

Kolaborativní filtrování

BI-VWM 2020/2021

Minářová Markéta, Křestan Radim

Popis projektu

Cílem projektu je implementovat algoritmus kolaborativního filtrování, který spočívá ve výpočtu podobnosti mezi uživateli a na základě toho pak podle těch nejpodobnějších určí, která položka z dané nabídky by se nám mohla líbit nejvíce. Základní motivací, proč daný algoritmus implementovat, je, že zákazníci na libovolné stránce/e-shopu mají doporučený obsah „na míru“, což by mělo mít za následek větší prodeje, případně vyšší spokojenost při používání stránek (youtube, amazon, atd.). Projekt jsme se rozhodli implementovat pomocí frameworku Django pro jazyk python, který umožňuje snadné tvoření webových stránek. Největší výhodou frameworku je jeho jednoduchost a zvladatelná „learning-curve“. Jako databázový systém jsme zvolili sqlite, což se později ukázalo lehce nešťastné, kvůli nedostatku datových typů. Co se týče vstupů a výstupů, tak vstupem je označení některých knih tlačítka like/dislike a následné kliknutí na tlačítko, které provede výpočty doporučovacího algoritmu. Výstup by měl být seznam pěti knih, které se pro vás vyhodnotí jako nejlepší. Konkrétní produkt, který doporučuje naše aplikace, jsou knížky, ale snadno by se dala přepsat i na jiný produkt.

Způsob řešení

Postupovali jsme systematicky - nejdříve jsme vypočítali podobnost se všemi ostatními uživateli a následně jsme odhadovali pravděpodobné hodnocení

knížek, které uživatel ještě neohodnotil. Pro výpočet podobnosti jsme použili Pearsonův koeficient a pro predikci druhý vzorec, který funguje dále popsaným způsobem.

$$\text{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}}$$

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in U} \text{simil}(u, u')(r_{u',i} - \bar{r}_{u'})$$

V daném vzorečku x/u reprezentuje uživatele, který je zrovna přihlášen a y/u' reprezentují ostatní uživatele. Důležité je povšimnout si také našeho „r s pruhem“, které slouží ke snížení relevance uživatelů, kteří by například hodnotili všechno negativně nebo všechno pozitivně, čímž by zkreslovali svoje hodnocení, tudíž „r s pruhem“ značí průměrné hodnocení nějakého uživatele. Funguje tak, že vlastně počítáme, jak moc se uživatelské hodnocení odlišuje od jeho průměru, tím pádem bude mít jedno kladné hodnocení v moři těch negativních obrovský dopad a naopak. Koeficient k slouží pak jako takové „zprůměrování sumy“ podobných uživatelů a jejich hodnocení knihy „i“.

Hodnoty si ukládáme do databáze, a to jako True/False pro Like/Dislike resp. pokud je hodnota True, tak v našem vzorečku hodnotu 1, pokud False, tak -1 a pokud uživatel knihu neohodnotil, tak se hodnota ponechává na nule. Na výpočet odmocnin jsme použili knihovnu math, které dodává funkci sqrt().

