```
#This program converts MIT QDC metadata records into the developed Common Terminology
#using the developed XML schema.
#developed by (Boaz) Sunyoung Jin, supervised by Professor Dave Dubin, and supported by
Dean Smith
#It is modified to measure semantic match rates in July 2014 from the version of March, 2014
#URIs are changed into http://www.ct.iopdl.org/1.1/ from
http://courseweb.lis.illinois.edu/~sunjin/CT/1.1/ on December 09, 2014.
import csv
import string
elementDic={} #dictionary that key is a QDC element name and key value is CT.
specialDic={'&':'&amp;','<':'&lt;','>':'&gt'}
specialkeys=specialDic.keys()
def makeDic():
    filename="MITQDCtoCTcrosswalkVI.csv" #set up each file name of MIT files
    fhandle=open(filename,"rb")#open each file
    fcontent=csv.reader(fhandle)#read each file

    nrow=0
    for row in fcontent:
        if nrow<3:
            nrow+=1
            continue
        elif "ignore" in row[3]:
            nrow+=1
            continue
        else:
            key=row[0]
            elementDic[key]=row[3]
            nrow+=1
            #print key, elementDic[key]
    return elementDic

def MITQDCtoCTconversion(elementDic):
    MITrecordDic={}
    tmatch=0
    notmatch=0
    pmatch=0
    hpmatch=0
    nopmatch=0
    smatch=0
    hsmatch=0
    nosmatch=0
    totalMITrecords=0 #count how many records MIT has
    for iden in range(1,29):#repeat reading and making a dictionary for MIT 28 files
        if iden<10:
            iden="0"+str(iden)
```

```
        filename="MIT"+str(iden)+".xml" #set up each file name of MIT files
        fhandle=open(filename,'r')#open each file
        fcontent=fhandle.read()#read each file
        fhandle.close()#close the file
        records=fcontent.split('<record>') #find records
        totalMITrecords=totalMITrecords+(len(records)-1)#accumulate total number of records of
each file
        print
        print 'filename:', filename, len(records)-1,'totalMITrecords:', totalMITrecords
        print
        filename2="CTMITuri"+str(iden)+".xml"#set up output file name
        fhandle2=open(filename2,'w')#open the output file
        value='<?xml version="1.0" encoding="UTF-8"?>\n'
        fhandle2.write(value)
        value='<!--filename:'+ str(filename)+'  records:'+str(len(records)-1)+'
totalMITrecords:'+str(totalMITrecords)+'-->\n'
        fhandle2.write(value)
        value='<CT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"'
        fhandle2.write(value)
        value='\txsi:schemaLocation="http://www.ct.iopdl.org/1.1/ http://www.ct.iopdl.org/1.1/ct1-
1.xsd"\n'
        fhandle2.write(value)
        value='\txmlns="http://www.ct.iopdl.org/1.1/">\n'
        fhandle2.write(value)
        fhandle2.write('\n')

        for i in range(1, len(records)):#repeat for each record of MIT QDC records
            MITrecordDic={}
            elements=records[i].split('</')#split by '</' to retrieve element name and value
            #print 'elements:', elements
            for j in range(len(elements)):
                presentelements=elements[j].split('>')
                #print 'presentelements:', presentelements
                present=len(presentelements)-1
                #print 'present:', present
                if present==1:
                    continue
                content=presentelements[present]#retrieve value
                for skey in specialkeys:
                    if skey in content:#replace special characters to validate in xml file based on CT
schema
                        content=content.replace(skey, specialDic[skey])
                #print 'content:', content
                elementsname=elements[j+1].split('>')[0]#retrieve element name in closing tag,
because starting tag has many urls.
                #print 'elementsname:', elementsname
```

```
            if elementsname in elementDic:#if the elementsname is in elementDic.
                CT=elementDic[elementsname]
                key=CT
                #CT=CT.lower()
                pmatch, hpmatch, nopmatch,smatch, hsmatch,
nosmatch=LexicalSemanticMatch(elementsname,CT,pmatch, hpmatch, nopmatch, smatch,
hsmatch,nosmatch)
                ##call the function that measures lexical and semantic match rates
                ##This part generates dictionary and investigates transfer rate.
                if key in MITrecordDic:
                    if content!=MITrecordDic[key]:
                        MITrecordDic[key]=content
                        keyelement=key.split(' ')[0]
                        #print key, CTMITDic[key]

element='\t<'+str(key)+'>'+str(MITrecordDic[key])+'</'+str(keyelement)+'>'+'\n'
                        fhandle2.write(element)
                        tmatch+=1#measuring transferred elements
                else:
                    MITrecordDic[key]=content
                    #print '5',MITrecordDic[key]
                    keyelement=key.split(' ')[0]
                    element='\t<'+str(key)+'>'+str(MITrecordDic[key])+'</'+str(keyelement)+'>'+'\n'
                    #print element
                    fhandle2.write(element)
                    tmatch+=1
            else:#if the elementsname is not in elementDic.
