Article Management System

0. Vizija

Ideja je bila kreirati projekat koji je nešto kompleksniji od zahtjeva samog predmeta, kako bi se taj isti projekat mogao prikazati u budućim prijavama za praksu, seminare, posao i slično.
Zato, projekat izlazi iz okvira standardnog „CRUD-esque“ formata, iako će definitivno imati i takve funkcionalnosti. Nadam se da to ne stvara veliki problem.

1. Koncept

Article Management System (trenutno nazvan „**Equilibrium**“, u projektnim fajlovima) jeste platforma za kreiranje artikala, njihovo predlaganje korisnicima platforme, te detaljniju obradu istih.

Jedna od primarnih motivacija za ovakvu platformu bila je platforma „Medium“, koja omogućuje objavljivanje članaka (često naučne prirode).

2. Funkcionalnost

Osnovna funkcionalnost se može zaključiti iz samog imena projekta:

(1) Korisnik može da doda novi članak
(2) Korisnik može da ukloni svoj članak
(3) Korisnik može da pogleda sadržaj već postojećeg članka
(4) Korisnik može da vrši pretragu članaka
(5) Korisnik može da zahtjeva preporuku članaka, u zavisnosti od njegovih prethodnih interesa
(6) Korisnik može da uspostavi interakciju sa člankom:
    a. „Lajkovati“ članak (reći da mu se „sviđa“ ili „ne sviđa“ članak)
    b. Postaviti komentar na članak
    c. Sačuvati članak
(7) Korisnik može da prikaže listu sačuvanih članaka
(8) Korisnik može da vidi statistiku svojih članaka (+ kroz vrijeme)
    a. Broj pregleda
    b. Broj lajkova / dislajkova („sviđa mi se“ / „ne sviđa mi se“ članak)
(9) Administrator (korisnik sa svim pravima) može da uradi sve navedeno, za sve članke

Dodatna funkcionalnost je, za sada, ne toliko jasno definisana, zbog kompleksnosti implementacije i dodatnih obaveza na oba fakulteta (kolokvijumi, ispitni rok), no:

(1) Administrator može da vidi statistiku čitave platforme (+ kroz vrijeme):
   a. **Broj korisnika**
   b. **Broj članaka**
   c. Generalno, **broj lajkova i dislajkova**
   d. Listu **svih tagova na člancima**, te broj članaka koji imaju određeni tag.

(2) Korisnik može da preuzme metapodatke (en. „**metadata**“) o svojim člancima – komprehenzivnu kolekciju podataka o člancima, kao jedan tekstualni dokument

(3) Dodatne funkcionalnosti pri pretraživanju, to jest **filtere**:
   a. Po **tag-u**, koje postavlja korisnik pri kreiranju članka
   b. Po **vremenu izdavanja**
   c. Po **dužini čitanja**

3. Implementacijski detalji

Čitava platforma implementirana je **OOP paradigmom**, koja, svejedno, odgovara ovakvom tipu programa.

Za sada, plan je kreirati platformu koja je **konzolna aplikacija**, ali sa dodatnim funkcionalnostima:

(1) Kretanje strelicama (na tastaturi) po prikazanom prozoru
Implementirano kako bi korisnik mogao da bira polje u koje trenutno unosi podatke, izabere opciju bez direktnog unosa rednog broja, slova ili riječi koja joj odgovara, i slično.

(2) Unos u više polja paralelno
Implementirano kako bi se zaobišao standardni Python ulaz, koji omogućuje unos u samo jednoj liniji.
Korisniku se može prikazati forma sa više polja za unos, koja čuvaju unijeti sadržaj i na kraju ga pakuju i mogu poslati dalje, kako bi se podaci uopšte mogli iskoristiti.
Da bi ideja bila malo jasnija – slika u prilogu bolje prikazuje navedenu ideju.

(3) Tekst u boji – već postoji kao opcija za printovanje, tako da definitivno nije „ručno“ implementiran, ali je implementiran bez neke biblioteke

(4) „**Scrolling**“, pri prikazivanju artikla (pomoću strelica).

Nekad u budućnosti biće implementiran „**metajezik**“ koji povezuje sve opcije koje se mogu dosegnuti strelicama, u smislu jezika koji opisuje šta se desi kada se pritisne neko dugme na tastaturi, ukoliko je selektovan neki objekat (recimo, ako se pritisne strelica prema dole, onda se pređe na sljedeće polje i sl.). Zbog manjka vremena trenutno to i nije baš moguće, ili je, bolje reći, „previše rizično za garantovati“, do januarskog roka.

