**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 Curs**

**La disciplina Analiza statistică a datelor**

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

**Chișinău, 2020**

**CUPRINS**

[Subtaskuri tipice EI 3](#_Toc40955674)

[Scraping web 3](#_Toc40955675)

[Metode web scraping 4](#_Toc40955676)

[Utilizare web scraping 6](#_Toc40955677)

[Instrumente utilizate 7](#_Toc40955678)

[Python 7](#_Toc40955679)

[HTML 7](#_Toc40955680)

[DOM 8](#_Toc40955681)

[Biblioteci utilizate în Python 8](#_Toc40955682)

[Tkinter 8](#_Toc40955683)

[Requests 9](#_Toc40955684)

[RegEx 9](#_Toc40955685)

[BeautifulSoup4 9](#_Toc40955686)

[XLWT 10](#_Toc40955687)

[OS 10](#_Toc40955688)

[Rake\_nltk 10](#_Toc40955689)

[NLTK 10](#_Toc40955690)

[Mechanize 11](#_Toc40955691)

[Site-uri utilizate pentru scraping 11](#_Toc40955692)

[Schema generală a procesului de web scraping 12](#_Toc40955693)

[Structura funcțională a aplicației 13](#_Toc40955694)

[Algoritmul aplicației 14](#_Toc40955695)

[Descrierea algortimilor utilizați 14](#_Toc40955696)

[Algoritmul de sumarizare( rezumarea textului) 14](#_Toc40955697)

[Algoritmul RAKE 16](#_Toc40955698)

[Evitarea blocării a web scraping 17](#_Toc40955699)

[Algoritmul aplicației 18](#_Toc40955700)

[Modulul lucrarefinal.py codul sursă 18](#_Toc40955701)

[Modulul gethtml.py codul sursă 28](#_Toc40955702)

[Descrierea aplicației și utilizării acesteia 30](#_Toc40955703)

[Concluzii 31](#_Toc40955704)

[Bibliografie 32](#_Toc40955705)

**Tema: Extragerea informaţiei**

**Extragerea informațiilor (IE), recuperarea informațiilor (IR) este sarcina de a extrage automat informații structurate din documente nestructurate și / sau semi-structurate care pot fi citite de mașini și alte surse reprezentate electronic. În majoritatea cazurilor, această activitate se referă la procesarea textului limbajului uman prin procesarea limbajului natural (PNL). Ultimele activități în procesarea documentelor multimedia, cum ar fi adnotarea automată și extragerea conținutului din imagini / audio / video / documente, pot fi considerate ca extracție de informații.**

În tehnologia informației moderne, datorită creșterii rapide a cantității de informații nestructurate (fără metadate) (în special pe internet), acest proces, cum ar fi extragerea informațiilor, a devenit tot mai important. Aceste informații pot fi construite în continuare prin trecerea la formularul de relație sau adăugarea de etichete XML. Când utilizați agenți inteligenți pentru a monitoriza sursele de știri, sunt necesare câteva metode pentru extragerea informațiilor și transformarea acesteia într-o formă mai convenabilă pentru utilizarea ulterioară.

# Subtaskuri tipice EI

Subtasks tipice pentru extragerea informațiilor:

* Recunoașterea elementelor (entităților) numite, de exemplu: nume de persoane, nume de organizații, nume geografice, evenimente, simboluri de timp și bani etc.
* Rezolvarea anaforei și coreferenței: căutarea de legături legate de același obiect. Un caz tipic al unor astfel de referințe este pronume anafora.
* Evidențierea terminologiei: găsirea unui text dat cuvinte cheie și expresii (colocări).
* Auto-abstractizare: selectarea informațiilor semantice, emoționale, evaluative etc. dintr-un text. Este generativ și declarativ.

# Scraping web

Scraping web (sau razuire sau razuire ← razuirea web engleză) este o tehnologie pentru obținerea de date web prin extragerea acestora din paginile cu resurse web. Răzuirea Web poate fi făcută manual de către un utilizator de computer, dar termenul se referă de obicei la procesele automatizate implementate folosind codul care efectuează solicitări GET către site-ul țintă. Răzuirea Web este utilizată pentru a sintetiza paginile web în forme mai convenabile pentru lucru. Paginile web sunt create folosind limbaje de marcare a textului (HTML și XHTML) și conțin o mulțime de date utile în cod. Cu toate acestea, majoritatea resurselor web sunt destinate utilizatorilor finali și nu pentru comoditatea utilizării automate, astfel încât a fost dezvoltată o tehnologie care „curăță” conținutul web. Descărcarea și vizualizarea paginii sunt cele mai importante componente ale tehnologiei, ele fac parte integrantă din selecția datelor.

# Metode web scraping

Scraping-ul web este un domeniu de dezvoltare activă care împărtășește o inițiativă ambițioasă de dezvoltare a interacțiunii om-calculator, care necesită descoperiri în procesarea și înțelegerea textului paginilor online prin inteligență artificială. Soluțiile moderne de răzuire variază de la sisteme speciale, intensificate în muncă, până la sisteme complet automatizate care pot transforma site-uri web întregi în informații structurate într-un format specific. În mod ideal, atunci când site-ul ale cărui date pe care doriți să le extrage le oferă prin intermediul API-ului accesul inter-domeniu permis. În cazul în care lucrurile nu stau așa, puteți apela la alte metode de răzuire.

