**Ministerul Educației, Culturii și Cercetării al Republicii Moldova**

|  |  |
| --- | --- |
| **D:\1-107\Stela Balan\DMSAC\Regulamente UTM, 2015\Logou nou UTM\Logo_inscript_vertical.png** | **UNIVERSITATEA TEHNICĂ A MOLDOVEI** |

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Informatica şi Ingineria Sistemelor**

**RAPORT**

**Lucrare de laborator Nr. 4**

**La disciplina Analiza statistică a datelor**

|  |  |
| --- | --- |
| **A efectuat:** | **st. gr.IA - 182**  **Ulmanu Cristian** |
| **A verificat:** | **conf.univ.**  **Bumbu Tudor** |

**Chișinău, 2020**

**Tema: Extragerea informaţiei**

**Sarcină:** **Dezvoltatarea unei aplicatii web/desktop/android care rezolva problema studiata.**

# Descrierea algortimilor utilizați

# Algoritmul de sumarizare( rezumarea textului)

Astăzi diverse organizații, fie că este vorba de cumpărături online, organizații guvernamentale și din sectorul privat, industria de catering și turism sau alte instituții care oferă servicii clienților sunt îngrijorați de clienții lor și solicită feedback de fiecare dată când le folosim serviciile. Luați în considerare faptul că este posibil ca aceste companii să primească cantități enorme de feedback pentru utilizatori în fiecare zi. Și ar deveni destul de obositor ca conducerea să stea și să analizeze fiecare dintre acestea.

Dar, tehnologiile de astăzi au ajuns într-o măsură în care pot îndeplini toate sarcinile ființelor umane. Iar domeniul care face ca aceste lucruri să se întâmple este Machine Learning. Mașinile au devenit capabile să înțeleagă limbile umane folosind prelucrarea limbajului natural. Astăzi se fac cercetări în domeniul analizei textului.

Și una dintre astfel de aplicații de analiză text și NLP este un rezumat de feedback care ajută la rezumarea și scurtarea textului în feedback-ul utilizatorului. Acest lucru poate fi realizat un algoritm pentru a reduce corpurile de text, dar păstrându-și semnificația inițială sau pentru a oferi o perspectivă excelentă asupra textului inițial.

Dacă sunteți interesat de Data Analytics, veți găsi foarte utilă învățarea despre procesarea limbajului natural. Python oferă o imensă asistență de bibliotecă pentru NLP. Vom folosi NLTK - setul de instrumente pentru limbajul natural. care ne va servi scopul corect.

Să înțelegem pașii -

**Pasul 1: Importarea bibliotecilor necesare**

Există două biblioteci NLTK care vor fi necesare pentru crearea unui rezumat eficient de feedback.

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize

Termeni folosiți:

*Corpus*

Corpus înseamnă o colecție de text. Ar putea fi seturi de date despre orice conținând texte, fie că este vorba de poezii ale unui anumit poet, corpuri de lucrări ale unui anumit autor etc.

*Tokenizatoare*

, împarte un text într-o serie de jetoane. Există trei tokenizatoare principale - cuvânt, propoziție și tokenizator regex. Vom folosi doar simbolul cuvântului și propoziției

**Pasul 2: Îndepărtarea cuvintelor de oprire și stocarea lor într-un șir separat de cuvinte.**

*Stop Word*

Orice cuvânt de tipul (este, a, an, the, for) care nu adaugă valoare sensului unei propoziții.

**Pasul 3: Crearea unui tabel de frecvențe de cuvinte**

Un dicționar python care va ține o evidență de câte ori apare fiecare cuvânt în feedback după eliminarea cuvintelor oprite. Putem utiliza dicționarul peste fiecare propoziție pentru a ști care propoziții au cele mai relevante conținut în textul general.

