the next row
        curr_col = 0
        curr_row = 1

        csv_wr.writerow(['MDR_Link', 'MDR_Key', 'Event',
         'Narrative', 'Event_Type', 'Patient_Outcome',
                        'Manufacture Year', 'Event_Year',
                         'Report_to_Manufacture_Year',
                         'Report_to_FDA',
                        'Report_Year', 'Time_to_Event',
                         'Time_to_Report',
                        'Manufacturer', 'Brand_Name',
                         'Generic_Name', 'Product_Code'])

    # For each file, read Each Line and Cross-Match it to FOIDEV
    for k, line in enumerate(mdrfoi_file):
        # st = time.time()

        MDRFOI_fields = field_extract(line, mdrfoi_field_numbers)
        if (MDRFOI_fields == None):
            continue
        MDR_Key = MDRFOI_fields[0]
        # print(MDRFOI_titles.index('EVENT_TYPE')) # 21, ljn changed
        Event_Type =
         MDRFOI_fields[MDRFOI_titles.index('EVENT_TYPE')]
        if MAUDE_Keys.count(MDR_Key) == 0:
            MAUDE_Keys.append(MDR_Key)
            AllCounts[0] = AllCounts[0] + 1
            if Event_Type == 'D':
                AllCounts[1] = AllCounts[1] + 1
            elif Event_Type == 'IN':
                AllCounts[2] = AllCounts[2] + 1

        if (device_MDR_Hash.has_key(MDR_Key)):  # or (MDR_Key ==
         '2222833'):
            # Get the report year
            if (MDRFOI_fields[MDRFOI_titles.index('DATE_RECEIVED')]
             != ''):
                Report_DateStr =
                 MDRFOI_fields[MDRFOI_titles.index('DATE_RECEIVED')]
                Report_Date = parser.parse(Report_DateStr)
                Report_Year = str(Report_Date.year)
            else:
                Report_Date = 'N/A'
                Report_Year = 'N/A'

            # Only if the report year is before the end year
```

```
if (int(Report_Year) <= end_year):
    # Get the rest of the fields from online records
    MDR_Link =
     'http://www.accessdata.fda
     .gov/scripts/cdrh/cfdocs/cfMAUDE/Detail
     .cfm?MDRFOI__ID=' + MDR_Key
    print (i, str(curr_row) + '=' + MDR_Key + '\n')
    try:
        s = time.time()
        [Patient_Outcome, Event, Narrative] =
         Get_Other_Fields(MDR_Link)
        e = time.time()
        print(1, e - s, MDR_Key)
        # MDR_HLink = 'HYPERLINK("' + MDR_Link + '";"'
         + MDR_Link + '")'
        MDR_HLink = 'HYPERLINK("' + MDR_Link + '";"' +
         MDR_Link + '")'

        # Correct the EVENT Type
        Event_Type =
         MDRFOI_fields
         [MDRFOI_titles.index('EVENT_TYPE')]
        if (Event_Type == '*') or (Event_Type == ''):
            Event_Type = 'O'

        # Extract all the time fields
        if
         (MDRFOI_fields
         [MDRFOI_titles
         .index('DEVICE_DATE_OF_MANUFACTURE')] != ''):
            Manufacture_DateStr = MDRFOI_fields[
                MDRFOI_titles
                 .index('DEVICE_DATE_OF_MANUFACTURE')]
                 .strip()
            Manufacture_Date =
             parser.parse(Manufacture_DateStr)
            Manufacture_Year =
             str(Manufacture_Date.year)
        else:
            Manufacture_Date = 'N/A'
            Manufacture_Year = 'N/A'

        if
         (MDRFOI_fields
         [MDRFOI_titles.index('DATE_OF_EVENT')] != ''):
            Event_DateStr =
             MDRFOI_fields
             [MDRFOI_titles.index('DATE_OF_EVENT')]
            Event_Date = parser.parse(Event_DateStr)
            Event_Year = str(Event_Date.year)
        else:
            Event_Date = 'N/A'
            Event_Year = 'N/A'
```

