Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

Script Python to transform bib file

Marco Righi, ISTI-CNR, Pisa, Italy

ISTI-TR-2022/042

Script Python to transform bib files
Righi M.
ISTI-TR-2022/042

This Technical Report describes a script written in Python to transform a bib into a proprietary format. Open Portal ISTI obtains the bib file, and the property format is defined in the following sections. The property SIlab uses the format in the annual report of the activity of the laboratory.

Keywords: Latex, Bib, Bibtex, Python.

Citation
Righi M. *Script Python to transform bib files*. ISTI Technical Reports 2022/042.
DOI: 10.32079/ISTI-TR-2022/042.

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

Script Python to transform bib files

Tecnichal Report

Marco Righi*

June 12, 2022

keyword: latex, bib, bibtex, Python

* *marco.righi@isti.cnr.it*, Istituto di Scienza e Tecnologia, Consiglio Nazionale delle Ricerche, 56124 Pisa, Italia

Contents

1 Abstract

This Technical Report describes a script written in Python [1] to transform a bib into a proprietary format.
Open Portal ISTI provides the bib file, and the property format is defined in the following sections. The property SIlab uses the format in the annual report of the activity of the laboratory [2].
***
This Technical Report describes a script written in Python [1] to transform a bib in a proprietary format.
The bib file is get by Open Portal ISTI and the property format is defined in the following sections. The property format is used by SIlab in the annual report of the activity of the laboratory [2].

2 Introduction

The SIlab annual report contains the main activities and the publications of the past year by the SIlab team. The number of publications and the necessary coherence between the bibliography and the short description of the papers has conduct the author to write a program to automatize the process. The chosen language is Python only for author's commodity, there was other good programming languages that could do this work well.

3 Proprietary Format

The proprietary format is a TeX string that is as follows.

3.1 Journal

- \textbf{Title}: \href{https://openportal.isti.cnr.it/results?qv="**Paper Title**"}{**Paper Title**}
- \textbf{Authors}: *the list of the authors*
- \textbf{Journal}: *the name of the journal*
- \textbf{Publisher}: *the name of the publisher*
- \textbf{DOI}: \doix{**DOI**}
- \textbf{Abstract}: \textit{Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.} \cite{**REF**}
- \\ \vspace{5mm} \\

3.2 Other publications (not journal)

- \textbf{Title}: \href{https://openportal.isti.cnr.it/results?qv="**Paper Title**"}{**Paper Title**}
- \textbf{Authors}: *the list of the authors*
- \textbf{DOI}: \doix{**DOI**}
- \textbf{Abstract}: \textit{Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus.

Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.} \cite{**REF**}

- \\ \vspace{5mm} \\

4 Code

```
#!/bin/python
import sys, os
import re
import urllib.request

class key_values:
    def __init__(self,citekey,values):
        self.citekey = citekey
        self.values = values


def get_data(datastring):
    attributevalue = []
    start = 0
    stop = datastring.find("@")
    if stop > 0:
        datastring=datastring[start:stop]
    else:
        datastring = datastring[start:]
    stop = datastring.rfind("}")
    datastring=datastring[start:stop] #key,att1={txt1},att2={txt2}.... last , can be present
    start = datastring.find("{")
    stop = datastring.find(",")
    citekey = datastring[start+1:stop]
    #from now starts attribute extrations
    cont = True #we suppose to have almost one attribute
    while cont:
        datastring = datastring[stop+1:]
        start = 0
        stop = datastring.find("}")
        analyzedstring = datastring[start:stop]
        analyzedstring = analyzedstring.strip();
        stop = analyzedstring.find("=")
        key = analyzedstring[start:stop]
        key = key.strip()
        start = analyzedstring.find("{")
        value = analyzedstring[start+1:]
        value = value.strip()
        attributevalue.append(key)
        attributevalue.append(value)
        stop = datastring.find(",")
        if stop == -1:
            cont = False
        else:
            stop = datastring.find("}")
            datastring = datastring[stop + 1:]
            stop = datastring.find(",")
    #print(len(attributevalue))
```

```
    returnvalue = key_values(citekey, attributevalue)
    return returnvalue


bibfile = sys.argv[1]
f = open(bibfile, "r")
recordstring = f.read()
classes = list(dict.fromkeys(re.findall('@(.*){', recordstring)))
recordstring=recordstring.replace('\n',' ')
#clean from {\
recordstringcopy=recordstring
recordstring=""
idx = 0
graphdepth = 0
while idx <= len(recordstringcopy)-2:
    simplycopy = True
    if (recordstringcopy[idx] == "{"):
        graphdepth = graphdepth+1
    if (recordstringcopy[idx] == "}"):
        graphdepth = graphdepth-1

    skipcopy = False
    if ( ( (recordstringcopy[idx] == "{") & (graphdepth >= 3) ) or
        ( (recordstringcopy[idx] == "}") & (graphdepth >= 2) ) ):
            skipcopy = True
            if recordstringcopy[idx] == "{":
                recordstring = recordstring + "__open_graph__"
            if recordstringcopy[idx] == "}":
                recordstring=recordstring+"__close_graph__"
    if ((recordstringcopy[idx] == "@") & (graphdepth >= 1)):
        skipcopy = True
        recordstring = recordstring + "__at__"
    if skipcopy == False:
        recordstring=recordstring+recordstringcopy[idx]
    idx = idx+1

# recordarray=recordstring.split"
for classelementname in classes:
    fileToWrite = classelementname+".tex"
    stringToWrite = ""
    # To generate, if present
    # Title
    # Author(s)
    # per Journal: journal + publisher
    # DOI
    # abstract
    # citeref

    attributestoprint = ["title", "author", "doi", "abstract"]
    if classelementname == "article":
        attributestoprint = ["title", "author", "journal", "publisher", "doi", "abstract"]

    elementnametosearch = "@" + classelementname + "{"
    classelementstarts = [i for i in range(len(recordstring)) if recordstring.startswith(elem
    for classelementstart in classelementstarts:
        #print(classelementname +" "+str(classelementstart))
```

```
        substringstartswithelement = recordstring[classelementstart+1:]
        stoptoken=substringstartswithelement.find("@")
        if stoptoken > 1:
            substringstartswithelement = substringstartswithelement[:stoptoken]
        keyvalue=get_data(substringstartswithelement)

        #print("key "+keyvalue.key)
        for idxatt in attributestoprint:
            for idx in range(int(len(keyvalue.values)/2)):
                #print(idx)
                if (keyvalue.values[2*idx] == idxatt):
                    # reinsert graph
                    txt = keyvalue.values[2 * idx + 1]
                    txt = txt.replace("__open_graph__", "{")
                    txt = txt.replace("__close_graph__", "}")
                    txt = txt.replace("__at__", "@")
                    txt = txt.replace("_", "\\_")
                    txt = txt.replace("&", "\\&")
                    attributetitletoprint = idxatt.capitalize()
                    if attributetitletoprint == "Doi":
                        attributetitletoprint = "DOI"
                    if attributetitletoprint == "Author":
                        if txt.find("and")>1:
                            attributetitletoprint = "Authors"
                        if txt.find(",")>1:
                            attributetitletoprint = "Authors"
                    stringprint = False
                    if idxatt == "title":
                        stringprint = True
                        httppage = "https://openportal.isti.cnr.it/results?qv=\"" + txt + "\"
                        stringToWrite = stringToWrite+"\\textbf{"+attributetitletoprint+"}: \
                                        httppage + "}{" + txt + "}\\\\ \n"
                        #print(httppage)
                        #contents = urllib.request.urlopen("\""+httppage+"\"").read()
                        #contents = urllib.request.urlopen("https://openportal.isti.cnr.it/re
                    if idxatt == "doi":
                        stringprint = True
                        stringToWrite = stringToWrite+"\\textbf{"+attributetitletoprint+"}: \
                    if idxatt == "abstract":
                        stringprint = True
                        stringToWrite = stringToWrite+"\\textbf{"+attributetitletoprint+"}: \
                    if  stringprint == False:
                        stringToWrite = stringToWrite + "\\textbf{" + attributetitletoprint +
        #add cite key
        stringToWrite = stringToWrite + "\cite{"+keyvalue.citekey+"}\\\\ \\vspace{5mm} \\\\ \
    with open(fileToWrite, 'w') as f:
        f.write(stringToWrite)
```

5 Conclusion and Future Works

The code does not face some peculiarity of TeX strings such as the presence of the underscore between dollars. This and others features will be implemented in future version of the code. The code is deployed in GitHub.

References

[1] S.F. Lott. *Modern Python Cookbook*. Packt Publishing, 2016.

[2] Leone G. R., Righi M., Carboni A., Caudai C., Colantonio S., Kuruoglu E. E., Leporini B., Magrini M., Paradisi P., Pascali M. A., Pieri G., Reggiannini M., Salerno E., Scozzari A., Tonazzini A., Fusco G., Galesi G., Martinelli M., Pardini F., Tampucci M., Buongiorno R., Bruno A., Germanese D., Matarese F., Coscetti S., Coltelli P., Jalil B., Benassi A., Bertini G., Salvetti O., and Moroni D. Si-lab annual research report 2020. Technical report, ISTI Annual Report, ISTI-2021-AR/001, pp.1–38, 2021, 2021.