



UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică
Aplicată

LUCRARE DE LICENȚĂ

COORDONATOR:
Conf. Dr. Pop Daniel

ABSOLVENT:
Oprica Andrei

TIMIȘOARA
2022

UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ: Informatică
Aplicată

**PROIECTAREA ȘI
IMPLEMENTAREA UNUI
SISTEM PENTRU
RECOMANDAREA DE
JURNALE/CONFERINȚE**

COORDONATOR:
Conf. Dr. Pop Daniel

ABSOLVENT:
Oprica Andrei

TIMIȘOARA
2022

Abstract

This project will consist of a website that aims to help members of the academic community that need help in finding a conference/journal in order to publish a scientific paper. The paper is required to have at least a title, an abstract description and some keywords. Through the means of this information along with machine learning algorithms implemented in the high-level language, Python, one or multiple suiting conferences or journals will be provided. Concretely, the web application will use the API technology that is going to access WikiCPF's (<http://www.wikicfp.com/cfp/>) database. This website offers a wide range of conferences and journals all around the world, in many domains and sub-domains. Each conference entry consists of the following information: a brief description about the event, title of the conference, categories, keywords, the date of the event, and a link to the official site of the host. My project will extract data from WikiCPF, after that it will analyze the title, abstract section and keywords introduced by the user. After that, by using machine learning algorithms (TF-IDF like algorithms and/or variants) built with Python, an appropriate recommendation list based on the user's criteria will be generated and precedingly shown on the front-end. A feature present in my project, will be the redirection to the conference's website where additional information about the event can be found, after the user chose a certain recommendation.

For example Wikicfp recommends many events, but doesn't have an advanced search, other solutions are websites be like:

Conferencealerts (<https://conferencealerts.com/>),

Resurchify(<https://www.resurchify.com/>);

And some others that recommend based on some search are:

JournalFinder(<https://journalfinder.elsevier.com/>),

JournalSuggerster (<https://journalsuggerster.springer.com/>).

All things considered, I chose to implement a solution that consists of extracting data from Wikicfp's database, through the use of an API, that combined with machine learning algorithms, will generate appropriate information.

Cuprins

Introducere	5
1 Prezentarea problematicii abordate	6
1.1 Descrierea problematicii	6
1.2 Abordări existente	7
1.2.1 JournalFinder	7
1.2.2 JournalSuggerer	8
2 Contribuții personale	9
3 Arhitectura aplicației	11
3.1 Descrierea arhitecturii	11
3.2 Descrierea cazurilor de utilizare	12
3.3 Tehnologiile utilizate	14
4 Facilitățiile aplicației	15
5 Detalii de implementare a sistemului	16
5.1 Pregătirea environment-ului pentru program	16
5.2 Pregătirea bazei de date și implementare script-ului pentru Web Scraping	17
5.3 Adaptarea algoritmului TF-IDF	19
5.4 Implementarea rutelor pentru recomandare pe baza informațiilor oferite de utilizator în partea de back-end	21
5.5 Implementarea unei rute ce doar oferă informații în partea de back-end	23
5.6 Prezentarea design-ului aplicației	25
6 Concluzii și direcții viitoare	29

Introducere

În această lucrare mi-am propus să dezvolt un sistem de recomandare de conferințe și jurnale. Acest sistem va fi practic un site web cu mai multe pagini în care sunt afișate informațiile despre conferințe și jurnale, dar oferă posibilitatea ca utilizatorul să introducă un abstract pe baza căruia se vor căuta potrivirea conferințelor și a jurnalelor. Datele despre jurnale și conferințe trebuie să fie importate și reactualizate constant într-o bază de date și cu ajutorul unor algoritmi de machine learning (spre exemplu TF-IDF [1]) se vor recomanda în funcție de preferințele utilizatorului. Pentru importarea datelor vreau să fac web scraping la site-ul WikiCFP (<http://www.wikicfp.com/cfp/>) unde se pot găsi foarte multe conferințe și jurnale și informații despre acestea, cum ar fi data limită pentru depunerea lucrăriilor științifice, data la care va avea loc, siteul oficial al conferinței/jurnalului unde se pot preda lucrările, cuvinte cheie pentru descrierea temei conferinței/jurnalului.

Algoritmii de machine learning, precum și algoritmii de web scraping se găsesc în librării de python, iar pentru partea de back-end a siteului vreau să utilizez microframeworkul flask [2], deoarece și acesta se găseste în python.

Cu acest sistem îmi propun să ajut persoanele din domeniul academic să găsească cât mai ușor conferințele și jurnalele la care doresc să participe și să își depună lucrăriile științifice.

Capitolul 1

Prezentarea problematicii abordate

1.1 Descrierea problematicii

În crearea acestui sistem sunt câteva probleme (puncte de implementat). Acestea ar fi următoarele: creearea unui script în python ce furnizează datele și salvarea datelor, implementarea sistemului bazat pe web ce va putea lua date de la utilizator pentru a îi recomanda un jurnal sau o conferință utilă.

Pentru crearea scriptului în python ce extrage date de pe siteul WikiCFP voi utiliza librăria BeautifulSoup [3] ce se folosește destul de des în web scraping. Apoi cu utilizarea anumitor funcții ce folosesc regex [4] voi elimina tagurile paginii de HTML și îmi voi furniza doar datele utile, acestea find link-uri către paginile fiecărui jurnal sau fiecărei conferințe, iar cu aceste link-uri pot extrage datele fiecărei pagini din care pot salva într-o bază de date, înregistrările necesare (spre exemplu: denumirea conferinței/jurnalului, un link către site-ul oficial al jurnalului/conferinței, categoriile acestora, când și unde se vor desfășura, termene limită pentru depunerea lucrăriilor științifice sau pentru aplicarea la acestea, descrierea sub forma de abstract).

Crearea sistemului se poate face cu microframework-ul flask [2] ce ajută foarte mult în a implementa ușor partea de back-end a siteului. Astfel momentan trecând cu vederea crearea unui design sistemul va trebui să poată colecta datele din baza de date [5] informațiile utile, urmând ca dacă sunt mai multe înregistrări să poată face o clasificare a acestora și să îi afișeze utilizatorului câteva jurnale/conferințe clasificate cu ajutorul unui algoritm de machine learning (TF-IDF [1]). Apoi urmează ca utilizatorul să aleagă una dintre recomandările primite și să îi apară informațiile utile despre acea înregistrare.

Și în final trebuie creată o interfață elegantă pentru acest sistem. Pentru care momentan mă gândesc să utilizez CSS alături de HTML.

1.2 Abordări existente

Am vazut câteva abordari similare. Similaritatea este dată de funcționalitate. Spre exemplu site-urile JournalFinder(<https://journalfinder.elsevier.com/>) și JournalSuggerer (<https://journalsuggerer.springer.com/>).

La aceste două site-uri am observat că se folosesc de informații introduse de către utilizator pentru a îi recomanda informații. Nu știu cum sunt implementate concret, însă diferența față de ceea ce mi-am propus să implementez este că acestea folosesc javascript. Nu știu ce au mai folosit cei ce au implementat aceste site-uri în afară de HTML și CSS. Nu stiu de unde aceste site-uri își furnizează datele și pe baza a ce algoritmi recomandă jurnale/conferințe. De asemenea site-ul meu va recomanda direct informațiile fără să aștepte ca utilizatorul să introducă date.