**Pasul 4: Alocați punctaj fiecărei propoziții în funcție de cuvintele pe care le conține și de tabelul de frecvențe**

Putem folosi metoda send\_tokenize () pentru a crea o serie de propoziții. În al doilea rând, vom avea nevoie de un dicționar care să păstreze scorul fiecărei propoziții, vom trece ulterior prin dicționar pentru a genera rezumatul.

**Pasul 5: Alocați un anumit punctaj pentru a compara propozițiile din feedback.**

O abordare simplă pentru a compara scorurile noastre ar fi găsirea punctajului mediu al unei propoziții. Media în sine poate fi un prag bun.

# Algoritmul RAKE

Marcajele algoritmului RAKE sunt

* capacitatea sa de a opera independent pe documente, fără a face referire la un corpus (independența domeniului); și
* precizia sa foarte rezonabilă, în ciuda simplității și eficienței computationale.

Rake - ul este construit pe observația că toate cuvintele cheie conțin , de obicei , mai multe cuvinte informative (numite cuvinte de conținut) , dar nu și punctuație stopwords . Așadar, într-un document despre diverse alimente pe bază de porumb, „frânte de porumb”, „floricele” și „fulgi de porumb” ar putea apărea ca cuvinte cheie, în timp ce „porumb pe păpușă” nu ar fi luat în considerare, deoarece are două cuvinte cheie foarte comune: „ pe ”și„ the ”. Întregul algoritm este următorul.

Având în vedere un document de introducere din care dorim să extragem cuvinte cheie,

Împărțiți documentul într-o serie de cuvinte, împărțindu-l la delimitatoare de cuvinte (ca spații și punctuație).

Împărțiți cuvintele în secvențe de cuvinte contigue, rupând fiecare secvență la un cuvânt de stop. Fiecare secvență este acum un „cuvânt cheie candidat”.

Să vedem ce avem până acum.

Luați în considerare textul scurt „O lingură de înghețată”. Îl despartim în cuvinte pentru a obține

["A", "scoop", "of", "ice", "cream"]

Pasul 2 aranjează aceste cuvinte în secvențe, evitând parolele. Cuvintele „A” și „din” trebuie să fie pe orice listă de opțiune pe care le utilizați. Așadar, citind tabloul de la stânga la dreapta, sărind cuvinte cheie și creând un nou cuvânt cheie candidat de fiecare dată când se întâlnește un cuvânt stop, obținem două cuvinte cheie candidate:

[„buză”, „înghețată”]

Acum înapoi la algoritm.

Calculați „punctajul” fiecărui cuvânt indivudual din lista cuvintelor cheie candidate. Aceasta se calculează folosind metrica:

grad (cuvânt) / frecvență (cuvânt)

Este ușor de înțeles care este frecvența unui cuvânt. Este pur și simplu numărul de ocazii în care apare cuvântul în întreaga listă de cuvinte cheie candidate. Așadar, frecvențele noastre de cuvinte sunt:

frecvență ("scoop") = 1

frecvență ("gheață") = 1

frecvență ("cremă") = 1

# Evitarea blocării a web scraping

Cum să evitați blocurile?

În general, site-urile web nu le plac pe răzuitoarele bot, dar probabil nu îl împiedică complet din cauza bot-urilor motorului de căutare care zgârie site-urile pentru a le clasifica. Există un standard de excludere a roboților care definește termenii și condițiile site-ului web cu crawler-urile bot, care se găsește de obicei în fișierul robots.txt al site-ului. De exemplu, fișierul robots.txt al Wikipedia poate fi găsit aici: https://en.wikipedia.org/robots.txt .

După cum vedeți, restricțiile Wikipedia nu sunt prea stricte. Cu toate acestea, unele site-uri web sunt foarte stricte și nu permit accesarea cu crawlere a unei părți a site-ului sau a tuturor acestuia.

Cum să faci față blocurilor?

Un mod de a face acest lucru este prin rotirea prin diferite proxy-uri și agenți utilizatori (antete) atunci când faceți cereri către site-ul web. De asemenea, este important să aveți în vedere cât de des faceți solicitări către site-ul web pentru a evita să fiți „spamer”.