                notmatch+=1##measuring none transferred elements
                if elementsname!="resumptionToken":
                    print 'notmatch', elementsname

        fhandle2.write('\n')
    value='</CT>'
    fhandle2.write(value)
    fhandle2.close()
  return tmatch, notmatch, totalMITrecords, pmatch, hpmatch, nopmatch,smatch,
hsmatch,nosmatch

def LexicalSemanticMatch(elementsname,CT,pmatch, hpmatch, nopmatch,
smatch,hsmatch,nosmatch):
  ##This part investigates lexical match rates.
  count=0
  semanticCheck=0
  halflexical=['datestamp','dateAccepted']
  perfectlexical=['uri','ispartofseries','isreplacedby','requires', 'isPartOf']
```

```
   semanticDic={'audience.educationlevel': 'description
type="audience"','contributor.department':'contributor name="corporate"',\
            'contributor.mitauthor':'contributor role="author"
authority="LCMARCRelators"','coverage.spatial':'subject type="spatial"',\
            'date.accessioned':'date type="available"','date.created':'date
type="issued"','date.updated':'date type="modified"',\
            'description.statementofresponsibility':'rights
type="holder"','eprint.version':'description type="edition"','dcterms:created':'date type="issued"',\
            'format.mimetype':'format type="medium"
authority="rfc2046"','relation.ispartofseries':'relation type="isPartOf"',\
            'relation.isreplacedby':'relation type="replacement"','relation.isversionof':'relation
type="original"','relation.journal':'relation type="isPartOf"',\
            'relation.requires':'relation type="requirement"','source':'relation
type="original"','datestamp':'date type="other"',\
            'dateAccepted':'date type="available"','isPartOf':'relation type="isPartOf"',
'setSpec':'identifier type="collection"'}

   if '[' in elementsname:
      elementsname=elementsname.split('[')[0]
   if '.' in elementsname or ':' in elementsname:
      if '.' in elementsname:
         con='.'
      else:
         con=':'
      names=elementsname.split(con)
      length=len(names)
      if 'dc' in names[0] or 'dcterms' in names[0]:
         start=1
      else:
         start=0
      for k in range(start, len(names)):
         if names[k] in perfectlexical and names[k] in semanticDic.keys():
            pmatch+=1
            smatch+=1
            if names[k] != "isPartOf":
               print 'pmatch and smatch 1', elementsname, CT, pmatch, smatch
            return pmatch, hpmatch, nopmatch,smatch,hsmatch,nosmatch
         elif names[k] in perfectlexical:
            pmatch+=1
            smatch+=1
            #print 'pmatch1 and smatch1', elementsname, CT, pmatch, smatch
            return pmatch, hpmatch, nopmatch,smatch,hsmatch,nosmatch
         elif names[k] in halflexical and names[k] in semanticDic.keys():
            hpmatch+=1
            smatch+=1
            if names[k] != "dateAccepted":
```

```
                print 'hmatch2 and smatch2', elementsname, CT, hpmatch, smatch
                return pmatch, hpmatch, nopmatch,smatch,hsmatch, nosmatch
            elif names[k] in CT or names[k] in CT.lower():
                count+=1

        if count==length-1:
            pmatch+=1
            smatch+=1
            #print 'pmatch3 and smatch3', elementsname, CT,pmatch, smatch
        elif length>2 and count==length-2:
            hpmatch+=1
            if 'contributor.department' not in elementsname and 'date.accessioned' not in
elementsname and\
                'coverage.spatial' not in elementsname and 'advisor' not in elementsname and 'degree'
not in elementsname and\
                'mimetype' not in elementsname and 'created' not in elementsname and 'sponsorship' not
in elementsname\
                and 'audience' not in elementsname and 'approver' not in elementsname and 'mitauthor'
not in elementsname\