1.2.1 JournalFinder

Acest site este utilizabil în căutarea jurnalelor. Pentru a căuta are nevoie să îi se furnizeze date precum:

- titlul lucrării;
- cuvinte cheie;
- domeniul de căutare.

Însă aceste informații sunt minimul necesar, încrucișându-se poate face o căutare mai detaliată dacă i se oferă date precum:

- tipul de publicare;
- impactul jurnalului;
- timpul publicației.

După ce se apasă butonul de căutare de jurnale, site-ul crează o pagină în care sunt afișate jurnalele ce se potrivesc cu datele introduse de către utilizator. Apoi se poate alege un jurnal și să apar mai multe detalii despre acesta. Dacă se apasă butonul pentru a afla mai multe despre respectivul jurnal ales, site-ul redirecționează utilizatorul pe o pagină în care apar date doar despre jurnalul respectiv.

1.2.2 JournalSuggester

Asemănător cu site-ul precedent și acesta are nevoie de date ca:

- un titlu;
- un text (acest text are rolul unui abstract, dar nu te obligă să scri un abstract întreg, deoarece funcționează și dacă este introdus doar un cuvânt, dar cred că acest lucru poate influența rezultatele);
- un domeniu de căutare.

Și acest site permite o căutare mai detaliată pentru care are nevoie de informații precum:

- factorul minim de impact;
- nota minima de acceptare;
- timpul maxim până la prima decizie;
- serviciul de indexare.

În cazul acestui site, când se apasă butonul de căutare, pagina afișează sub acel buton o listă cu titlul și cîteva informații legate de factorul de impact, prima decizie și nota de acceptare. Dacă un astfel de rezultat este accesat site-ul redirecționează utilizatorul către o pagină în care se aduc informații suplimentare spre exemplu:

- un link ce duce către site-ul jurnalului;
- publicul țintă;
- scopul jurnalului;
- subcategoriile de interes.

Capitolul 2

Contribuții personale

Contribuțiile mele personale le-aș putea împărții în două părți pentru că aplicația are în principiu două părți:

- prima parte este cea de furnizare de date și popularea bazei de date;
- a doua parte este aceea de funcționalitate a sistemului, ce extrage date, le ordonează și le afișează.

În partea de furnizare de date și popularea bazei de date am folosit librăriile BeautifulSoup și requests pentru a obține informațiile, dar a trebuit să studiez cum apar informațiile pe site-ul WikiCFP pentru a putea extrage prin expresii regex doar datele de care am nevoie și să încerc să automatizez procesul deoarece procesul are două părți:

- în prima fază script-ul trebuie să aibe o listă cu link-uri de pagini ce au mai multe link-uri către paginile ce au informație strictă pentru fiecare înregistrare din tabel;
- în cea de a doua parte script-ul folosește link-urile pentru a extrage informațiile și le introduce în baza de date.

Acum o problemă a script-ului este că trebuie rulat de mai multe ori, deoarece folosesc o structură de tip listă și poate da erori dacă sunt introduse mai mult de 15 link-uri ce fiecare furnizează căte 20 de înregistrări. O altă limitare a script-ului este aceea că nu poate extrage datele pentru a fi de același fel și de aceea este nevoie ca unele înregistrări să fie șterse prin comenzi de sql, astfel ne rezultă o bază de date relațională. De asemenea aceste operații durează pentru că nu am reușit să complexități foarte bune, deoarece am o complexitate de $O(n^2)$ la puterea a două.

În a doua parte, adică cea de extragere, procesare și afișare a recomandărilor am conceput și implementat cum să structurez datele pentru a le transmite către template și afișarea acestora. Pentru partea de stilizare am folosit doar CSS și contribuția a fost aceea de a implementa design-ul. Practic am pus o imagine pe fundal, tabelul în care apar recomandările l-am colorat puțin și am stilizat bara de rulare și am mai stilizat bara de navigație.

În vederea funcționării sistemului, momentan o pagină are doar un număr de 200 de înregistrări pe categorie, este funcționabil și pentru mai multe înregistrări, dar este posibil ca dacă ar fi scalat foarte mult, fiind vorba de peste câteva mii de înregistrări (spre exemplu 10000) sunt destul de convins că vă există probleme de timp pentru că utilizați o sortare prin selecție, astfel are o complexitate de $O(n \log n)$ înmulțit cu logaritm din n . De asemenea sistemul este rulat pe localhost, nu sunt sigur că ar face față unui număr foarte mare de utilizatori.

Capitolul 3

Arhitectura aplicației

3.1 Descrierea arhitecturii

Stilul arhitectural: Client-Server

Motivarea alegerii acestui stil:

1. Problema: Sistemul poate fi utilizat de mai mulți utilizatori deodata, acesta ofera un serviciu utilizatorului, anume acela de ai recomanda conferințe și jurnale pe baza unui titlu, un abstract și cuvinte cheie.
2. Context: Stilul arhitectural Client-Server, ofera posibilitatea utilizatorului de a cere anumite informații de la server, care acesta din urmă îl va oferi. Am ales acest stil arhitectural deoarece ofer doar un serviciu, spre exemplu dacă aș fi avut mai multe servicii aș fi ales stilul arhitectural microservicii.
3. Soluție:
 - Modelul sistemului: în partea de back-end a sistemului se vor importa și reactualiza date cu referință la informațiile necesare pentru conferințe și jurnale; în frontend se pot introduce datele după care sunt căutate conferințele și jurnalele, iar cu datele introduse, se vor căuta conferințele și jurnalele din baza de date cu ajutorul unor algoritmi de machine learning;
 - Componente: informațiile despre conferințe și jurnale;
 - Conectori: după căutarea datelor, acestea vor fi afișate pe o pagină creată cu ajutorul template-urilor puse la dispoziție de microframework-ul flask.
4. Variante: Limbajul de programare python permite multe facilități în colectarea, aranjarea și afișarea datelor. Spre exemplu python are microframework-ul flask ce ajută la back-end; librării cu algoritmi de machine learning; și spre exemplu librăria BeautifulSoup ce ajută la web scraping-ul datelor.

Astfel voi avea o bază de date ce are înregistrări despre jurnale și conferințe. Utilizatorul, care vrea să obțină informații despre anumite jurnale sau conferințe trebuie să acceseze pagina dedicată categoriei de care este interesat, de asemenea utilizatorul poate introduce un abstract și poate selecta o categorie pentru a se face o căutare

mai amănunțită ce îi poate ușura mult căutarea. Serverul interoghează baza de date, clasifică datele și le afișează utilizatorului.

Mai jos voi afișa o imagine pentru a exemplifica schița arhitecturii.