Implementarea unui server proxy se poate face cu ușurință în Python. O listă de proxies gratuite poate fi găsită aici (Rețineți că proxy-urile gratuite sunt de obicei mai puțin stabile și mai lente decât cele plătite. Dacă nu găsiți cele gratuite suficient de bune pentru nevoile dvs., puteți lua în considerare obținerea unui serviciu plătit).

Analizând lista de proxies gratuite, se poate utiliza BeautifulSoup pentru a obține adrese IP și porturi. Structura site-ului menționat mai sus poate fi văzută mai jos.

Bazin de antet

Există multe anteturi HTTP care pot fi transmise ca parte a unei solicitări atunci când utilizați pachetul de solicitări în Python. Am trecut de două elemente de antet (care au fost suficiente pentru noi), respectiv antetul Accept (permisiunile utilizatorului) și agentul utilizator (Pseudo-Browser).

Grupul de anteturi pseudo aleatoare a fost creat după cum urmează (a se vedea codul de mai jos):

Creați un obiect de dicționar cu „acceptă” unde fiecare antet accept este legat de un browser specific (în funcție de agentul utilizator). O listă cu anteturile acceptate poate fi găsită aici . Această listă conține valori implicite pentru fiecare agent de utilizator și poate fi modificată.

Obțineți un agent de utilizator aleatoriu folosind pachetul fals-utilizatori în Python. Acest lucru este foarte ușor de utilizat așa cum se vede în codul de mai jos. Vă sugerăm să creați o listă de agenți utilizatori în prealabil doar în cazul în care falsificatorul nu este disponibil. Un exemplu de user-agent: „Mozilla / 5.0 (Windows NT 6.2; rv: 21.0) Gecko / 20130326 Firefox / 21.0”

# Algoritmul aplicației



# Modulul lucrarefinal.py codul sursă

import tkinter as tk #interfata GUI

from tkinter import ttk

import requests #apelam siteul

import re #expresii regulate

from bs4 import BeautifulSoup #convertim html

from xlwt import Workbook #inscriem in excel

from docx import Document #inscriem in word

import os #cream fisiere si salvam in ele

from rake\_nltk import Rake #extragem cuvintele cheie

import nltk #analizam textul

from os import path #stergem fisiere

from nltk.corpus import stopwords #gasim stopwords

import gethtml #modul creat de mine se foloseste la cautarea cheilor dupa url

#cream interfata grafica

form=tk.Tk()

form.title("Lucrare de An ASD")

form.geometry("1200x1000")

tab\_parent=ttk.Notebook(form)

tab1=ttk.Frame(tab\_parent)

tab2=ttk.Frame(tab\_parent)

tab\_parent.add(tab1, text="Extract Articles")

tab\_parent.add(tab2, text="Extract Keywords")

#===First tab===

#===GUI====

label1=tk.Label(tab1, text="Keyword:")

entry1=tk.Entry(tab1)

text1=tk.Text(tab1, height=15, width=25)

label1.grid(row=0,column=0,padx=15,pady=15)

entry1.grid(row=0,column=1,padx=15,pady=15)

#functie de inscriere keywords in xls

def sheet(text):

#adresa site-ului, adaugam xls

doclink="http://www.scholarpedia.org"+text

print(doclink)

wb = Workbook()

sheet1 = wb.add\_sheet('Sheet 1')

headers1 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

document = Document()

link1=requests.get(doclink,headers=headers1).text

soup1=BeautifulSoup(link1,'lxml')

#extragem info din tagul p si h1

selectall2=soup1.find\_all("p")

selecttitle=str(\*soup1.find\_all("h1"))

print(selecttitle)