1. „Copy-paste” manual  
     
   Uneori, chiar și cea mai bună tehnologie de scraping web nu poate înlocui munca manuală a unei persoane atunci când un utilizator copiază și lipeste textul. În unele cazuri, aceasta este singura soluție posibilă, de exemplu, atunci când site-urile web blochează razuirea web și copiază textul.
2. Acces la servicii proxy  
     
   Dacă site-ul este un document html sau xml și îi sunt permise cereri de mai multe domenii, atunci puteți obține conținutul documentului folosind o solicitare la unul dintre serviciile proxy disponibile pe Internet.
3. Potrivirea modelelor de text  
     
   Un mod simplu, dar puternic, de a prelua informațiile din paginile web. Se poate baza pe comanda UNIX grep (efectuează o căutare în unul sau mai multe fișiere după un model) sau pe potrivirea expresiilor obișnuite ale limbajelor de programare (de exemplu, Perl sau Python).
4. Analizare HTML  
     
   Multe site-uri web constau într-un număr mare de pagini generate dinamic din principala sursă structurată - baza de date. Datele din aceeași categorie sunt de obicei codificate în pagini similare folosind un script comun sau un șablon. În extragerea datelor, un program care detectează astfel de tipare într-o sursă de informații specifice, extrage conținutul său și îl traduce într-o formă, numită coajă. Se presupune că paginile analizate ale sistemului corespund unui șablon comun și că pot fi ușor identificate în termeni de o schemă URL comună. În plus, unele limbaje semi-structurate de interogare a datelor, cum ar fi XQuery și HTQL, pot fi utilizate pentru a analiza pagini HTML și a extrage și transforma conținutul paginii.
5. Model de obiect de document (DOM)  
     
   DOM este un program cu o API pentru documente HTML și XML. Prin încorporarea unui browser web complet, precum Internet Explorer sau un control al browserului Mozilla, programele pot extrage conținut dinamic generat de scripturile din partea clientului. Răspândirea arborelui DOM vă permite să accesați informații în părțile sale separate.
6. Agregarea verticală a datelor  
     
   Există mai multe companii care au dezvoltat platforme online speciale care creează și controlează numeroși roboți. Bots funcționează fără implicare directă umană și, în același timp, interacțiunea lor cu utilizatorii se produce fără comunicare cu site-ul țintă. Pregătirea include crearea unei baze de cunoștințe, datorită cărora este posibilă munca de bot. Bots cumulează date despre proprietățile individuale ale fiecărei resurse în conformitate cu condițiile specificate pentru compararea și analiza ulterioară a valorilor proprietății obținute. Fiabilitatea platformei este măsurată de calitatea informațiilor primite (de obicei numărul de câmpuri) și de scalabilitatea acesteia (până la sute sau mii de site-uri). Această scalabilitate este folosită în principal pentru a converti datele localizate la sfârșitul codului lung de site-uri pe care agregatorii convenționale le consideră dificile sau consumă prea mult timp pentru a colecta conținut.
7. Recunoașterea adnotărilor semantice  
     
   Unele pagini pot conține meta-date sau marcaje și adnotări semantice; folosind metoda de recunoaștere a adnotării semantice, puteți extrage astfel de pagini.
8. Analizoare de pagini  
     
   Dezvoltările se realizează în domeniul inteligenței artificiale, atunci când viziunea mașinii identifică datele, le interpretează, așa cum ar face o persoană, le preia.

Tehnologia de razuire web este convenabilă pentru transferul datelor de pe paginile web către forme mai convenabile, dar există și metode de analiză care, în cazul unei API deschise, pot rezolva mai productiv problema.

# Utilizare web scraping

Răzuirea web a devenit un instrument important pentru colectarea automată de informații pe Internet. Face parte din sistemele de informații de marketing (MIS) pentru formarea bazelor de date sau a băncilor de date, datorită cărora informațiile necesare sunt furnizate sub formă de rapoarte tabulare. Tehnologiile motoarelor de căutare, agregatoarele de conținut sunt, de asemenea, interconectate cu programele de răzuire web.

Scraping-ul web este o formă de copiere a datelor în care anumite informații necesare pentru primire sunt colectate de pe Internet și acumulate, de regulă, într-o bază de date locală centrală sau o foaie de calcul pentru extragerea sau analiza ulterioară. Software-ul de razuire a paginilor web poate accesa World Wide Web direct folosind Protocolul de transfer de hipertext, precum și printr-un browser web.

Conținutul paginii rezultate poate fi analizat, formatat, copiat într-o foaie de calcul și așa mai departe. Scraping-ul web ia de obicei o bucată de date dintr-o pagină pentru a o folosi în alt scop. Un exemplu de razuire este căutarea pe diverse site-uri și copierea numelor, numere de telefon, adrese de e-mail, adrese URL ale unui anumit subiect pentru a crea o bază de date de contacte.

Practic, răzuitoarele web vor rezolva următoarele probleme:

* Căutați informațiile necesare;
* Copiați date de pe Internet;
* Monitorizarea actualizărilor pe site-uri.

Scraping-ul web poate fi un instrument independent și poate servi la preluarea de informații direcționate, poate deveni, de asemenea, o componentă a dezvoltării web utilizate pentru indexarea web, minarea web și minarea datelor, monitorizarea online, modificările prețurilor și compararea acestora, pentru observare pentru concurență și alte colectări de date.