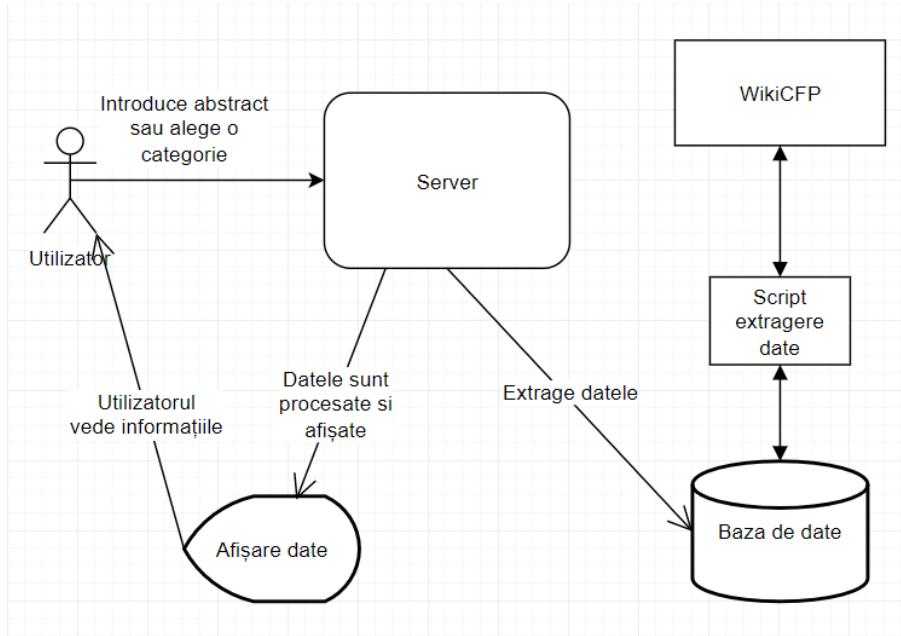


Figura 3.1: Schița Arhitecturii Sistemului

3.2 Descrierea cazurilor de utilizare

În practică sunt două cazuri de utilizare și anume acela în care utilizatorul accesează una dintre paginile oferite de server, iar serverul îi recomandă conferințele și/sau jurnalele potrivite pe baza paginii ce a fost selectată; și acela în care utilizatorul introduce un abstract și selectează o categorie, iar pe baza potrivirii abstractului utilizatorului cu abstractul conferinței sau a jurnalului sunt recomandate înregistrările.

Astfel dacă utilizatorul introduce abstractul și selectează categoria și apasă butonul "Submit", va fi redirectionat pe o pagină cu recomandări făcute special pentru informația ce a introdus-o. Din acest punct el poate să navigheze înapoi la pagina de start sau să acceseze una dintre paginile ce sunt explicitate în cele ce urmează.

Spre exemplu dacă utilizatorul accesează pagina ce are ca titlu "Computer Science", serverul va redirectiona utilizatorul către pagina ce extrage date cu categoria respectivă, ordonează informațiile cu algoritmul de TF-IDF și afișează un tabel cu datele ordonate.

Mai jos voi afișa o imagine pentru a exemplifica cazurile de utilizare.

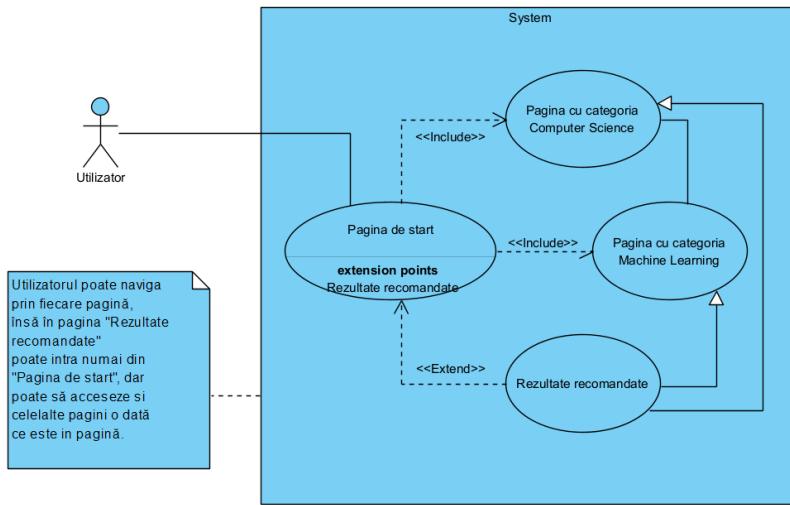


Figura 3.2: Diagrama cazurilor de utilizare

Astfel în momentul în care utilizatorul accesază site-ul se află la pagina de start. În acest punct el va putea alege să acceseze pagina cu conferințe și jurnale legate de una dintre următoarele categorii:

- Computer Science;
- Machine Learning.

Când una dintre aceste pagini este accesată este afișat un tabel cu mai multe înregistrări ale fiecare înregistrare a tabelului are urmatoarele informații:

- un titlu;
- categoriile și subcategoriile;
- o descrierea (un abstract);
- un link către pagina principală a conferinței/jurnalului;
- mai multe date reprezentând când, unde se organizează conferințele/jurnalele.

Astfel cazul de utilizare al aplicației va fi acela de a recomanda conferințe/jurnale fără a primi informații de la utilizator. Pentru acest caz de utilizare se vor folosi datele extrase din baza de date. Aceste date trebuie sortate prin algoritmul TF-IDF, care va calcula un scor pentru fiecare dintre conferințe/jurnale, după care acestea vor fi sortate în ordine descrescătoare a scorului.

3.3 Tehnologiile utilizate

Pentru început pot spune că am folosit ca și editor PyCharm (<https://www.jetbrains.com/pycharm/>) de la JetBrains, am utilizat acest editor pentru că este foarte capabil și intuitiv. Acesta permite interacțiunea cu baza de date, ceea ce îmi este foarte de folos din moment ce datele extrase de pe WikiCFP nu vin întotdeauna în formatul dorit, astfel pot deschide o consolă de interacțiune cu baza de date și să elimin datele ce nu se încadrează în formatul ce îl doresc.

Pentru a avea back-up-uri la sistem folosesc GitHub (<https://github.com/>). Acesta poate fi foarte ușor de utilizat în combinație cu Pycharm. Astfel folosesc GitHub pentru a versiona sistemul de recomandare de conferințe și jurnale; și pentru a putea fi accesat ușor codul sursă al sistemului.

Pentru realizarea acestui sistem de recomandare de conferințe/jurnale am nevoie de câteva tehnologii. Am nevoie de o baza de date în care voi stoca informațiile obținute prin web scraping de pe WikiCFP. Am nevoie de un server ce mă va ajuta să genționez partea de back-end al aplicației și de o interfață web pe care să afișez datele procesate (sortate pe baza scorului obținut cu ajutorul algoritmului de TF-IDF) sub formă de tabel.

Astfel voi avea nevoie de următoarele tehnologii:

- Python, limbajul de programare îmi oferă librării și microframework-uri foarte utile, ca spre exemplu:
 - Flask, acest microframework mă va ajuta să gestionez datele extrase și introduse în baza de date. Va avea un rol de server;
 - librăriile requests și BeautifulSoup, mă ajută să scriu script-uri prin care pot face web scraping.
- Sqlite, este o bază de date foarte rapidă pentru un trafic mic-mediu și este perfectă pentru ceea ce îmi propun să dezvolt în această aplicație;
- HTML și CSS, cele mai ușoare metode prin care voi face interfața aplicației (partea de front-end);

Capitolul 4

Facilitățiile aplicației

O facilitate foarte importantă pentru acest program este utilizarea algoritmului de TF-IDF pentru a obține câte un scor pentru fiecare conferință sau fiecare jurnal ce se află în baza de date. Pe baza scorurilor obținute sortez ordinea de afișare a înregistrăriilor din baza de date. TF-IDF vine de la “Term Frequency - Inverse Data Frequency”, astfel algoritmul poate fi împărțit în trei părți. În cele ce urmează voi explica cele trei părți ale algoritmului din punct de vedere matematic:

- Prima parte este TF, această funcție ne furnizează frecvența cuvintelor într-un document. Partea ce influențează în această etapă este împărțirea apariției unui cuvânt la totalul de cuvinte din document.
- A doua parte este IDF, această parte calculează influența cuvintelor rare din document. Astfel cuvintele rare obțin un scor mare în IDF. Aceasta se calculează ca logaritm din numărul total de documente împărțit la numarul de documente ce conțin un anumit cuvânt.
- A treia parte, respectiv TF-IDF reprezintă înmulțirea dintre TF și IDF.