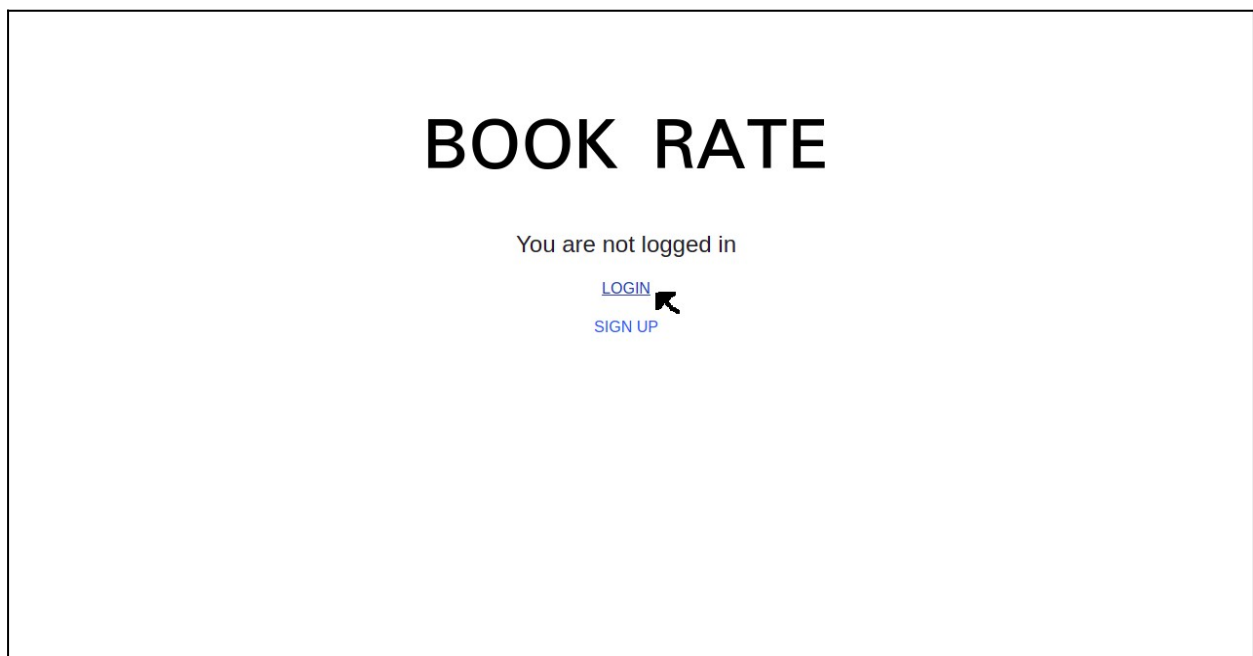
Vzhledem k tomu, že náš model příliš detailně nepracuje s tím, jak moc se daná kniha má líbit, jenom to porovnává s ostatními, tak by se teoreticky ve vzorečku mohlo vynechat „r_u s pruhem“, jelikož to slouží pouze k tomu, aby výsledné hodnocení reflektovalo to, jak uživatel normálně hodnotí.

Implementace

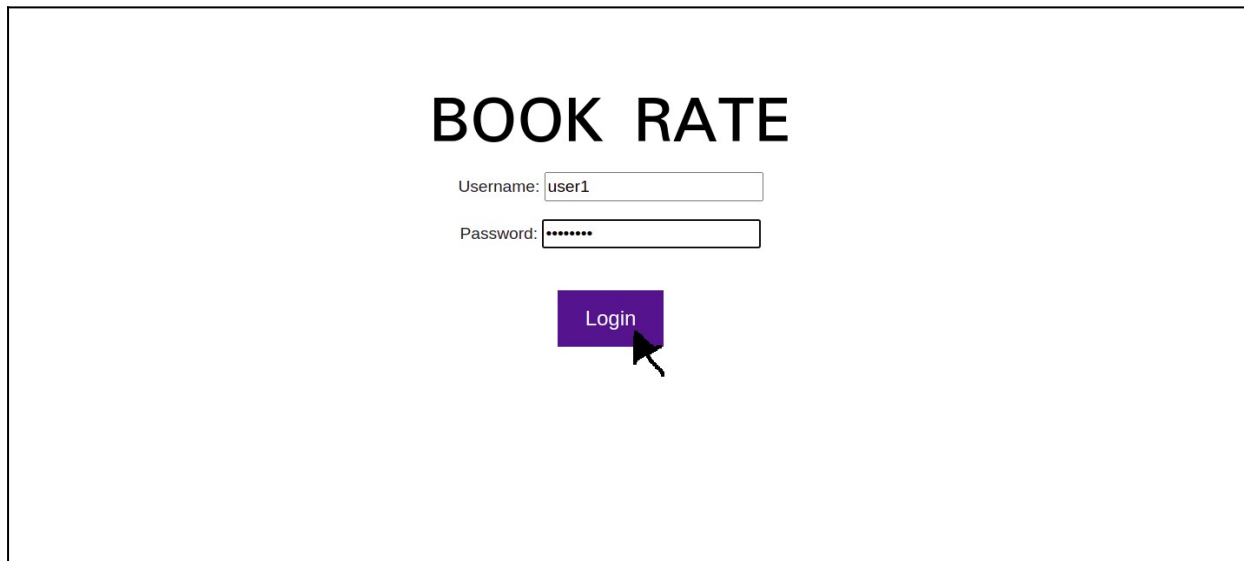
Jak už bylo řečeno, základem celého projektu je jazyk python a jeho framework Django. Kromě zabudovaných knihoven a utilit Django jsme nepoužili skoro žádnou jinou knihovnu, krom math na složitější výpočty a Bootstrap. V databázi je určitý počet uživatelů, kteří mají vyplněné svoje preference, podle kterých se vytváří výpočty nových uživatelů. Pokud však vytvoříte dostatek nových účtů, tak můžete algoritmus zkreslit. Výpočet se provádí pro uživatele po zmáčknutí tlačítka, při pouhém likování nedochází k úpravě preferencí!