                and 'submitted' not in elementsname and 'updated' not in elementsname and 'citation'
not in elementsname\
                and 'govdoc' not in elementsname and 'mitlicense' not in elementsname and 'pmid' not
in elementsname\
                and 'isversionof' not in elementsname and 'journal' not in elementsname:
                print 'hpmatch ', elementsname, CT, 'hpmatch=', hpmatch
            smatch, hsmatch, nosmatch=semanticMatch (elementsname, CT, smatch,
hsmatch,nosmatch)
        else:
            nopmatch+=1
            if 'description.statementofresponsibility' not in elementsname and 'version' not in
elementsname and \
                'source' not in elementsname and 'dcterms:created' not in elementsname:
                print 'nopmatch', elementsname, CT, 'nopmatch=', nopmatch
            smatch, hsmatch, nosmatch=semanticMatch (elementsname, CT, smatch,
hsmatch,nosmatch)

    elif elementsname in CT:
        pmatch+=1
        smatch+=1
        if elementsname!='identifier' and elementsname!='id' and elementsname!='collection' :
            print 'pmatch and smatch4', elementsname, CT,pmatch, smatch
    elif elementsname in halflexical:
        hpmatch+=1
        smatch, hsmatch, nosmatch=semanticMatch (elementsname, CT, smatch,hsmatch, nosmatch)
        if elementsname!= 'datestamp':
```

```
            print 'hpmatch and smatch 4', elementsname, CT, 'hpmatch=', hpmatch, 'smatch=',smatch,
'nosmatch=', nosmatch
    else:
        nopmatch+=1
        smatch, hsmatch, nosmatch=semanticMatch (elementsname, CT, smatch, hsmatch,nosmatch)
        if elementsname !='setSpec':
            print 'nopmatch and smatch 4', elementsname, CT, 'nopmatch=', nopmatch,
'smatch=',smatch, 'nosmatch=', nosmatch

    return pmatch, hpmatch, nopmatch,smatch,hsmatch,nosmatch

def semanticMatch (elementsname, CT, smatch, hsmatch, nosmatch):
    semanticDic2={'dc.audience.educationlevel': 'description
type="audience"','dc.contributor.department':'contributor name="corporate"',\
                'dc.contributor.mitauthor':'contributor role="author"
authority="LCMARCRelators"','dc.coverage.spatial':'subject type="spatial"',\
                'dc.date.accessioned':'date type="available"','dc.date.created':'date
type="issued"','dc.date.updated':'date type="modified"',\
                'dc.description.statementofresponsibility':'rights
type="holder"','dc.eprint.version':'description type="edition"','dcterms:created':'date
type="issued"',\
                'dc.format.mimetype':'format type="medium"
authority="rfc2046"','dc.relation.ispartofseries':'relation type="isPartOf"',\
                'dc.relation.isreplacedby':'relation
type="replacement"','dc.relation.isversionof':'relation
type="original"','dc.relation.journal':'relation type="isPartOf"',\
                'dc.relation.requires':'relation type="requirement"','dc.source':'relation
type="original"','datestamp':'date type="other"',\
                'dcterms:dateAccepted':'date type="available"','dcterms:isPartOf':'relation
type="isPartOf"', 'setSpec':'identifier type="collection"'}

    if elementsname in semanticDic2.keys():
        smatch+=1
        if elementsname!= 'datestamp' and elementsname!='setSpec' and 'department' not in
elementsname and 'spatial' not in elementsname\
            and 'accessioned' not in elementsname and 'statementofresponsibility' not in
elementsname and 'mimetype' not in elementsname\
            and 'created' not in elementsname and 'audience' not in elementsname and 'mitauthor' not
in elementsname and 'updated' not in elementsname\
            and 'version' not in elementsname and 'journal' not in elementsname and 'source' not in
elementsname:
            print 'smatch=',smatch, elementsname
        return smatch, hsmatch, nosmatch
    element=elementsname.split('dc')[1]
    element=element.split('.')