O altă facilitate a aplicației este cea de colectare a datelor prin web scraping cu ajutorul librăriei BeautifulSoup, requests și regex pentru a colecta, edita datele, iar cu sqlite3 salvez datele obținute anterior într-o baza de date. Totuși datele nu sunt întotdeauna în felul dorit pentru a fi o bază de date relațională, astfel în continuare folosesc două comenzi de sql în consolă pentru a elimina înregistrările ce nu se potrivesc.

Aceste două facilități prezentate anterior nu se observă, deoarece reprezintă funcționalități de back-end ale aplicației.

O a treia facilitate ce este observabilă de utilizator este recomandarea de conferințe și jurnale. Aceastea sunt afișate într-un mod stilizat într-un tabel în funcție de categoria principală pe care o dorește utilizatorul, acesta poate să acceseze categoria de ”computer science” sau de ”machine learning”, acestea fiind categoriile din care am obținut date.

O a treia facilitate ce este observabilă de utilizator este chiar funcționalitatea principală a aplicației, adică aceea de a da posibilitatea utilizatorului să introducă un abstract și să selecteze o categorie pe baza cărora să îi furnizeze recomandări de conferințe și jurnale al căror abstract se potrivește cel mai bine cu abstractul oferit de utilizator.

Capitolul 5

Detalii de implementare a sistemului

Programul implementat complet se găsește în următorul repository de github la următoarea adresă: <https://github.com/AndreiOprica/AppLicenta>

5.1 Pregătirea environment-ului pentru program

Pentru editor și compilator am utilizat Pycharm de la jetbrains. Am creat un proiect de tip Flask și am folosit versiunea 3.10 de python. Apoi trebuie deschis un terminal în Pycharm și trebuie instalate librăriile:

- librăria BeautifulSoup (cu comanda: "pip install beautifulsoup4");
- librăria requests (cu comanda: "pip install requests");
- librăria sqlite (cu comanda: "pip install sqlite");
- librăria copy (cu comanda: "pip install copy").

Acstea librării sunt necesare pentru următoarele utilizări:

- librăria BeautifulSoup pentru a face web scraping peste WikiCFP și a extrage datele necesare;
- librăria requests pentru a da cereri de tip GET către WikiCFP;
- librăria sqlite pentru a introduce înregistrări în baza de date în faza de web scraping și pentru a extrage înregistrările în variabile pentru a fi procesatice și afișate;
- librăria copy pentru a copia înregistrările și a le pune liste cu înregistrările într-un dicționar fără a avea probleme în structura obiectelor.

Acestea sunt librăriile utilizate ce trebuie și instalate și necesitatea lor. În sistem mai folosesc libraria re pentru expresii regex, ca să pot să elimin tag-urile de HTML și toate expresiile din pagini ce nu sunt folosite pentru sistem.

5.2 Pregătirea bazei de date și implementare script-ului pentru Web Scraping

Pentru a crea baza de date și a o lega cu programul trebuie creat un fișier în Pycharm, de tipul sqlite (spre exemplu "identifier.sqlite"). Apoi trebuie create tabele, aici sunt două variante:

- cu ajutorul tool-ului din Pycharm;
- cu o comandă de sql.

Pentru a crea tabelele în consolă, trebuie deschisă o consolă și rulat următoarea comandă, cu excepția că trebuie schimbată denumirea tabelului, și trebuie create câte pagini pe diferite categorii se doresc. De asemenea folosesc tipul text la toate coloanele pentru că în python lucrez cu string-uri, iar listele în care sunt salvate datele sunt de diferite lungimi, urmând ca după ce sunt adăugate datele, le vom elimina pe cele de lungimi diferite cu două comenzi de sql. Comanda de creare a tabelului este:

Sunt necesare douăzeci de coloane în tabel, deoarece există îregistriările ce au mai multe sau mai puține coloane, iar pentru a mă asigura că nu primesc erori, am considerat că este mai bine să pun mai multe coloane.

În implementarea script-ului pentru web scraping am creat câteva funcții. Pentru început trebuie importate librăriile BeautifulSoup, requests, re și sqlite3. Apoi am creat două modele (pattern-uri), unul pentru a recunoaște link-urile către paginile cu ce au informația necesară, iar celălalt pentru a șterge tag-urile din paginile cu informații. Apoi am făcut două funcții ce folosesc pattern-urile de regex în scopurile menționate anterior.

După această etapă am creat două funcții ce au următoarele sarcini:

- prima funcție face un request de get către url-ul ce îl primește ca argument, face parsing la pagină, caută toate tag-urile de tip ”jaξ” și folosește funcția ce extrage link-urile, le formatează și le returnează ca listă;
- a doua funcție face request către linkul ce îl are ca și argument și formatează informația și o retunează sub forma de listă.

Această parte de implementare arată astfel:

```
links_patterns = re.compile(r'/servlet/event.showcfp?\S*')
remove_tags_pattern = re.compile(r'<[^>]+>')

def find_links(text):
    return re.findall(links_patterns, text)

def remove_tags(text):
    return remove_tags_pattern.sub(' ', text)
```

```

def scrap_pages_links(uri):
    req = requests.get(uri)
    soup = BeautifulSoup(req.text, 'html.parser')
    find = soup.findAll('a')
    text = find_links(str(find))
    for e in range(len(text)):
        one_string = str(text[e])
        formated = one_string.split('\n')
        text[e] = 'http://www.wikicfp.com/cfp' + formated[0]
    return text

def scrap_page(uri):
    req = requests.get(uri)
    soup = BeautifulSoup(req.text, 'html.parser')
    text = soup.findAll('div', {"class": "contsec"})
    text_without_tags = remove_tags(str(text))
    edited_data = text_without_tags.split("\n")
    edited_data = [element for element in edited_data
                  if element]
    for element in range(len(edited_data)):
        edited_data[element] = edited_data[element].strip()
    no = -1
    for element in range(len(edited_data)):
        if edited_data[element] == 'Related_Resources':
            no = element - 1
            break
    if no != -1:
        del edited_data[no:len(edited_data)]
    del edited_data[0:3]
    return edited_data

```

Acum trebuie utilizată o listă cu mai multe pagini ce au fiecare căte douăzeci de link-uri (voi da un exemplu ce conține două astfel de link-uri), apoi se apelează funcțiile pe rând și este creată o listă cu liste în formatul ce îl are baza de date creată anterior. Această secvență arată astfel:

```

urls = [
    'http://www.wikicfp.com/cfp/call?conference=computer
     %20science&page=1',
    'http://www.wikicfp.com/cfp/call?conference=computer
     %20science&page=2'
]
data = []
for url in urls:
    links_from_url = scrap_pages_links(url)
    for el in links_from_url:
        element = []
        scraped_data = scrap_page(el)