# Instrumente utilizate

## Python

Python este un limbaj de programare dinamic multi-paradigmă, creat în 1989 de programatorul olandez Guido van Rossum. Van Rossum este și în ziua de astăzi un lider al comunității de dezvoltatori de software care lucrează la perfecționarea limbajul Python și implementarea de bază a acestuia, CPython, scrisă în C. Python este un limbaj multifuncțional folosit de exemplu de către companii ca Google sau Yahoo! pentru programarea aplicațiilor web, însă există și o serie de aplicații științifice sau de divertisment programate parțial sau în întregime în Python. Popularitatea în creștere, dar și puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de către programatori specializați și chiar și la predarea limbajului în unele medii universitare. Din aceleași motive, multe sisteme bazate pe Unix, inclusiv Linux, BSD și Mac OS X includ din start interpretatorul CPython.

Python pune accentul pe curățenia și simplitatea codului, iar sintaxa sa le permite dezvoltatorilor să exprime unele idei programatice într-o manieră mai clară și mai concisă decât în alte limbaje de programare ca C. În ceea ce privește paradigma de programare, Python poate servi ca limbaj pentru software de tipul object-oriented, dar permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (garbage collector). Alt avantaj al limbajului este existența unei ample biblioteci standard de metode.

Implementarea de referință a Python este scrisă în C și poartă deci numele de CPython. Această implementare este software liber și este administrată de fundația Python Software Foundation.

## HTML

Hypertext Markup Language (HTML) este limbajul de marcare standard pentru documentele concepute pentru a fi afișate într-un browser web. Poate fi asistat de tehnologii precum Cascading Style Sheets (CSS) și limbaje de script cum ar fi JavaScript.

Navigatoarele Web primesc documente HTML de pe un server web sau de la stocarea locală și le redau în pagini web multimedia. HTML descrie structura unei pagini web în mod semantic și inițial a inclus indicii pentru aspectul documentului.

Elementele HTML sunt blocurile de construcție ale paginilor HTML. Cu construcții HTML, imagini și alte obiecte, cum ar fi formulare interactive, pot fi încorporate în pagina redată. HTML oferă un mijloc de a crea documente structurate notând semantică structurală pentru text cum ar fi titluri, paragrafe, liste, link-uri, citate și alte elemente. Elementele HTML sunt delimitate prin etichete, scrise cu paranteze de unghi. Etichete precum <img /> și <input /> introduc direct conținut în pagină. Alte etichete, cum ar fi <p> ​​înconjoară și oferă informații despre textul documentului și pot include alte etichete ca sub-elemente. Navigatoarele nu afișează etichetele HTML, ci le folosesc pentru a interpreta conținutul paginii.

HTML poate încorpora programe scrise într-un limbaj de script cum ar fi JavaScript, care afectează comportamentul și conținutul paginilor web. Includerea CSS definește aspectul și aspectul conținutului. World Wide Web Consortium (W3C), fost întreținător al HTML-ului și actual mentenant al standardelor CSS, a încurajat încă din 1997 utilizarea CSS peste HTML prezentativ explicit.

## DOM

Modelul de obiect de document (DOM) este o interfață inter-platformă și independentă de limbaj care tratează un document XML sau HTML ca o structură de arbore în care fiecare nod este un obiect reprezentând o parte a documentului. DOM reprezintă un document cu un arbore logic. Fiecare ramură a copacului se termină într-un nod și fiecare nod conține obiecte. Metodele DOM permit accesul programatic la arbore; cu ele se poate schimba structura, stilul sau conținutul unui document. Nodurile pot avea dispozitive de gestionare a evenimentelor atașate de ele. Odată declanșat un eveniment, gestionatorii de evenimente sunt executați.

Principala standardizare a DOM a fost gestionată de World Wide Web Consortium, care a elaborat ultima dată o recomandare în 2004. WHATWG a preluat dezvoltarea standardului, publicându-l ca document viu. W3C publică acum instantanee stabile ale standardului WHATWG.

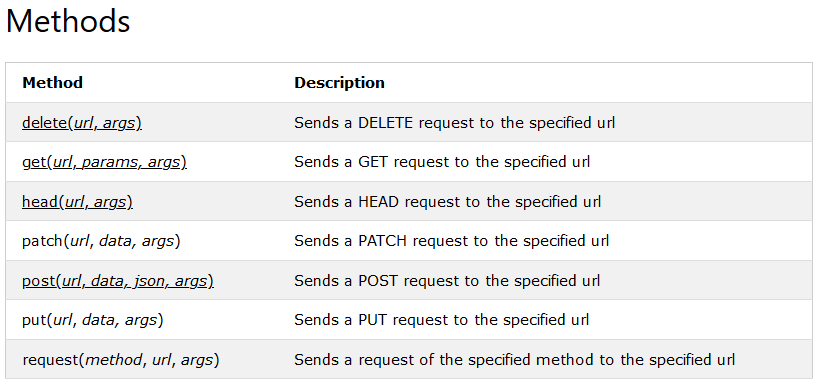
# Biblioteci utilizate în Python

## Tkinter

* Tkinter este biblioteca GUI standard pentru Python. Python atunci când este combinat cu Tkinter oferă un mod rapid și ușor de a crea aplicații GUI. Tkinter oferă o interfață puternică orientată pe obiect la setul de instrumente Tk GUI. Crearea unei aplicații GUI folosind Tkinter este o sarcină ușoară. Tot ce trebuie să faceți este să efectuați următorii pași -
* Importați modulul Tkinter.
* Creați fereastra principală a aplicației GUI.
* Adăugați unul sau mai multe dintre widget-urile menționate mai sus la aplicația GUI.
* Introduceți bucla evenimentului principal pentru a lua măsuri împotriva fiecărui eveniment declanșat de utilizator.