#print(\*selectall,sep='\n')

stri2 = ""

i=0

for lin in selectall2:

stri2+=str(lin)

i=i+1

#filtram textul de taguri html

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

stri2=re.sub(clean, '', stri2)

sheet1.write(0, 0, 'Cuvinte Cheie')

sheet1.write(0, 1, 'Rake Rank')

#utilizam algortimtul rake

r=Rake()

r.extract\_keywords\_from\_text(stri2)

print("\n".join(r.get\_ranked\_phrases()))

print(\*r.get\_ranked\_phrases\_with\_scores(),sep='\n')

#rank,key=r.get\_ranked\_phrases\_with\_scores()

rank=r.get\_ranked\_phrases\_with\_scores()

print(rank)

word=[0 for x in range(len(rank))]

ranked=[0 for x in range(len(rank))]

j=0

for khh in rank:

ranked[j],word[j]=khh

j=j+1

#inscriem in xls

for g in range(len(ranked)):

sheet1.write(g+1, 0, word[g])

sheet1.write(g+1, 1, ranked[g])

if path.exists("key.xls"):

os.remove("key.xls")

wb.save('key.xls' )

else:

wb.save('key.xls' )

#functie de inscriere in document a articolelor

def document(text):

#adresa site-ului, adaugam doc

doclink="http://www.scholarpedia.org"+text

print(doclink)

headers1 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

document = Document()

link1=requests.get(doclink,headers=headers1).text

soup1=BeautifulSoup(link1,'lxml')

#extragem info din tagul p si h1

selectall2=soup1.find\_all("p")

selecttitle=str(\*soup1.find\_all("h1"))

print(selecttitle)

#print(\*selectall,sep='\n')

stri2 = ['' for stri2 in range(len(selectall2))]

i=0

for lin in selectall2:

stri2[i]=str(lin)

i=i+1

#filtram textul de taguri html

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

selecttitle=re.sub(clean, '', selecttitle)

document.add\_heading(selecttitle, level=1)

for i in stri2:

clean = re.compile('<.\*?>')

i=re.sub(clean, '', i)

document.add\_paragraph(i, style='Intense Quote')

#inscriem in doc file

if path.exists("document.docx"):

os.remove("document.docx")

document.save('document.docx' )

else:

document.save('document.docx' )

#funtie de inscriere a rezumatului in document

def documentsummary(text):

doclink="http://www.scholarpedia.org"+text

print(doclink)

headers1 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

document = Document()

link1=requests.get(doclink,headers=headers1).text

soup1=BeautifulSoup(link1,'lxml')

selectall2=soup1.find\_all("p")

selecttitle=str(\*soup1.find\_all("h1"))

print(selecttitle)

#print(\*selectall,sep='\n')

stri2 = ['' for stri2 in range(len(selectall2))]

i=0

for lin in selectall2:

stri2[i]=str(lin)

i=i+1

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

selecttitle=re.sub(clean, '', selecttitle)

document.add\_heading(selecttitle, level=1)

for i in range(len(selecttitle)):

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

stri2[i]=re.sub(clean, '', stri2[i])

i=0

#utilizarea algoritmului de rezumare a textului

while i<len(stri2):

text=str(stri2[i])

print(text)

stopWords = set(stopwords.words("english"))

words = nltk.word\_tokenize(text)

freqTable = dict()

for word in words:

word = word.lower()

if word in stopWords:

continue

if word in freqTable:

freqTable[word] += 1

else:

freqTable[word] = 1

sentences = nltk.sent\_tokenize(text)

sentenceValue = dict()

for sentence in sentences:

for word, freq in freqTable.items():

if word in sentence.lower():

if sentence in sentenceValue:

sentenceValue[sentence] += freq

else:

sentenceValue[sentence] = freq

sumValues = 0

for sentence in sentenceValue:

sumValues += sentenceValue[sentence]

if len(sentenceValue)>0:

average = int(sumValues / len(sentenceValue))

summary = ''

for sentence in sentences:

if (sentence in sentenceValue) and (sentenceValue[sentence] > (1.2 \* average)):

summary = " " + sentence

print(summary)