Koriste se 2 ti tekstualnih fajlova: **.txt** i **.csv**.
Za oba tipa podatka lično ću implementirati parser(e) (koji svejedno nije teško implementirati, ali jedna od poenti predmeta jeste da se nauče parsirati fajlovi).

3.1 Korisnik

Podaci o korisnicima čuvaju se u kolektivnom .**csv** fajlu, koji sadrži sve relevantne kolone, kao što su **id, username, password, ime, prezime, datum rođenja, mjesto boravka, profesija, preference** i sl.

Postoji šansa dodavanja enkripcije / dekripcije na lozinku.

Gore navedene preference predstavljaju listu preferenci koje je korisnik pokazao pri interakciji sa člancima. Na primjer, ako je korisnik čitao članak koji govori o kvantnom računarstvu, te mu se članak svidio, čuvaće se **relevantni podaci** (u nastavku će biti više riječi o ovome), koji govore da mu se sviđaju članci koji se odnose na kvantno računarstvo. Ovo će se koristiti dalje za preporučivanje adekvatnih članaka, ukoliko to korisnik zahtjeva.

3.2 Članak

Tekst članka se čuva u tekstualnom fajlu, dok se „metapodaci“ (**id**, **autor**, **datum izdavanja**, **broj riječi**, **broj lajkova, broj dislajkova**, **dužina čitanja**, **tagovi**, itd) čuvaju u zasebnom fajlu.

Tekstualni sadržaj članka biće čuvan u **.txt** fajlu (radi lakšeg pregleda članka), kao i njegovi metapodaci, te će svaki fajl imati zaseban folder koji je relevantan za njega. Svaki članak će se unijeti i u jedan globalni **.csv** fajl, koji sadrži gore navedene metapodatke + sam tekst članka, tako da je čitavu „bazu podataka“ članaka jednostavno eksportovati kao **dataset**, koji bi se mogao koristiti, primjerice, za treniranje Machine Learning algoritama.

3.3 Machine Learning

Jedna od glavnih tačaka projekta jeste zapravo kreirati adekvatan Machine Learning model koji analizira, te preporučuje članke korisniku.

Za sada nije definitivno odlučena metoda pristupa, s obzirom da se prvi put bavim ovim dijelom mašinskog učenja.

O potencijalnim metodama može se pročitati na sljedećoj strani.

Ono što je vjerovatno jeste da će biti implementiran neki od sljedećih pristupa:

(1) **Latent Dirichlet Allocation** + **proizvoljni recommendation pristup baziran na konkretnoj sličnosti** – LDA će služiti za ekstrakciju „topic-a“ (tema) iz datog artikla, dok će se za preporučivanje koristiti neki algoritam koji mjeri sličnost između korisničkih preferenci (**Cosine Similarity**, **Pearson Correlation**, ...).
(2) **Latent Dirichlet Allocation** + **Neural Network-based recommendation** – LDA će služiti za ekstrakciju tema, dok će se kreirati neuralna mreža koja određuje koji artikal preporučiti korisniku. Pristup se može zasnivati na binarnoj klasifikaciji („preporuči“ - „ne preporuči“, tj. standardna logistička regresija), ili nekom drugom algoritmu za preporučivanje (svakako, vjerovatno „**content-based**“).
(3) **Korištenje unesenih tagova + Neural Network-based recommendation** – Tagovi artikla, te preference korisnika, kombinovaće se u neuronskoj mreži da kreira optimalne preporuke, moguće i kroz **embedded layer.**
(4) **RNN / LSTM Neural Network-based topic/keyword extraction** + **proizvoljni recommendation pristup baziran na konkretnoj sličnosti** – Kreirati (rekurentnu) neuralnu mrežu koja može da vrši ekstrakciju tema iz artikla, dok neki drugi algoritam (Cosine Similarity, Pearson Correlation, ...) vrši preporučivanje.
(5) **Korištenje unesenih tagova** + **proizvoljni recommendation pristup baziran na konkretnoj sličnosti** – u ovom slučaju i nije više baš pristup mašinskim učenjem, ali je najjednostavniji pristup za generisanje preporuka. Poželjno za izbjeći, ali biće implementiran u najgorem slučaju.

Za sada nije moguće garantovati pristup:

- **Pristup (1)** zvuči poprilično zanimljivo, ali je potencijalno limitirajuć u smislu kasnije analize podataka
- **Pristup (2)** je moguć, no binarna klasifikacija je možda „previše“ za navedeni use-case
- **Pristup (3)** je moguć, ukoliko se pronađe adekvatna literatura
- **Pristup (4)** je kompleksan, tako da nije baš vjerovatan
- **Pristup (5)** je moguć za implementaciju u jako kratkom vremenskom periodu, ali jednostavno „može se kreirati nešto bolje“.