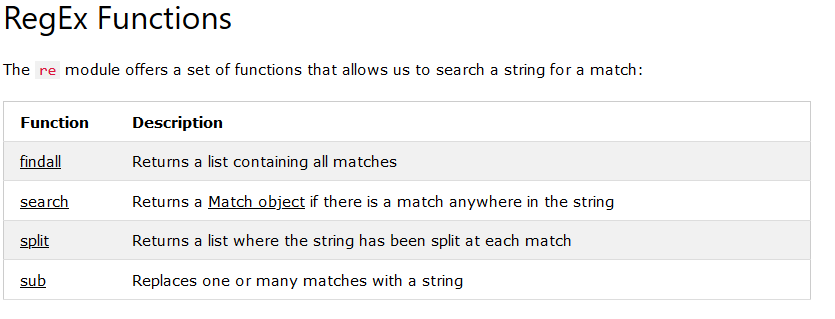
## Requests

Modulul de solicitări vă permite să trimiteți cereri HTTP folosind Python. Cererea HTTP returnează un obiect de răspuns cu toate datele de răspuns (conținut, codificare, stare, etc.).



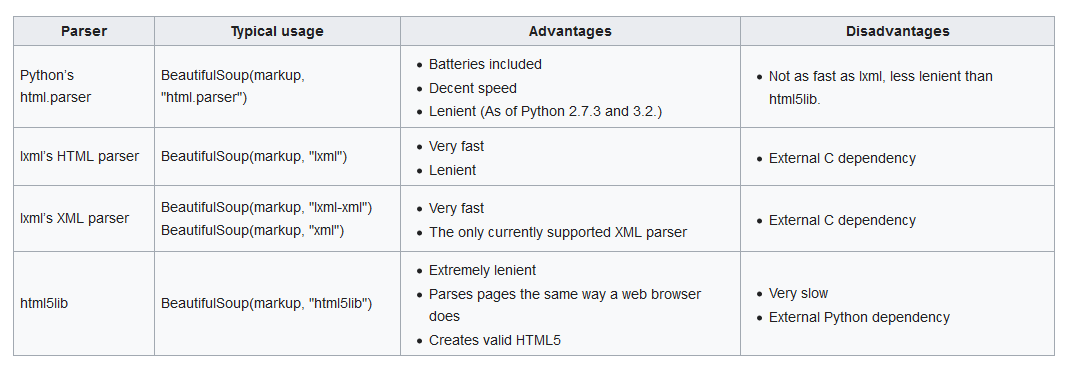
## RegEx

Un regEx, sau expresie regulată, este o secvență de caractere care formează un model de căutare. RegEx poate fi utilizat pentru a verifica dacă un șir conține modelul de căutare specificat.



## BeautifulSoup4

Beautiful Soup este un pachet Python pentru analizarea documentelor HTML și XML (incluzând marcaj malformat, adică etichete neîncheiate, denumite astfel după supă de etichete). Creează un arbore de analiză pentru paginile analizate care pot fi utilizate pentru a extrage date din HTML, care este utilă pentru razuirea web.  
  
Este disponibil pentru Python 2.7 și Python 3.



## XLWT

Aceasta este o bibliotecă pe care dezvoltatorii să o folosească pentru a genera fișiere cu foi de calcul compatibile cu versiunile Microsoft Excel 95 - 2003. Pachetul în sine este Python pur, fără dependențe de module sau pachete în afara distribuției standard Python.

## OS

Acest modul oferă un mod portabil de utilizare a funcționalității dependente de sistemul de operare. Dacă doriți doar să citiți sau să scrieți un fișier, consultați open (), dacă doriți să manipulați căi, consultați modulul os.path și dacă doriți să citiți toate liniile din toate fișierele de pe linia de comandă, consultați modulul de introducere fișier . Pentru crearea fișierelor și directoarelor temporare, consultați modulul tempfile, iar pentru gestionarea fișierelor și a directorilor la nivel înalt, consultați modulul shutil.

## Rake\_nltk

RAKE scurt pentru algoritmul de extragere rapidă a cuvintelor cheie, este un algoritm de extragere a cuvintelor cheie independent de domeniu, care încearcă să determine frazele cheie dintr-un corp de text, analizând frecvența aspectului cuvântului și coincidența acestuia cu alte cuvinte din text.

## NLTK

Natural Language Toolkit, sau mai frecvent NLTK, este o suită de biblioteci și programe pentru procesarea simbolică și statistică a limbajului natural (NLP) pentru limba engleză scrisă în limbajul de programare Python. A fost dezvoltat de Steven Bird și Edward Loper în Departamentul de Informatică și Știința Informației din Universitatea din Pennsylvania. NLTK include demonstrații grafice și date de eșantion. Este însoțită de o carte care explică conceptele care stau la baza sarcinilor de procesare a limbajului, susținute de setul de instrumente, plus o carte de bucate.