```

```

for ele in scraped_data:
    if ele != '':
        element.append(ele)
rows = 20 - len(element)
for ele in range(rows):
    element.append(' ')
data.append(element)

```

Apoi trebuie să ne conectăm la baza de date, să adăugăm datele (denumirea tabelului trebuie să fie cea folosită la crearea tabelului) și în final să închidem baza de date:

```

conn = sqlite3.connect('identifier.sqlite')
c = conn.cursor()
c.executemany('INSERT INTO table_name
VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);', data)
conn.commit()
conn.close()

```

Iar în final pentru a ne asigura că datele introduse în tabel sunt în formatul pe care îl dorim trebuie să actualizăm baza de date cu ajutorul unor comenzi în sql pentru a ne asigura că informațiile din tabel sunt în formatul pe care îl doresc.

5.3 Adaptarea algoritmului TF-IDF

În adaptarea algoritmului de TF-IDF am avut de recreat trei funcții destul de simple, dar ce au impactul de a-mi asigura ordinea afișării conferințelor și a jurnalelor. În ceea ce urmează voi exemplifica funcțiile și le voi explica. Prima funcție este cea de TF:

```

def computetf(word_dict, bow):
    tf_dict = {}
    bow_count = len(bow)

    for word, count in word_dict.items():
        tf_dict[word] = count/float(bow_count)

    return tf_dict

```

Funcția computetf primește un dicționar cu frecvența cuvintelor, și totalul de cuvinte. Acesta împarte numărul de apariții a unui cuvânt la totalul de cuvinte ce se găsesc și le pune în dicționar la cheia cuvântului. Apoi returnează dicționarul. După urmează funcția de IDF, ce arată astfel:

```

def computeidf(doc_list):
    import math

    n = len(doc_list)

```

```

idf_dict = dict.fromkeys( doc_list[0].keys() , 0)
for doc in doc_list:
    for word, val in doc.items():
        if val > 0:
            idf_dict[word] += 1

for word, val in idf_dict.items():
    idf_dict[word] = math.log10(n / float(val))

return idf_dict

```

În funcția computeidf se primește o listă cu două rezultate a funcției computetf și pentru fiecare cuvânt din listă se face calcul de logaritm de apariții ale unui cuvânt pe numărul total de cuvinte și se adaugă în dicționar, acesta urmând să fie returnat mai departe. După această etapă urmează calcularea scorului cu combinarea funcțiilor și rezultarea funcției TF-IDF

```

def computetfidf(tf_bow, idfs):
    tfidf = {}

    for word, val in tf_bow.items():
        tfidf[word] = val * idfs[word]

    score = 0
    no = 0

    for word, value in tfidf.items():
        score += value
        no += 1

    return score / no

```

Funcția computetfidf primește ca argumente rezultatele de la cele două funcții anterioare, înmulțește valorile celor două rezultate ale funcțiilor anterioare pe fiecare cuvânt, apoi face un scor pentru fiecare document adunând valorile și la final împărțiindu-le la totalul de cuvinte (fiecare cuvânt este numărat o data și cuvintele sunt distințe).

5.4 Implementarea rutelor pentru recomandare pe baza informațiilor oferite de utilizator în partea de back-end

Pentru a implementa căutarea am nevoie de două rute. Prima este chiar pagina de start, ce în cazul în care primește metoda "POST" extrage abstractul introdus de utilizator și redirecționează utilizatorul către pagina cu rezultate. Altfel este afișată doar pagina de start. Această implementare arată astfel:

```
@app.route('/', methods=['GET', 'POST'])
def start_page():
    if request.method == 'POST':
        abstract = request.form["abstract"]
        return redirect(url_for('result', abstract=abstract))

    return render_template('index.html')
```

Pentru a crea pagina cu rezultate a trebuit să extrag form-ul cu abstractul introdus de utilizator și opțiunea selectată. Apoi m-am conectat la baza de date și am ales tabelul pe care în interoghez în funcție de opțiunea ce a fost selectată de utilizator. După am adăugat informațiile în mai multe liste. Această parte arată astfel:

```
@app.route("/result", methods=['GET', 'POST'])
def result():
    abstract = request.form.get('abstract')
    option = request.form.getlist('options')

    conn = sqlite3.connect('identifier.sqlite')
    cur = conn.cursor()

    if option[0] == 'option1':
        cur.execute('SELECT * FROM ComputerScience')
        print(option[0])
    elif option[0] == 'option2':
        cur.execute('SELECT * FROM MachineLearning')
        print(option[0])

    c0 = []
    c1 = []
    c2 = []
    c3 = []
    c4 = []
    c5 = []
    c6 = []
    c7 = []
    c8 = []
    c9 = []

    for row in cur:
```

```

c1.append(row[0])
c2.append(row[1])
c3.append(row[3])
c4.append(row[5])
c5.append(row[7])
c6.append(row[9])
c7.append(row[11])
c8.append(row[12])
c9.append(row[14])

```

Apoi am calculat un scor cu TF-IDF cu abstractul introdus de utilizator și abstractele fiecarei conferințe sau fiecarui jurnal. Această secvență arată astfel:

```

bow_abstract = abstract.split(" ")
for i in range(len(c8)):
    bow_description = c9[i].split(" ")
word_set = set(bow_abstract).union(set(bow_description))
word_dict_abstract = dict.fromkeys(word_set, 0)
word_dict_description = dict.fromkeys(word_set, 0)

for word in bow_abstract:
    word_dict_abstract[word] += 1

for word in bow_description:
    word_dict_description[word] += 1

tf_bow_abstract = compute_tf(word_dict_abstract,
                             bow_abstract)
tf_bow_description = compute_tf(word_dict_description,
                                 bow_description)

idfs = compute_idf([word_dict_abstract,
                     word_dict_description])

tfidf_bow_category = compute_tfidf(tf_bow_abstract,
                                    idfs)
tfidf_bow_description = compute_tfidf(tf_bow_description,
                                       idfs)

c0.append((tfidf_bow_category + tfidf_bow_description))

```

Apoi am ordonat conferințele și jurnalele extrase din baza de date pe baza scorului obținut anterior. După am creat un dicționar cu primele 10 înregistrări ce au scorul cel mai mare și le-am transmis către template. Această secvență arată în felul următor:

```

for i in range(len(c0)):
    max_score = i

```

```

for j in range(i+1, len(c0)):
    if c0[max_score] < c0[j]:
        max_score = j

    c0[i], c0[max_score] = c0[max_score], c0[i]
    c1[i], c1[max_score] = c1[max_score], c1[i]
    c2[i], c2[max_score] = c2[max_score], c2[i]
    c3[i], c3[max_score] = c3[max_score], c3[i]
    c4[i], c4[max_score] = c4[max_score], c4[i]
    c5[i], c5[max_score] = c5[max_score], c5[i]
    c6[i], c6[max_score] = c6[max_score], c6[i]
    c7[i], c7[max_score] = c7[max_score], c7[i]
    c8[i], c8[max_score] = c8[max_score], c8[i]
    c9[i], c9[max_score] = c9[max_score], c9[i]

# put data in a dictionary and return data to a template
output = {}
lengtht = 10
if len(c1) < lengtht:
    lengtht = len(c1)
for i in range(lengtht):
    lst = [c0[i], c1[i], c8[i], c9[i], c2[i], c3[i],
           c4[i], c5[i], c6[i], c7[i]]
    output[i] = copy.copy(lst)

return render_template('result.html', output=output.items())

```

Astfel este implementată partea de back-end a rutelor de recomandare pentru abstractul introdus de utilizator.