    for i in range (len(element)):
```

```
        if element[i] in CT:
            hsmatch+=1
            if 'advisor' not in elementsname and 'degree' not in elementsname and 'sponsorship' not in
elementsname and 'approver' not in elementsname\
                and 'submitted' not in elementsname and 'citation' not in elementsname and 'govdoc'not
in elementsname and 'mitlicense' not in elementsname\
                and 'pmid' not in elementsname:
                print 'hsmatch=',hsmatch, elementsname, CT
            return smatch, hsmatch, nosmatch

    nosmatch+=1
    print 'nosmatch=',nosmatch, elementsname
    return smatch, hsmatch, nosmatch

def MITQDCcsvtoCTconversion(tmatch, notmatch, total, elementDic, pmatch, hpmatch,
nopmatch,smatch, hsmatch, nosmatch):
    for iden in range(29,32):
        print tmatch, notmatch, total
        filename="MIT"+str(iden)+".csv" #set up each file name of MIT files
        fhandle=open(filename,"rb")#open each file
        fcontent=csv.reader(fhandle)#read each file
        print filename
        filename2="CTMITuri"+str(iden)+".xml"#set up output file name
        fhandle2=open(filename2,'w')#open the output file
        value='<?xml version="1.0" encoding="UTF-8"?>\n'
        fhandle2.write(value)
        value='<!--filename:'+ str(filename)+'-->\n'
        fhandle2.write(value)
        value='<CT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"\n'
        fhandle2.write(value)
        value='\txsi:schemaLocation="http://www.ct.iopdl.org/1.1/ http://www.ct.iopdl.org/1.1/ct1-
1.xsd"\n'
        fhandle2.write(value)
        value='\txmlns="http://www.ct.iopdl.org/1.1/">\n'
        fhandle2.write(value)
        fhandle2.write('\n')
        record=0
        nrow=0
        for row in fcontent:
            if nrow==0:
                elements=row
                nrow+=1
                #print elements
            else:
                MITrecordDic={}
                ncol=0
```

```
            record+=1

            for col in row:
               elementsname=elements[ncol]
               if col!='' and elementsname!='' and  elementsname in elementDic:
                  CT=elementDic[elementsname]
                  key=CT
                  #elementsname=elementsname.lower()
                  #CT=CT.lower()
                  pmatch, hpmatch, nopmatch,smatch,hsmatch,
nosmatch=LexicalSemanticMatch(elementsname,CT,pmatch, hpmatch, nopmatch,
smatch,hsmatch, nosmatch)
                  ##This part generates dictionary and investigates transfer rate.
                  for skey in specialkeys:
                     if skey in col:#replace special characters to validate in xml file based on CT
schema
                        col=col.replace(skey, specialDic[skey])
                  if key in MITrecordDic:
                     if col!=MITrecordDic[key]:
                        MITrecordDic[key]=col
                        keyelement=key.split(' ')[0]
                        #print key, CTMITDic[key]

element='\t<'+str(key)+'>'+str(MITrecordDic[key])+'</'+str(keyelement)+'>'+'\n'
                        fhandle2.write(element)
                        tmatch+=1
                  else:
                     MITrecordDic[key]=col
                     #print '5',MITrecordDic[key]
                     keyelement=key.split(' ')[0]

element='\t<'+str(key)+'>'+str(MITrecordDic[key])+'</'+str(keyelement)+'>'+'\n'
                     #print element
                     fhandle2.write(element)
                     tmatch+=1
                  ncol+=1
               elif col!='' and col!=' 'and elementsname!='    ' and elementsname!=' 'and
elementsname!='' and elementsname not in elementDic:
                  notmatch+=1
                  ncol+=1
                  if 'eprint.grantNumber' not in elementsname:
                     print 'nopmatch', elementsname

            nrow+=1
            fhandle2.write('\n')
         value='</CT>'
```

```
        fhandle2.write(value)
        total=total+record
        print '<!--filename:', filename,'  records:',record,'  totalMITrecords:',total,'-->\n'
        value='<!--filename:'+ str(filename)+'  records:'+str(record)+'
totalMITrecords:'+str(total)+'-->\n'
        fhandle2.write(value)
        fhandle2.close()
        fhandle.close()#close the file
    return tmatch, notmatch, total, pmatch, hpmatch, nopmatch, smatch, hsmatch,nosmatch

elementDic=makeDic()
transfer, notransfer, total, pmatch, hpmatch, nopmatch,smatch, hsmatch, nosmatch=
MITQDCtoCTconversion(elementDic)
transfer, notransfer, total, pmatch, hpmatch, nopmatch,smatch, hsmatch,nosmatch=
MITQDCcsvtoCTconversion(transfer, notransfer, total, elementDic, pmatch, hpmatch,
nopmatch,smatch, hsmatch, nosmatch)
print "total records of MIT:", total
totaltransfer=transfer+notransfer
print "total transferred and non transferred ", totaltransfer
print "transfer:", transfer,"transfer rate:", transfer/float(totaltransfer)*100
print "notransfer:", notransfer,"notransfer rate:", notransfer/float(totaltransfer)*100

totallexical=pmatch+hpmatch+nopmatch
print "total lexical matched or not:", totallexical
print "perpect match:", pmatch,"perpect match rate:", pmatch/float(totallexical)*100
print "half match:", hpmatch, "half match rate:", hpmatch/float(totallexical)*100
print "no match:", nopmatch,"no match rate:", nopmatch/float(totallexical)*100

totalsemantic=smatch+hsmatch+nosmatch
print "total semantic matched or not:", totalsemantic
print "semantic match:", smatch,"semantic match rate:", smatch/float(totalsemantic)*100
print "hsemantic match:", hsmatch,"semantic match rate:", hsmatch/float(totalsemantic)*100
print "nosemantic match:", nosmatch,"nosemantic match rate:",
nosmatch/float(totalsemantic)*100
```