```
if
 (MDRFOI_fields
 [MDRFOI_titles.index('DATE_REPORT')] != ''):
    ReportMade_DateStr =
     MDRFOI_fields
     [MDRFOI_titles.index('DATE_REPORT')]
    ReportMade_Date =
     parser.parse(ReportMade_DateStr)
    ReportMade_Year = str(ReportMade_Date.year)
else:
    ReportMade_Date = 'N/A'
    ReportMade_Year = 'N/A'

if
 (MDRFOI_fields
 [MDRFOI_titles
 .index('DATE_REPORT_TO_MANUFACTURER')] != ''):
    ReportMan_DateStr =
     MDRFOI_fields
     [MDRFOI_titles
     .index('DATE_REPORT_TO_MANUFACTURER')]
    ReportMan_Date =
     parser.parse(ReportMan_DateStr)
    ReportMan_Year = str(ReportMan_Date.year)
else:
    ReportMan_Date = 'N/A'
    ReportMan_Year = 'N/A'

if Manufacture_Date != 'N/A' and Event_Date !=
 'N/A' and Event_Date > Manufacture_Date:
    Time_to_Event = str((Event_Date -
     Manufacture_Date).days)
else:
    Time_to_Event = 'N/A'
if Event_Date != 'N/A' and Report_Date != 'N/A'
 and Report_Date > Event_Date:
    Time_to_Report = str((Report_Date -
     Event_Date).days)
else:
    Time_to_Report = 'N/A'

    # Write the extracted of MDRFOI Columns
     from online records

# newsheet.write(curr_row, 0,
 xlwt.Formula(MDR_HLink))
newsheet.write(curr_row, 0, MDR_Link)
newsheet.write(curr_row, 1, Patient_Outcome)
newsheet.write(curr_row, 2, Event)
newsheet.write(curr_row, 3, Narrative)
newsheet.write(curr_row, 4, Manufacture_Year)
newsheet.write(curr_row, 5, Event_Year)
newsheet.write(curr_row, 6, Report_Year)
newsheet.write(curr_row, 7, Time_to_Event)
newsheet.write(curr_row, 8, Time_to_Report)
```

```
                        curr_col = 9

                        # Write the rest of MDRFOI Columns
                        for i in range(0, len(MDRFOI_titles)):
                            if MDRFOI_titles[i].find('EVENT_TYPE') > -1:
                                newsheet.write(curr_row, curr_col + i,
                                 Event_Type)
                            else:
                                newsheet.write(curr_row, curr_col + i,
                                 MDRFOI_fields[i])
                        # Write FOIDEV Columns
                        for i in range(0, len(FOIDEV_titles)):
                            newsheet.write(curr_row, curr_col +
                             len(MDRFOI_titles) + i,
                             device_MDR_Hash[MDR_Key][i])

                            # Write selected columns to CSV file
                        Manufacturer = device_MDR_Hash[MDR_Key][4]
                        Brand_Name = device_MDR_Hash[MDR_Key][2]
                        Generic_Name = device_MDR_Hash[MDR_Key][3]
                        Product_Code = device_MDR_Hash[MDR_Key][6]
                        # print (Manufacturer)
                        # print (Brand_Name)
                        # print (Generic_Name)
                        # print (Product_Code)
                        csv_wr.writerow([MDR_Link, MDR_Key, Event,
                         Narrative, Event_Type, Patient_Outcome,
                                        Manufacture_Year, Event_Year,
                                         ReportMan_Year,
                                         ReportMade_Year, Report_Year,
                                        Time_to_Event, Time_to_Report,
                                        Manufacturer, Brand_Name,
                                         Generic_Name, Product_Code])

                        # Remove the record from the hash to avoid
                         duplicate records
                        device_MDR_Hash.pop(MDR_Key)