5.5 Implementarea unei rute ce doar oferă informații în partea de back-end

Pentru început voi exemplifica cum se definește o simplă funcție ce randează un template. Această funcție este ruta ce afișează o simplă pagină de start.

```

@app.route('/')
def start_page():
    return render_template('index.html')

```

În același mod se scrie o funcție în care trebuie schimbată ruta (spre exemplu din '/' în '/ComputerScience') și denumim la fel și funcția. După în această funcție trebuie să ne conectăm la baza de date și să extragem toate datele din tabel (spre exemplu tabelul ComputerScience ce este creat și are informații), urmând să creăm obiecte (liste) în care urmează să punem înregistrările extrase din tabel. Mai trebuie o listă goale în care se vor adăuga scorurile obținute. Această parte va fi așa:

```
conn = sqlite3.connect('identifier.sqlite')
```

```

cur = conn.cursor()
cur.execute('SELECT * FROM ComputerScience')

c0 = []
c1 = []
c2 = []
c3 = []
c4 = []
c5 = []
c6 = []
c7 = []
c8 = []
c9 = []

```

```

for row in cur:
    c1.append(row[0])
    c2.append(row[1])
    c3.append(row[3])
    c4.append(row[5])
    c5.append(row[7])
    c6.append(row[9])
    c7.append(row[11])
    c8.append(row[12])
    c9.append(row[14])

```

După ce am salvat înregistrările în liste pe coloane, am aplicat algoritmul de TF-IDF asupra lor, fiecare indice al listelor reprezintă o înregistrare din tabel. Aplicarea algoritmului de TF-IDF supra înregistrărilor arată astfel:

```

for i in range(len(c8)):
    bow_category = c8[i].split(" ")
    bow_description = c9[i].split(" ")
    word_set = set(bow_category).union(set(bow_description))
    word_dict_category = dict.fromkeys(word_set, 0)
    word_dict_description = dict.fromkeys(word_set, 0)

    for word in bow_category:
        word_dict_category[word] += 1

    for word in bow_description:
        word_dict_description[word] += 1

    tf_bow_category = compute_tf(word_dict_category,
                                 bow_category)
    tf_bow_description = compute_tf(word_dict_description,
                                    bow_description)

    idfs = compute_idf([word_dict_category,
                        word_dict_description])

```

```

tfidf_bow_category = compute_tfidf(tf_bow_category ,
                                    idfs)
tfidf_bow_description = compute_tfidf(tf_bow_description ,
                                       idfs)
c0.append((tfidf_bow_category + tfidf_bow_description))

```

După ce am aplicat algoritmul și am salvat scorul obținut de algoritm, am făcut o simplă ordonare prin selecție a tuturor listelor pe baza scorului obținut la TF-IDF. După tot ce a mai trebuit făcut este crearea unui dicționar, utilizarea unei structuri repetitive pe indicii unei liste (toate au aceeași lungime), creearea unei liste în structura repetitivă și copierea listei în dicționar cu cheia egală cu indicele listelor. Am folosit astfel formarea dicționarului ce trebuie transmis la template pentru a putea afișa înregistrările pe linii și nu pe coloane. Această parte arată astfel:

```

output = {}
for i in range(len(c1)):
    lst = [c1[i], c8[i], c9[i], c2[i], c3[i], c4[i], c5[i],
           c6[i], c7[i]]
    output[i] = copy.copy(lst)

return render_template('computerScience.html',
                      output=output.items())

```

Astfel se transmite dicționarul cu template-ul 'computerScience.html'. Implementarea pagini ce recomandă conferințe și jurnale cu categoria Machine Learning se face în același format, tot ce trebuie editat este extragerea datelor din tabelul bazei de date ce conține conferințele și jurnalele dorite; și mai trebuie să schimbe template-ul.

5.6 Prezentarea design-ului aplicației

În cele din urmă aici am folosit doar HTML și CSS. Am avut două tipuri de pagini:

- primul tip este cel de start; am o singură pagină de acest fel și am utilizat-o pentru a descrie utilitatea sitului de recomandare, pe cine ar ajuta; pentru a colecta abstractul și opțiunea introdusă de către utilizator și să redirecționeze către pagina cu rezultate;
- în al doilea tip am două pagini ce sunt în esență la fel, ca și implementare, ceea ce diferă între ele este faptul că primesc date (conferințe și jurnale) din tabele diferite, deci datele sunt diferite în funcție de categorie; și pagina ce este accesată din pagina de start după ce utilizatorul a introdus datele necesare, în această pagină sunt afișate maxim 10 înregistrări ce se potrivesc cel mai bine cu datele introduse de către utilizator, acestea fiind afișate în aceeași modalitate ca și cele două pagini, cu diferența că se extrage opțiunea și abstractul utilizatorului și sunt procesate și afișate doar 10 recomandări.

Partea cea mai relevantă din al doilea tip de pagini este afișarea datelor. Astfel am făcut un table ce are "thead" pentru antet și "tbody" pentru datele furnizate.

Aceste date vin sub forma unui dicționar, astfel le parcurg cu o structură repetitivă de tip ”for” pe chei și valori, afișez cheia, apoi mai folosesc o structură repetitivă de același fel pentru a parcurge valorile cheii și a le afișa.

În toate tipurile de pagini am o bară de navigație pentru a accesa celelalte pagini. În cazul în care s-ar adăuga pagini noi pentru a recomanda și alte categorii, tot ce ar trebui editat ar fi stilizarea în css a bării de navigație și adăugarea noilor butoane ce redirecționează către pagina dorită.

Astfel după cum se poate observa în următoarea poză, se observă prima pagină (pagina de casă) care conține bara de navigație pentru a accesa cu ușurință celelalte pagini. De asemenea am pus un text în care am scris o scurtă descriere a sistemului, ce face și pentru cine este destinat și loc pentru a facilita utilizatorul să introducă informațiile lui.