NLTK este destinat să sprijine cercetarea și predarea în PNL sau în domenii strâns legate, incluzând lingvistica empirică, știința cognitivă, inteligența artificială, regăsirea informațiilor și învățarea mașinii. NLTK a fost utilizat cu succes ca instrument de predare, ca instrument individual de studiu și ca platformă pentru prototipare și sisteme de cercetare a construcțiilor. Există 32 de universități din SUA și 25 de țări care utilizează NLTK în cursurile lor. NLTK acceptă funcționalități de clasificare, tokenizare, stemming, tagging, parsing și raționament semantic.

## Mechanize

Navigare web programatică de stat, după modulul Perl de Andy Lester WWW :: Mecanizați. mecanize.Browser implementează interfața urllib2.OpenerDirector. Obiectele browserului au o stare, inclusiv istoricul navigării, starea formei HTML, cookie-urile etc. Setul de caracteristici și scheme URL adresate de obiectele Browserului poate fi configurat. Biblioteca oferă, de asemenea, o API care este în mare parte compatibilă cu urllib2: programul dvs. urllib2 va funcționa probabil dacă înlocuiți „urllib2” cu „mecanizare” peste tot.

Caracteristicile includ: ftp:, http: și fișier: scheme URL, istoricul browserului, hyperlink și suport pentru formulare HTML, cookie-uri HTTP, HTTP-EQUIV și Actualizare, Referer [sic] antet, robots.txt, redirecționări, proxy, și Basic și Digest Autentificare HTTP.

O mare parte din codul provenit inițial din codul Perl de Gisle Aas (libwww-perl), Johnny Lee (suport pentru cookie-uri MSIE) și nu în ultimul rând Andy Lester (WWW :: Mecanizați). urllib2 a fost scris de Jeremy Hylton.

# Site-uri utilizate pentru scraping

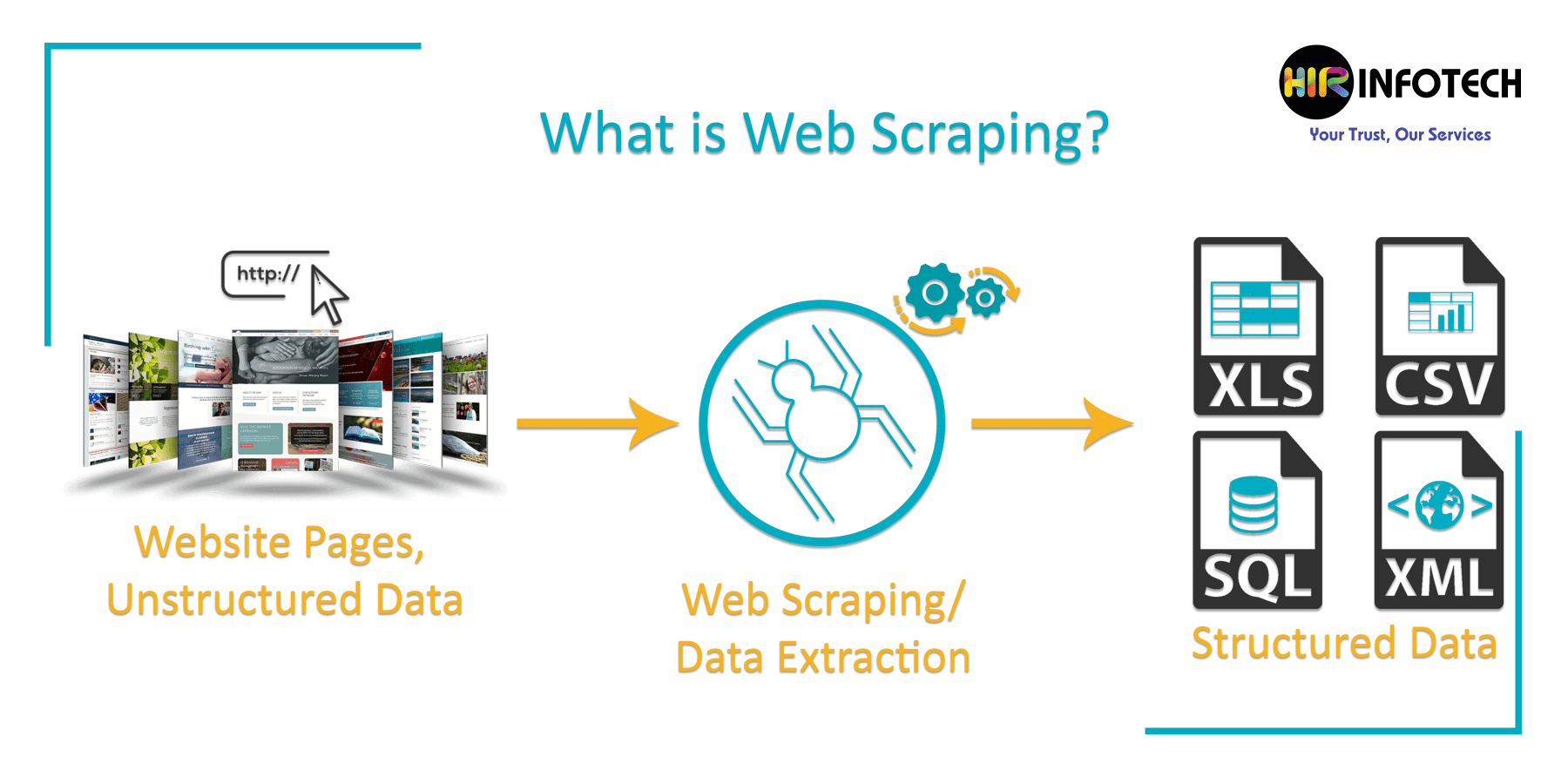
Scholarpedia este o enciclopedie online bazată pe wiki în limba engleză, cu caracteristici asociate în mod obișnuit cu reviste academice online cu acces deschis, care își propune să aibă conținut de calitate.