                        # Goto the next row
                        et2 = time.time()
                        print(2, et2 - e, MDR_Key)
                        print(filename, 'Extracted line %d' % k)
                        curr_row = curr_row + 1

                    except Exception, e:
                        print('Exception error in %s, ignore it!' %
                         MDR_Key)

        mdrfoi_file.close()

    print(str(curr_row) + ' MDRFOI records cross-matched with FOIDEV
     records, and saved to the XLS file.')
    #
     newbook.save(data_dir+device_name+'_MAUDE_Data_'+str(end_year)+'plus
     .xls')
    newbook.close()
    return AllCounts


def get_numbers(foi):
    '''
    每个文件的需要采集的numbers不一样，所以采用动态生成即可
    :param foi: foidev or mdrfoi filename
    :return:
    '''
    reader = open(data_dir + foi + '.txt', "rb")
    reader.next()
    line = reader.next()
    segs = line.split('|')
    num_list = []
    for i, s in enumerate(segs):
        if (s != ""):
            num_list.append(i + 1)

    reader.close()
    print(num_list)
    return num_list


# def cal_numbers(files):

###### Parameters
# Fields of interest (Numbers are based on Field Numbers provided on the
 FDA Website)
# FOIDEV_Field_Numbers = [1,2,7,8,9,19,26,27,29,30,31,34,36,44]
FOIDEV_Field_Numbers = [1, 2, 7, 8, 9, 19, 26, 27]  #
 自2009年之后的数据，只有28个字段，09年之前的数据有45个字段
# MDRFOI_Field_Numbers =
 [1,2,3,4,6,7,8,9,10,11,12,14,16,17,25,26,27,28,30,31,32,68,69,70,72,75]
MDRFOI_Field_Numbers = [1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 14, 16, 17, 22,
 25, 26, 27, 28, 30, 31, 32, 51, 56, 68, 69,
                        70, 72, 75]
# Years
start_year = 2000  # 2015 #2000
current_year = 2020  # 2014
end_year = 2019  # 2008 # 2013
# Device of Interest
device_name = ['daVinci', 'pacemaker', 'patient_monitor']
# Device Keywords
device_keywords = [['da vinci', 'davinci', 'davency', 'davincy', 'davincy',
                    'intuitive surgical', 'intuitivesurgical'],
                    ['pacemaker']]
# Data Directory
data_dir = './datas/'

####### Generate the FOIDEV Filenames
foidev_files = []
for years in range(start_year, current_year):
    foidev_files.append('foidev' + str(years))
```

```
foidev_files = foidev_files + ['foidevadd', 'foidevchange', 'foidev']  #
 ljn changed
####### Generate the MDRFOI Filenames
# mdrfoithru2018.zip
 解压后是mdrfoiThru2018.zip，foidevChnage.zip解压之后是foidevchange.txt,需要注意
MDRFOI_files = ['mdrfoithru' + str(current_year - 1), 'mdrfoi',
 'mdrfoichange']

####### Download Maude Data
# MAUDE_Download(FOIDEV_files, MDRFOI_files, data_dir)

# MAUDE_Download(['foidev2014'], [], data_dir)

####### Extract FOIDEV files for the device of interest
foidev_extract(foidev_files, FOIDEV_Field_Numbers, device_name[0],
 device_keywords[0], data_dir)

# foidev_extract(FOIDEV_files, device_name[0], device_keywords[0], data_dir)


####### Cross-match the MDRFOI and FOIDEV records
# AllCounts = MAUDE_Merge_Tables(end_year, FOIDEV_files, MDRFOI_files,
 FOIDEV_Field_Numbers,
#                                 MDRFOI_Field_Numbers, device_name[0],
 data_dir)

print('\n')
print('Check the reports that are not from intuitive to make sure they are
 related to da Vinci')
print('The report 2222833 is manually added in order to compare with
 cardiac surgery records')
```