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

summary=re.sub(clean, '', summary)

document.add\_paragraph(summary, style='Intense Quote')

else:

document.add\_paragraph(text, style='Intense Quote')

i=i+1

if path.exists("documentrez.docx"):

os.remove("documentrez.docx")

document.save('documentrez.docx' )

else:

document.save('documentrez.docx' )

#functie de afisare a rezultatelor cautarii dupa cuvant cheie

def titlecontent():

query=entry1.get()

who\_link="http://www.scholarpedia.org/w/index.php?title=Special%3ASearch&profile=default&search="+query+"&fulltext=Search"

headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

link=requests.get(who\_link,headers=headers).text

soup=BeautifulSoup(link,'lxml')

selectall=soup.find\_all("div",{"class": "searchresult"})

#print(\*selectall,sep='\n')

stri = ['' for stri in range(len(selectall))]

i=0

for lin in selectall:

stri[i]=str(lin)

i=i+1

for i in stri:

clean = re.compile('<.\*?>')

i=re.sub(clean, '', i)

selectall1=soup.find\_all("div",{"class": "mw-search-result-heading"})

#print(\*selectall,sep='\n')

stri1 = ['' for stri1 in range(len(selectall))]

i=0

for lin in selectall1:

stri1[i]=str(lin)

i=i+1

for i in stri1:

clean = re.compile('<.\*?>')

i=re.sub(clean, '', i)

kl = ['' for kl in range(len(selectall))]

i=0

for lin in selectall1:

kl[i]=str(lin.find('a').get('href'))

i=i+1

btntxt=[0 for x in range(len(selectall))]

btntxt1=[0 for x in range(len(selectall))]

btntxt2=[0 for x in range(len(selectall))]

txt=[0 for x in range(len(selectall))]

for x in range(20):

txt[x] = tk.Text(tab1,height=2, width=35)

clean = re.compile('<.\*?>')

stri1[x]=re.sub(clean, '', stri1[x])

btntxt[x]=tk.Button(tab1, text="Document\_"+str(x),command =lambda x1=x: document(kl[x1]))

btntxt1[x]=tk.Button(tab1, text="Summarize\_"+str(x),command =lambda x1=x: documentsummary(kl[x1]))

btntxt2[x]=tk.Button(tab1, text="Keywords\_"+str(x),command =lambda x1=x: sheet(kl[x1]))

print(kl[x])

txt[x].insert(tk.INSERT, stri1[x])

txt[x].grid(column=0, row=x+1)

btntxt[x].grid(column=2, row=x+1)

btntxt1[x].grid(column=3, row=x+1)

btntxt2[x].grid(column=4, row=x+1)

txt1=[0 for x in range(len(selectall))]

for x in range(20):

txt1[x] = tk.Text(tab1,height=2, width=80)

clean = re.compile('<.\*?>')

stri[x]=re.sub(clean, '', stri[x])

txt1[x].insert(tk.INSERT, stri[x])

txt1[x].grid(column=1, row=x+1)

#functie de inscriere a rezultatelor cautarii in xls

def excel():

query=entry1.get()

wb = Workbook()

sheet1 = wb.add\_sheet('Sheet 1')

who\_link="http://www.scholarpedia.org/w/index.php?title=Special%3ASearch&profile=default&search="+query+"&fulltext=Search"

headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

link=requests.get(who\_link,headers=headers).text

soup=BeautifulSoup(link,'lxml')

selectall=soup.find\_all("div",{"class": "searchresult"})

#print(\*selectall,sep='\n')

stri = ['' for stri in range(len(selectall))]

i=0

for lin in selectall:

stri[i]=str(lin)

i=i+1

for i in stri:

clean = re.compile('<.\*?>')

i=re.sub(clean, '', i)

selectall1=soup.find\_all("div",{"class": "mw-search-result-heading"})