Příklady výstupu

Obrázek č.1: Přihlášení



Obrázek č.2: Přihlášení



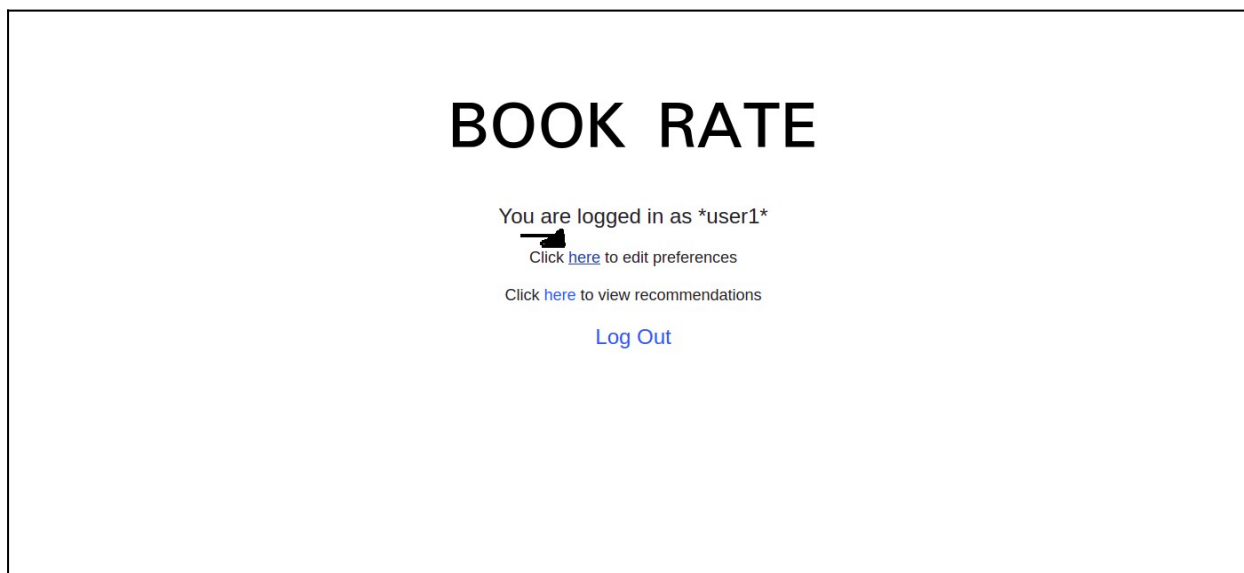
BOOK RATE

Username:

Password:

[Login](#)