Articolele Scholarpedia sunt scrise de autorii experți invitați și sunt supuse revizuirii de la egal la egal. Scholarpedia listează numele și afilierea reală a tuturor autorilor, curatorilor și redactorilor implicați într-un articol: cu toate acestea, procesul de revizuire de la egal la egal (care poate sugera modificări sau adăugiri și trebuie să fie satisfăcut înainte de apariția unui articol) este anonim. Articolele Scholarpedia sunt stocate într-un depozit online și pot fi citate ca articole de jurnal convenționale (Scholarpedia are numărul ISSN ISSN 1941-6016). Sistemul de citare al Scholarpedia include suport pentru numerele de revizuire.

Proiectul a fost creat în februarie 2006 de Eugene M. Izhikevich, pe când era cercetător la Neurosciences Institute, San Diego, California. Izhikevici este și redactorul-șef al enciclopediei.

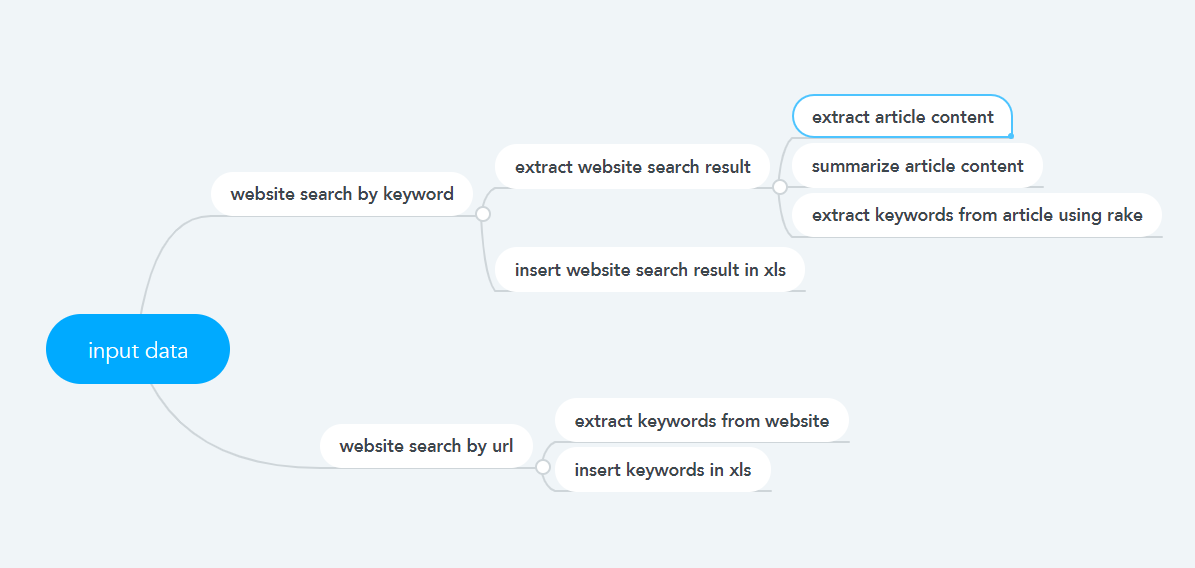
New York Times Company este o companie americană de mass-media care își publică numele de ziar, The New York Times. Arthur Ochs Sulzberger Jr. a ocupat funcția de președinte din 1997. Are sediul central în Manhattan, New York.

# Schema generală a procesului de web scraping



Procesul de web scraping presupune preluarea informației concrete de pe website-uri și sistematizarea, sumarizarea și structurarea lor astfel ca ele să poată fi memorizate în diferite tipuri de fișiere pentru utilizarea ulterioară. Procesul presupune în primul rând investigharea structurii site-ului de unde preluăm informația, pentru a determina din care elemente este necesar de extras informația. Pe urmă printr-un limbaj de programare, în cazul meu Python, se creează un script care preluează datele din anumite elemente ale DOM și după acelea ele pot fi sortate, rezumate, extrase cuvintele cheie și această informație poate fi înscrisă în xls sau doc sau în cadrul unei baze de date.