#print(\*selectall,sep='\n')

stri1 = ['' for stri1 in range(len(selectall))]

i=0

for lin in selectall1:

stri1[i]=str(lin)

i=i+1

for i in stri1:

clean = re.compile('<.\*?>')

i=re.sub(clean, '', i)

for x in range(20):

clean = re.compile('<.\*?>')

stri1[x]=re.sub(clean, '', stri1[x])

for x in range(20):

clean = re.compile('<.\*?>')

stri[x]=re.sub(clean, '', stri[x])

sheet1.write(0, 0, 'DENUMIRI ARTICOLE')

sheet1.write(0, 1, 'DESCRIERE ARTICOLE')

for x in range(20):

sheet1.write(x+1, 0, stri1[x])

sheet1.write(x+1, 1, stri[x])

wb.save(query+'.xls')

btn1=tk.Button(tab1, text="Search",command = titlecontent)

btn1.grid(row=0,column=2,padx=15,pady=15)

btn2=tk.Button(tab1, text="Excel Save",command = excel)

btn2.grid(row=0,column=3,padx=15,pady=15)

#===Second tab===

#extragerea keywords din site dupa url

def keywordextr():

text21.delete("1.0",tk.END)

query=entry21.get()

print(gethtml.gethtml(query))

article=gethtml.getarticle(query)

word=gethtml.getkeywords(article)

index=gethtml.getkeywordsindex(article)

for w in range(30):

text21.insert(tk.INSERT, str(word[w])+" "+str(index[w])+"\n")

print(gethtml.getkeywords(article))

#inscrierea keywords in excel

def excelfinal():

wb = Workbook()

sheet1 = wb.add\_sheet('Sheet 1')

sheet1.write(0, 0, 'Cuvinte Cheie')

sheet1.write(0, 1, 'Nr. aparitii')

query=entry21.get()

print(gethtml.gethtml(query))

article=gethtml.getarticle(query)

word=gethtml.getkeywords(article)

index=gethtml.getkeywordsindex(article)

for w in range(30):

sheet1.write(w+1, 0, word[w])

sheet1.write(w+1, 1, index[w])

if path.exists("keywords.xls"):

os.remove("keywords.xls")

wb.save('keywords.xls' )

else:

wb.save('keywords.xls' )

label21=tk.Label(tab2, text="Site:")

entry21=tk.Entry(tab2,width=50)

text21=tk.Text(tab2, height=30, width=20)

label21.grid(row=0,column=0,padx=15,pady=15)

entry21.grid(row=0,column=1,padx=15,pady=15)

btn21=tk.Button(tab2, text="Extract keywords",command = keywordextr)

btn21.grid(row=0,column=2,padx=15,pady=15)

text21.grid(row=1,column=0,padx=15,pady=15)

btn22=tk.Button(tab2, text="Excel Save",command = excelfinal)

btn22.grid(row=0,column=3,padx=15,pady=15)

tab\_parent.pack(expand=1, fill='both')

form.mainloop()

# Modulul gethtml.py codul sursă

import mechanize

from bs4 import BeautifulSoup

import requests

import re

def gethtml(text):

br=mechanize.Browser()

htmltext=br.open(text)

return htmltext

def gethtmlfile(text):

br=mechanize.Browser()

htmlfile=br.open(text)

return htmlfile

def getarticletext(text):

