WP2024-29

Python spell checker for free text in EXFOR

(N. Otsuka, 2024-04-22, CP-D/1107)

I wrote a simple spell checker for free text in EXFOR entries for review of preliminary tapes. This script checks each set of characters consisting of lower cases. Compilers can run it by (1) copy & paste the Python script x4_spells.py (174 lines) from the appendix of this memo, and (2) create x4_spells.dic (collection of words not in a standard English dictionaries). It also requires installation of the Python module "pyspellchecker":

```
pip install pyspellchecker
```

Example: Spell check of PRELIM.D141:

```
X4_SPELLS (Ver-2024-04-22) run on 22-04-2024
--------------------------------------------
Name of EXFOR file [exfor.txt] --> prelim.d141

           isobutane gas of about 2.5 mbar pressure.              D642700100020
           ^^^^^^^^^                  ^^^^
TITLE      Examining the correlation between multi-neutron        D644400100003
                                             ^^^^^
FACILITY   (VDGT,3INDNSD) pelletron accelerator                   D644400100014
                          ^^^^^^^^^
           of 5 mbar of isobutane gas, was placed at the focal    D644400100017
                ^^^^    ^^^^^^^^^
…
                         counting statics, deadtime (8%)          D644800100024
                                           ^^^^^^^^
TITLE      Systematic study of prescission neutron multiplicity:  D645600100003
                               ^^^^^^^^^^^
           revealing the role of entrance channel magicity        D645600100004
                                                  ^^^^^^^^
DETECTOR   (HPGE) 100 cc detector, standard 152Eu reference       D645800100016
                      ^^
            - Statistical uncertainty in the photopeak counts     D645900100019
                                             ^^^^^^^^^

Detected typos:
  cc  x 2
  ck
  das
  deadtime
  hr
  isobutane  x 2
  linac
  magicity
  mbar  x 2
  multi
  pelletron  x 2
  photopeak
  prescission

Spell checking completed. Good bye!
```

The dictionary file used for this checking is

```
atm
cm
deg
min
mg
mm
multiwire
nsec
sec
yr
```