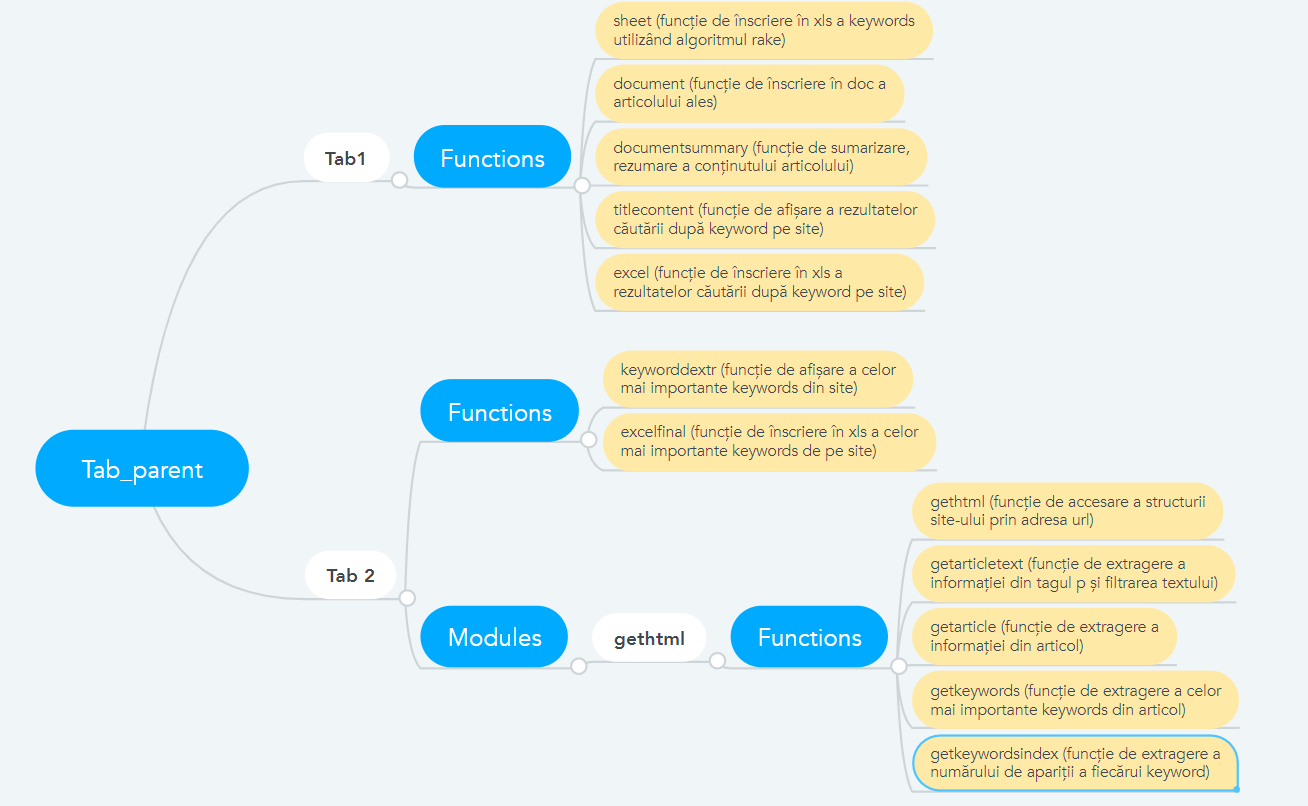
# Structura funcțională a aplicației



Aplicația este separată în două componente independente, una răspunde de căutarea informației dintr-un site anumit după un cuvânt cheie. După ce introducem cuvântul cheie noi vom avea ca rezultat o listă de articole unde putem alege 3 opțiuni pentru fiecare articol. Putem pur și simplu să extragem informația din el, putem să rezumăm informația și să extragem informația cu algortimul rake.

Al doilea component permite să extragem informația (cuvinte-cheie) din adresa site-ului pe care îl introducem.

# Algoritmul aplicației



În schema data este descrisă structura ei funcțională. Ea constă din 2 module grafice Tab1 și Tab2 și fiecare modul își are propriul set de funcții care vor fi utilizate.

# 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]

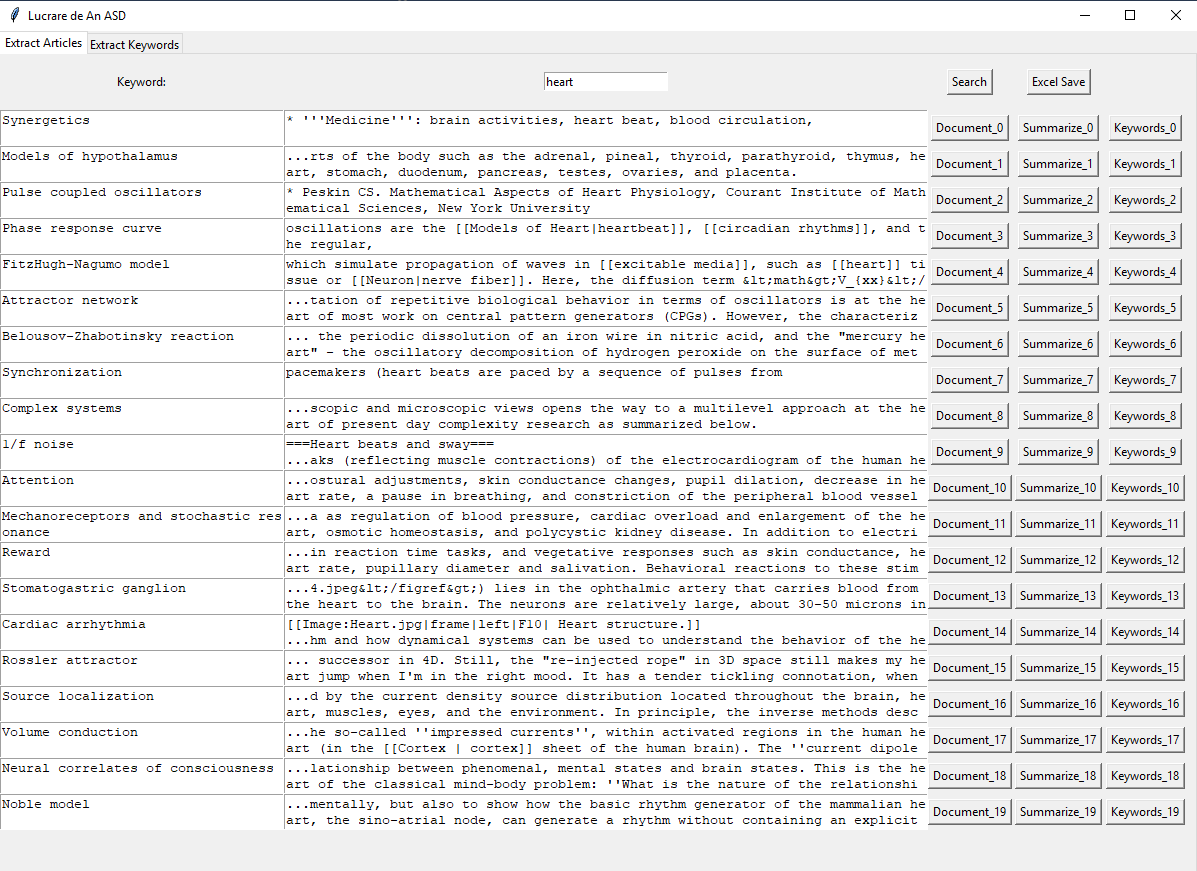
top30=[]