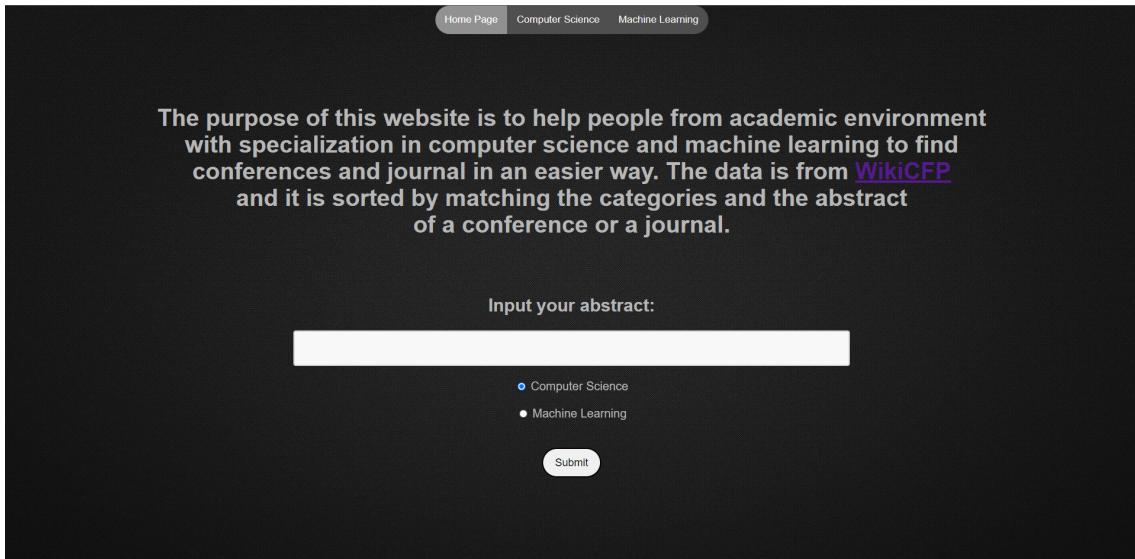


Figura 5.1: Pagina de start a sistemului

După ce sunt introduse datele și este apasat butonul ”Submit”, utilizatorul este redirecționat către pagina cu rezultate. Am două exemple ale acestei pagini de rezultate sunt:

- Abstractul lucrării mele de licență cu optiunea de Computer Science:

Home Page	Computer Science	Machine Learning				
Here are the first 10 results based on the input abstract and the selected category						
no.	SCORE	TITLE	CATEGORIES	ABSTRACT	LINK	W1
1	0.0016177067114053786	ICSMLE 2022 : International Conference on Statistics and Machine Learning in Electronics	Categories computer science electronics machine learning statistics	Rationale The aim of the conference Statistics and Machine Learning in Electronics is to bring together young and experienced researchers, students and educators with interests in how statistics, machine learning and fuzzy logic could facilitate analysis, design, testing, diagnosis, measurement and manufacturing of electronic circuits, modules and devices. Final scientific results and research in progress will be presented and discussed, revealing new models, methods, algorithms, techniques, and methodologies. The conference is base for forming an international scientific network, spreading new and innovative ideas and the best practices. Topics Statistics, machine learning and fuzzy logic methods, algorithms, techniques, methodologies, models in: - Electronic circuit design - Electronic circuit analysis - Electronic circuits and devices testing and diagnosis - Electronics circuits and devices measurement - Manufacturing of electronic components and devices - Electronic process management - Quality management in electronics - Digital twins	Link: https://sites.google.com/view/easmle/conference	N 1 20 N 1 20
2	0.0015335100115374198	EAMM 2022 : Handbook of Engineering Applications of Modern Metaheuristics	Categories metaheuristics optimization artificial intelligence computer	All papers must be original and not simultaneously submitted to another journal or conference. EAMM2022 proceedings will be published in the Studies in Computational Intelligence series by Springer Nature Switzerland AG All books published in the series are submitted for consideration in Web of Science. Each article is expected to cover Single Objective Optimization. Thus, we strongly encouraged authors to test the performance of their proposed state of the art algorithms on either any real-world engineering application such as Electrical and power systems, machine learning, Robotics and Expert Systems, Pattern Recognition, Image processing, Bioinformatics, and biomedical engineering, Electronics and communication engineering, Manufacturing Science and Operation Research or one of CEC 2010-2017 for constrained benchmarks. Volume editors Dr. Tayyaz Akan (Rahkar-Farshi)	Link: https://easychair.org/cfp/eamm2022	N

Figura 5.2:

- Abstractul lucrării mele de licență cu optiunea de Machine Learning:

Home Page	Computer Science	Machine Learning						
Here are the first 10 results based on the input abstract and the selected category								
no.	SCORE	TITLE	CATEGORIES	ABSTRACT	LINK	WHEN	WHERE	SUBMISSIONS DEADLINE
1	0.0017722425772487903	RiTA 2022 : the 10th International Conference on Robot Intelligence Technology and Applications (RiTA 2022)	Categories machine learning artificial intelligence navigation SLAM	The 10th International Conference on Robot Intelligence Technology and Applications (RiTA 2022) will be held at Griffith University (Gold Coast Campus), Australia, on December 7 \square 9, 2022. We seek the latest research results, theories, or experimental developments, which will feature a high quality conference. RiTA will bring together researchers, scientists and experts in the field of robot intelligence technologies, and also to facilitate robotics community and collaborations. Conference Topics include: Machine learning	Link: http://2022.icrita.org/	Dec 7, 2022 - Dec 9, 2022	Gold Coast	Aug 31, 2022

Figura 5.3:

Astfel mai sus se pot observa înregistrările și scorul ce apare la început. Scorul este dat de algoritm TF-IDF. Pentru a clarifica puțin vreau să menționez că o valoare de 1.0 este maximă pentru scor și ar însemna că cele două texte comparate sunt identice, astfel chiar și un scor de 0.09 este foarte mare.

Acum voi afișa cum arată celelalte două pagini ale căror scoruri sunt calculate între abstractul și categoriile lor:

- Pagina cu recomandări de Computer Science:

no.	TITLE	CATEGORIES	ABSTRACT	
1	ICSMLE 2022 : International Conference on Statistics and Machine Learning in Electronics	Categories computer science electronics machine learning statistics	Rationale The aim of the conference Statistics and Machine Learning in Electronics is to bring together young and experienced researchers, students and educators with interests in how statistics, machine learning and fuzzy logic could facilitate analysis, design, testing, diagnosis, measurement and manufacturing of electronic circuits, modules and devices. Final scientific results and research in progress will be presented and discussed, revealing new models, methods, algorithms, techniques, and methodologies. The conference is base for forming an international scientific network, spreading new and innovative ideas and the best practices. Topics Statistics, machine learning and fuzzy logic methods, algorithms, techniques, methodologies, models in: - Electronic circuit design - Electronic circuit analysis - Electronic circuits and devices testing and diagnosis - Electronics circuits and devices measurement - Manufacturing of electronic components and devices - Electronic process management - Quality management in electronics - Digital twins	Link: https://www.ccsenet.org/online-first/icsmle-2022
2	ICATS 2022 : 1st International Conference On Advance in Technology and Science	Categories science engineering medical science computer science	Welcome to the 1st International Conference On Advance in Technology and Science 2022 (ICATS®-2022). The conference will be held on 31st March 2022 in Digital Mode Online. The main objective of ICATS®-2022 is to present the research from different areas of science and technology. This conference provides a platform for researchers and scientists across the world to exchange and share their experiences and research results about all aspects of electronics and information technology. This conference also provides an opportunity to interact and establish professional relations for future collaboration. The conference aims to promote innovations and work of researchers, engineers, students and scientists from across the world on Advancement in engineering, science, medical science. The basic idea of the conference is what more can be done using the existing technology. Papers accepted in ICATS® will be publish in Scopus Web of Science index Journal.	Link: http://www.ccsenet.org/online-first/icats-2022
3	EAMM 2022 : Handbook of Engineering Applications of Modern Metaheuristics	Categories metaheuristics optimization artificial intelligence computer science	All papers must be original and not simultaneously submitted to another journal or conference. EAMM2022 proceedings will be published in the Studies in Computational Intelligence series by Springer Nature Switzerland AG All books published in the series are submitted for consideration in Web of Science. Each article is expected to cover Single Objective Optimization. Thus, we strongly encouraged authors to test the performance of their proposed state of the art algorithms on either any real-world engineering application such as Electrical and power systems, machine learning, Robotics and Expert Systems, Pattern Recognition, Image processing, Bioinformatics, and biomedical engineering, Electronics and communication engineering, Manufacturing Science and Operation Research or one of CEC 2010-2017 for constrained benchmarks. Volume editors Dr. Tayyaz Akan (Rahkar-Farsi) Ayvansaray University, Turkey, University of Pardubice, Czech Republic Dr. Ahmed M. Anter Beni-suef University, Beni-suef, Egypt Prof. A.	Link: http://www.ccsenet.org/online-first/eamm-2022