Appendix: x4_spells.py

(Note added to WP2024-30: This is updated from the version in CP-D/1107)

```
#!/usr/bin/python3
ver="2024-05-03"
######################################################
# X4_SPELLS Ver. 2024-05-03
# (Spell checker for free text in EXFOR)
#
# Naohiko Otuka (IAEA Nuclear Data Section)
######################################################
# Install pyspellchecker by the following command:
#  pip install pyspellchecker
######################################################
import datetime
import os
import re
import sys
import argparse

from spellchecker import SpellChecker
spell = SpellChecker()

def main():
  global known_words

  time=datetime.datetime.now()
  time=time.strftime("%Y-%m-%d %H:%M:%S")

  get_args(ver)
  (file_exfor,file_dict,file_log)=get_input()

  known_words=[]
  lines=get_file_lines(file_dict)

  f=open(file_log, 'w')
  f.write("           -----------------------------------------------------------------\n")
  f.write("           X4_SPELLS report "+time+" - unrecognized words in "+file_exfor+"\n")
  f.write("           -----------------------------------------------------------------\n")
  f.write("\n")

  for line in lines:
    array=re.split("(,|\s)", line)
    for item in array:
      known_words.append(item)

  lines=get_file_lines(file_exfor)
  typos=free_text_analyser(f,lines)

  num=dict()

  if (len(typos)>0):
    f.write("Unrecognized words:\n")
    typos.sort()
    typos_unique=list(dict.fromkeys(typos))
    for word in typos_unique:
      num[word]=typos.count(word)

    for typo in typos_unique:
      if (num[typo]==1):
        f.write(" "+typo+"\n")
      else:
        f.write(" "+typo+" x"+str(num[typo])+"\n")
    f.write("\n")

  f.close()

  lines=get_file_lines(file_log)
  for line in lines:
    print(line, end="")

  print("Spell checking completed. Good bye!")


def get_args(ver):
  global args

  parser=argparse.ArgumentParser(\
          usage="Check English spells in free text of EXFOR file",\
          epilog="example: x4_spells.py -i exfor.txt -d x4_spells.dic -o spells.log")
  parser.add_argument("-v", "--version",\
         action="version", version=ver)
  parser.add_argument("-f", "--force",\
   help="never prompt", action="store_true")
  parser.add_argument("-i", "--file_exfor",\
   help="name of input EXFOR file")
  parser.add_argument("-d", "--file_dict",\
   help="name of input known word dictionary file")
  parser.add_argument("-l", "--file_log",\
   help="name of output log file")

  args=parser.parse_args()


def get_input():
  time=datetime.datetime.now()
  date=time.strftime("%d-%m-%Y")
  print()
  print("X4_SPELLS (Ver-"+ver+") run on "+date)
  print("-------------------------------------------")

  file_exfor=args.file_exfor
  if file_exfor==None:
    file_exfor=input("Name of EXFOR file [exfor.txt] ---------------------> ")
    if file_exfor=="":
      file_exfor="exfor.txt"
  if not os.path.exists(file_exfor):
    print(" ** File "+file_exfor+" does not exist.")
  while not os.path.exists(file_exfor):
    file_exfor=input("Name of EXFOR file [exfor.txt] ---------------------> ")
    if file_exfor=="":
      file_exfor="exfor.txt"
    if not os.path.exists(file_exfor):
      print(" ** File "+file_exfor+" does not exist.")

  file_dict=args.file_dict
  if file_dict==None:
    file_dict=input("Name of known word dictionary file [x4_spells.dic] -> ")
    if file_dict=="":
      file_dict="x4_spells.dic"
  if not os.path.exists(file_dict):
    print(" ** File "+file_dict+" does not exist.")
  while not os.path.exists(file_dict):
    file_dict=input("Name of known word dictionary file [x4_spells.dic] --> ")
    if file_dict=="":
      file_dict="x4_spells.dic"
    if not os.path.exists(file_dict):
      print(" ** File "+file_dict+" does not exist.")

  file_log=args.file_log
  if file_log==None:
    file_log=input("output log file name [x4_spells.log] ---------------> ")
  if file_log=="":
    file_log="x4_spells.log"
  if os.path.isfile(file_log):
    msg="File '"+file_log+"' exists and must be overwritten."
    print_error(msg,"")

  print()
  return file_exfor, file_dict, file_log


def print_underline_multi(f,col1s,col2s):
  char=""
  ioff=0
  for i, col1 in enumerate(col1s):
    length=col2s[i]-col1
    char+=" "*(col1-ioff)+"^"*length
    ioff=col1+length
  f.write(char+"\n")


def get_file_lines(file):
  if os.path.exists(file):
    f=open(file, "r")
    lines=f.readlines()
    f.close()
  else:
    msg="File "+file+" does not exist."
    print_error_fatal(msg)
  return lines


# Extraction of free text
def free_text_analyser(f,lines):
  all_typos=[]
  sys_id="ENDENTRY"
  for line in lines:
    if re.compile("^BIB       ").search(line):
      sys_id="BIB"
      nparen=0
      code=""

    elif re.compile("^ENDBIB     ").search(line):
      sys_id="ENDBIB"

    elif sys_id=="BIB":
      content=line[11:66]
      (code,text,nparen)=code_extraction(content,code,nparen)
      if nparen==0:
        code=""
        all_typos=spell_checker(f,line,text,all_typos)

  return all_typos


def spell_checker(f,line,text,all_typos):
  words=re.split("(,|-|:|\s|\(|\))", text)
  typos=[]
  for i, word in enumerate(words):
    if re.compile("^[a-z]+$").search(word):
      result=spell.unknown([word])
      if len(result)==1 and word not in known_words:
        typos.append(word)
        all_typos.append(word)

  col1s=[]
  col2s=[]
  char=line
  for typo in typos:
    col1=char.find(typo)
    col2=col1+len(typo)
    col1s.append(col1)
    col2s.append(col2)
    length=len(typo)
    char=char.replace(typo,"X"*length,1)

  if len(typos)>0:
    line=line.rstrip("\n")
    f.write(line+"\n")
    print_underline_multi(f,col1s,col2s)

  return all_typos


def code_extraction(content,code,nparen):
  chars=list(content)
  nchar=0
  text=""
  for char in chars:
    if char=="(":
      nparen+=1
    elif char==")":
      nparen-=1
    if (nparen>0):
      nchar+=1
    if nparen==0:
      if nchar!=0:
        nchar+=1
      code+=content[0:nchar]
      text=content[nchar:]
      text=text.rstrip()
      return code, text, nparen
    else:
      text+=" "

  code+=content.rstrip()
  return code, "", nparen
```

```
def print_error_fatal(msg,line):
  print("** "+msg)
  print(line)
  exit()


def print_error(msg,line):
  print("** "+msg)
  print(line)

  if args.force:
    answer="Y"
  else:
    answer=""

  while answer!="Y" and answer!="N":
    answer=input("Continue? [Y] --> ")
    if answer=="":
      answer="Y"
    if answer!="Y" and answer!="N":
      print(" ** Answer must be Y (Yes) or N (No).")
  if answer=="N":
    print("program terminated")
    exit()


if __name__ == "__main__":
  main()
  exit()
```