for w in top\_words:

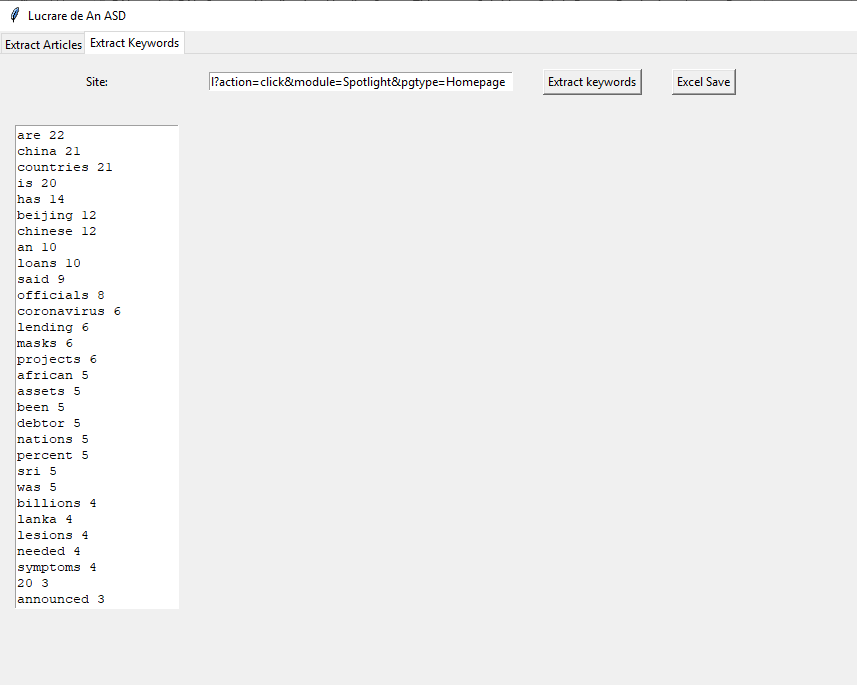
top30.append(w[1])

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.

# Concluzii

* Crearea softului de web scraping permite eficientizarea căutării și preluării datelor din rețaua globală.
* Noi putem face rezumatul unui masiv enom de date, putem extrage cuvintele cheie, putem să înscriem informația dorită direct în fișiere pentru ca să o păstram.
* Asemenea aplicații ușurează viața și permit la dezvoltarea unor domenii ca data-science deoarece noi putem să mărim volumul de date pe care îl posedăm, să-l structurăm și căutările ulterioare vor fi mai reușite.

# 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>
11. <https://habr.com/en/post/56461/>
12. <https://cyberleninka.ru/article/n/web-scraping-i-klassifikatsiya-tekstov-metodom-naivnogo-bayesa/viewer>
13. <http://www.cs.bilkent.edu.tr/~guvenir/courses/CS550/Seminar/freitag2000-ml.pdf>
14. <https://pypi.org/project/mechanize/>
15. <https://web.archive.org/web/20110902235619/http://aclweb.org/anthology-new/W/W08/W08-0208.pdf>
16. <https://pypi.org/project/rake-nltk/>
17. <https://pypi.org/project/python-docx/>
18. <https://docs.python.org/3/library/os.html>
19. <https://pypi.org/project/xlwt/>
20. <https://www.crummy.com/software/BeautifulSoup/#Download>
21. <https://www.w3schools.com/python/python_regex.asp>
22. <https://pypi.org/project/requests/>
23. <https://www.w3schools.com/python/module_requests.asp>
24. <https://www.tutorialspoint.com/python/python_gui_programming.htm>
25. <https://web.archive.org/web/20170427220310/http://www.digital-web.com/articles/the_document_object_model/>
26. <https://www.w3.org/TR/2012/CR-html5-20121217/>
27. <https://docs.python.org/3/tutorial/index.html>