Za implementaciju navedenih pristupa biće korišten **PyTorch – machine learning framework**, moderni industrijski i akademski standard, te automatski izbor u oblasti kompjuterske vizije i obrade prirodnih jezika. Takođe, zbog kompatibilnosti, koristiće se **numpy** i **pandas**, ali samo u ovom dijelu za treniranje modela. Notebook u kojem je kreiran model biće priložen.

Zamolio bih, ukoliko je ikako moguće, malo slobode u ovom dijelu implementacije, zato što je moja buduća karijera u ovoj oblasti, te bih volio da se ne rezortiram na konkretan pristup, već da „istražim“ svoje mogućnosti i tako naučim što je više moguće. Nadam se da ovo ne stvara problem.

4. Dodatni komentari

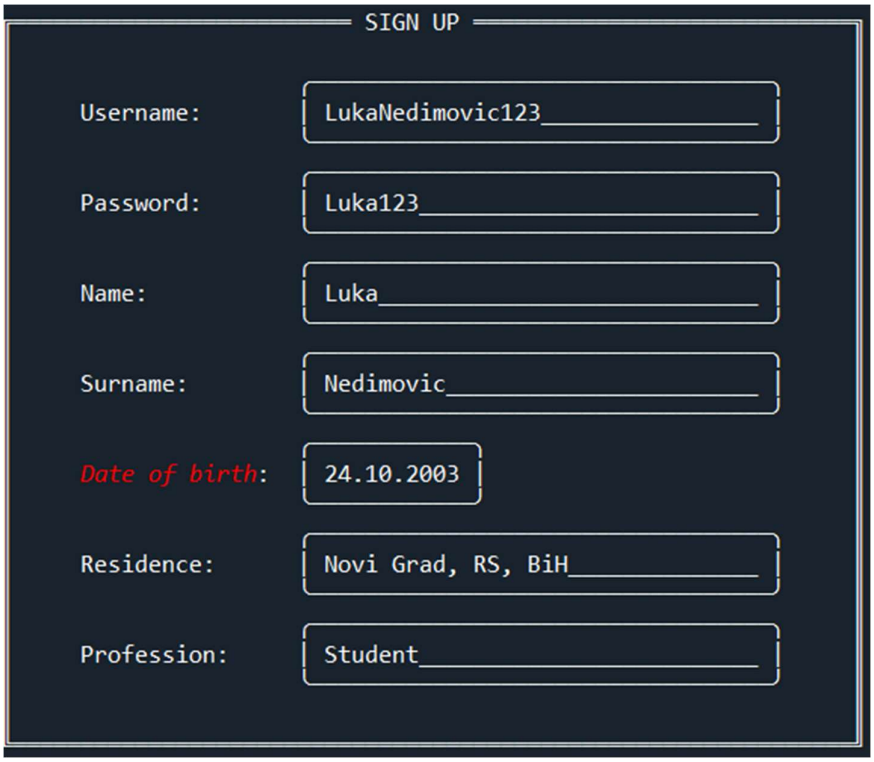
Ukoliko je potrebno, za treniranje će se koristiti dataset pronađen na internetu. U momentu pisanja projekta, sljedeći dataset (od 190 hiljada Medium artikala) izgleda jako zanimljivo i sa mnogo potencijala: **link**

„Data Science“ dio programa još nije definitivno odlučen, ali biće dodavane funkcionalnosti nakon što se implementiraju osnovne funkcionalnosti.
Iako je gore navedena osnovna funkcionalnost za iscrtavanje grafika, za šta će se koristiti **matplotlib**, poželjno je implementirati još neku funkcionalnost, kao topic analysis generalno članaka, šta korisnici generalno preferiraju i slično.

Program će se izvršavati u **beskonačnoj petlji**, koja osvježava i iscrtava trenutni ekran sa novim vrijednostima / u novom stanju.

Kod je ispisan i komentarisan na engleskom jeziku, zbog njegovog objavljivanja nakon završetka ispitnog roka.

5. Appendix



Primjer izgleda jednog **Prompt**-a, koji može da prima više ulaznih vrijednosti odjednom, te da ih upakuje i proslijedi na dalju obradu.
Postoji mogućnost estetičke promjene korisničkog interfejsa, ali funkcionalnost ostaje.