Figura 5.4:

- Pagina cu recomandări de Machine Learning:

no.	TITLE	CATEGORIES	ABSTRACT	
1	RITA 2022 : the 10th International Conference on Robot Intelligence Technology and Applications	Categories machine learning artificial intelligence navigation SLAM	The 10th International Conference on Robot Intelligence Technology and Applications (RITA 2022) will be held at Griffith University (Gold Coast Australia, on December 7-9, 2022. We seek the latest research results, theories, or experimental developments, which will feature a high conference. RITA will bring together researchers, scientists and experts in the field of robot intelligence technologies, and also to facilitate robotics and collaborations. Conference Topics include Machine learning & Artificial intelligence & Decision making & Reasoning and inference & Navigation and SLAM & Dynamics and control & Human-robot interaction & Manipulation & Swarm and multi-robot system Optimization in robotics & HW/SW for robot computing	
2	CAIAC 2021 : International Cumhuriyet Artificial Intelligence Applications Conferencee 2021	Categories artificial intelligence machine learning ai applications	International Cumhuriyet Artificial Intelligence Applications Conferencee 2021 (CAIAC®-2021) will provide an excellent international forum knowledge and results in theory, methodology and applications of Artificial Intelligence. The Conference looks for significant contributions to all researchers and practitioners from both academia as well as industry to meet and share cutting-edge development in the field. Authors are solicited to contribute to the conference by submitting articles that illustrate research results, projects, surveying works and industrial experiences that describe significant advanced areas, but are not limited to. CAIAC is organized by Sivas Cumhuriyet University Artificial Intelligence and Data Science Applications and Center.	
3	ICSMLE 2022 : International Conference on Statistics and Machine Learning in Electronics	Categories computer science electronics machine learning statistics	Rationale The aim of the conference Statistics and Machine Learning in Electronics is to bring together young and experienced researchers, students and educators with interests in how statistics, machine learning and fuzzy logic could facilitate analysis, design, testing, diagnosis, measurement and manufacturing of electronic circuits, modules and devices. Final scientific results and research in progress will be presented and discussed, revealing new models, methods, algorithms, techniques, and methodologies. The conference is base for forming an international scientific network, spreading new and innovative ideas and the best practices. Topics Statistics, machine learning and fuzzy logic methods, algorithms, techniques, methodologies, models in: - Electronic circuit design - Electronic circuit analysis - Electronic circuits and devices testing and diagnosis - Electronics circuits and devices measurement - Manufacturing of electronic components and devices - Electronic process management - Quality management in electronics - Digital twins	
4	SIMMAC 2022 : International Symposium on Mathematical Methods Applied to the Sciences	Categories applied mathematics machine learning statistics modeling	The XXIII International Symposium of Mathematical Methods Applied to Sciences (XXIII SIMMAC) is the most important applied mathematics in Central America. It is organized by the Center for Research in Pure and Applied Mathematics (CIMPA) of the University of Costa Rica (UCR) ever with the collaboration of the School of Mathematics (EMat). Since 1978, this activity has been developed with the great participation of mathematicians from related disciplines from Central America, Europe, North America and South America. This edition will be virtual, and will take place in UCR Global in February 21-25, 2022. Topics: Data Analysis, Numerical Analysis, Applications, Approximation / Approximation Biomathematics Classification, Optimal Control, Differential Equations, Statistical Computing, Multivariate Statistics, Operations Research, Financial Mathematics Mining, Modeling, Optimization, Probability, Stochastic Processes, Dynamical Systems.	

Figura 5.5:

Astfel se pot observa înregistrările din categoria Computer Science, cum apar înregistrările în tabel și se pot utiliza săriile de rulare pentru a naviga în tabel.

De asemenea mai sus se poate observa interfața ce am implementat-o pentru acest sistem de recomandare de conferințe și jurnale. Pagina Machine Learning este identică ca și format cu pagina de Computer Science. Singurele diferențe sunt datele ce se extrag din tabele diferite pentru că sunt pe categorii diferite și deoarece prin acest proces nu se asigură că nu există înregistrări duplicate în tabel.

Capitolul 6

Concluzii și directii viitoare

În concluzie, sistemul funcționează cum am dorit pentru o fază incipientă. Datele sunt extrase prin web scraping destul de bine și am reușit să le filtrez și să le elimin foarte eficient intrările ce nu se potriveau, iar sistemul de recomandare face exact ceea ce trebuie, adică extrage datele, le pune un scor pe baza categoriei și a abstractului prin potrivirea acestora, pentru scor se utilizează algoritmul de TF-IDF, apoi sunt sortate descrescător pe baza scorului și sunt afișate într-un tabel. Datele sunt separate prin categorii direct de când sunt extrase, fiind salvate în tabele diferite pentru categoria diferență, astfel încât-o categorie nu există duplicări de informație. De asemenea înregistrările recomandate pe baza input-ului utilizatorului au o potrivire foarte bună.

Pentru directiile viitoare aș avea cateva idei de îmbunătățire a sistemului:

- Automatizarea script-ului pentru web scraping astfel încât datele să fie actualizate mai ușor;
- Crearea unei structuri de date ce reține data calendaristică; Astfel ar putea apărea două utilități importante ale acesteia:
 - prima ar fi aceea de sortare a recomandărilor pe baza datei calendaristice simultan cu scorul obținut de algoritmul TF-IDF;
 - a doua ar putea fi aceea de a elimina din lista ce se afișează, înregistrările ce au trecut de data calendaristică curentă.
- Altă îmbunătățire este cea de urcare a aplicației web pe un domeniu public.

Acestea sunt câteva directii viitoare și implementarea actuală a programului se găsește la adresa: <https://github.com/AndreiOprica/AppLicenta>.

Bibliografie

- [1] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37, 2008.
- [2] Armin Ronacher. Flask documentation, 2020.
- [3] Leonard Richardson. Beautiful soup documentation. *Dosegljivo: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> [Dostopano: 7. 7. 2018]*, 2007.
- [4] AM Kuchling. Regular expression howto. *Regular Expression HOWTO—Python*, 2(10), 2014.
- [5] Sebastian Nemetz, Sven Schmitt, and Felix Freiling. A standardized corpus for sqlite database forensics. *Digital Investigation*, 24:S121–S130, 2018.