#link1=requests.get(text).text

soup1=BeautifulSoup(text,'lxml')

tags=""

for tag in soup1.findAll('p'):

clean = re.compile('<.\*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

tags+=re.sub(clean, '', str(tag))

print (tags)

return tags

def getarticle(url):

htmltext=gethtml(url)

return getarticletext(htmltext)

def getkeywords(text):

common=open("content.txt").read().split('\n')

word\_dict={}

word\_list=text.lower().split()

for word in word\_list :

if word not in common and word.isalnum():

if word not in word\_dict:

word\_dict[word]=1

if word in word\_dict:

word\_dict[word]+=1

top\_words=sorted(word\_dict.items(),key=lambda kv: (-kv[1], kv[0]),reverse=False)[0:30]

top30=[]

for w in top\_words:

top30.append(w[0])

return top30

def getkeywordsindex(text):

common=open("content.txt").read().split('\n')

word\_dict={}

word\_list=text.lower().split()

for word in word\_list :

if word not in common and word.isalnum():

if word not in word\_dict:

word\_dict[word]=1

if word in word\_dict:

word\_dict[word]+=1

top\_words=sorted(word\_dict.items(),key=lambda kv: (-kv[1], kv[0]),reverse=False)[0:30]

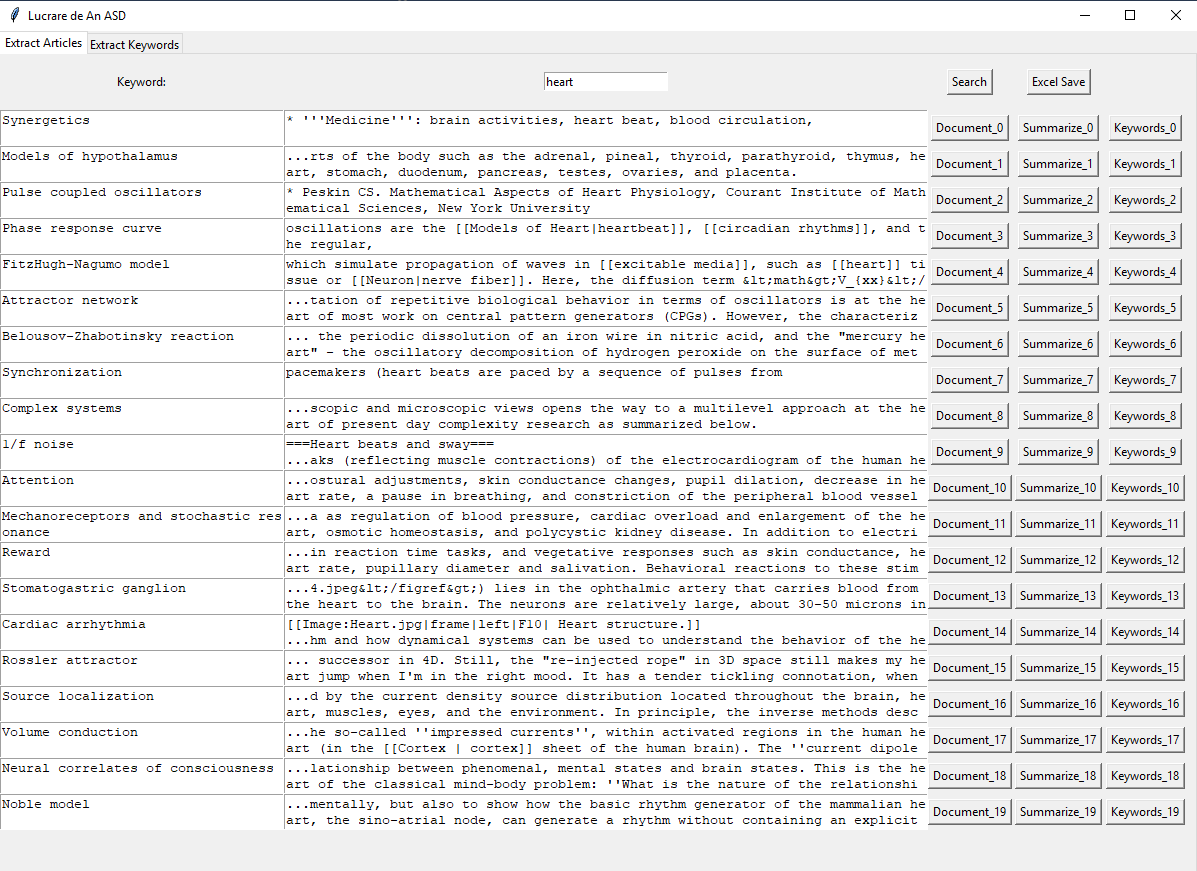
top30=[]

for w in top\_words:

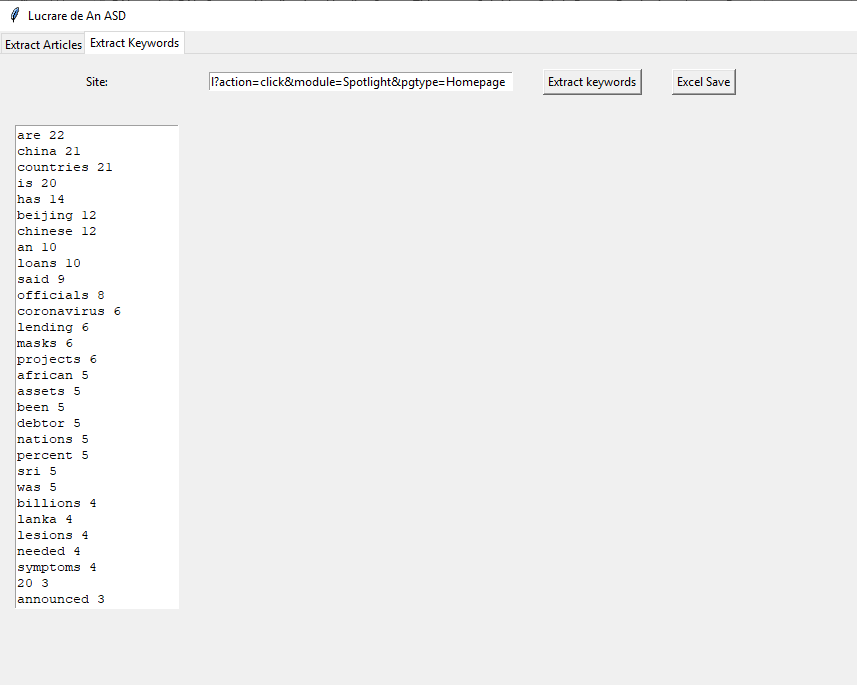
top30.append(w[1])

return top30

# Descrierea aplicației și utilizării acesteia



Deci GUI a aplicației are 2 taburi accesând primul noi putem căuta pe siteul <http://www.scholarpedia.org> după o cheie pe care noi o dorim, odată introdusă cheia în caseta text și apăsând pe search vor apărea rezultatele căutării, în prima coloană vor fi denumirile articolelor, în a doua descrierea lor și la fiecare vor fi 3 butoane, unul de extragere a informației neprelucrate, unul de extragere a rezumatului și ultimul buton de extragere a cuvintelor cheie prin algoritmul rake.



Al doilea tab permite extragerea cheilor după frecvența lor, noi introducem adresa url a site-ului și în caseta text ne va arăta top 30 cele mai importante cuvinte chei.

# Bibliografie

1. <https://habr.com/ru/post/280238/>
2. <https://towardsdatascience.com/deep-learning-for-specific-information-extraction-from-unstructured-texts-12c5b9dceada>
3. <https://www.geeksforgeeks.org/python-text-summarizer/>
4. <https://www.airpair.com/nlp/keyword-extraction-tutorial>
5. <https://codelingo.wordpress.com/2017/05/26/keyword-extraction-using-rake/>
6. <https://habr.com/ru/post/133337/>
7. <https://www.tutorialspoint.com/python/python_gui_programming.htm>
8. <https://www.homeandlearn.uk/python-database-form-tabs2.html>
9. <https://medium.com/@jorlugaqui/how-to-strip-html-tags-from-a-string-in-python-7cb81a2bbf44>
10. <https://python-docx.readthedocs.io/en/latest/user/quickstart.html>