Obrázek č.3: Kliknout na odkaz pro preference



BOOK RATE

You are logged in as *user1*

[Click here](#) to edit preferences

[Click here](#) to view recommendations

[Log Out](#)

Obrázek č.4: Zakliknout pár knížek

Select books you liked or disliked

Alice in Wonderland, Lewis Carroll

LikeDislike

The Maze Runner, James Dashner

LikeDislike

The Hunger Games, Suzanne Collins

LikeDislike

Catching Fire, Suzanne Collins

LikeDislike

Mockingjay, Suzanne Collins

LikeDislike

1984, George Orwell

LikeDislike

To Kill a Mockingbird, Harper Lee

LikeDislike

The Lord of the Rings, J.R.R. Tolkien

LikeDislike

The Great Gatsby, F. S. Fitzgerald

LikeDislike

Pride and Prejudice, Jane Austen

LikeDislike

The Hobbit, J.R.R. Tolkien

LikeDislike

Little Women, L. M. Alcott

LikeDislike

Jane Eyre, Charlotte Bronte

LikeDislike

Animal Farm, George Orwell

LikeDislike

Romeo and Juliet, William Shakespeare

LikeDislike

Wuthering Heights, Emily Bronte

LikeDislike

Gone Girl, Gillian Flynn

LikeDislike

IT, Stephen King

LikeDislike

Atonement, Ian McEwan

LikeDislike

Quo Vadis?, Henryk Sienkiewicz

LikeDislike

Obrázek č.5 Kliknout na odkaz pro doporučování

BOOK RATE

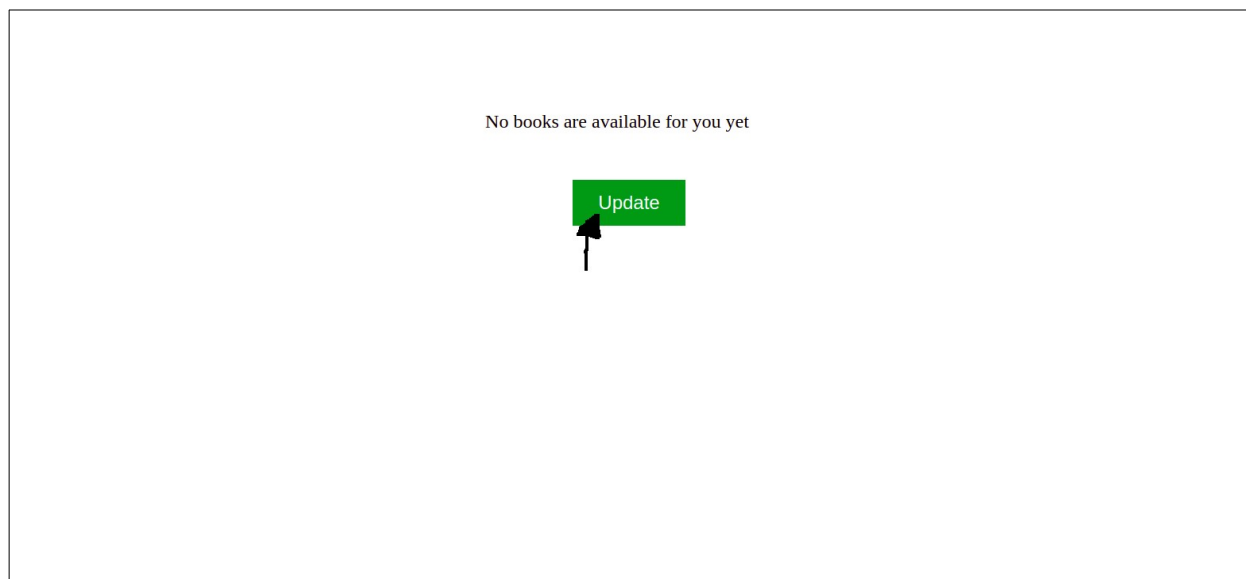
You are logged in as *user1*

Click [here](#) to edit preferences

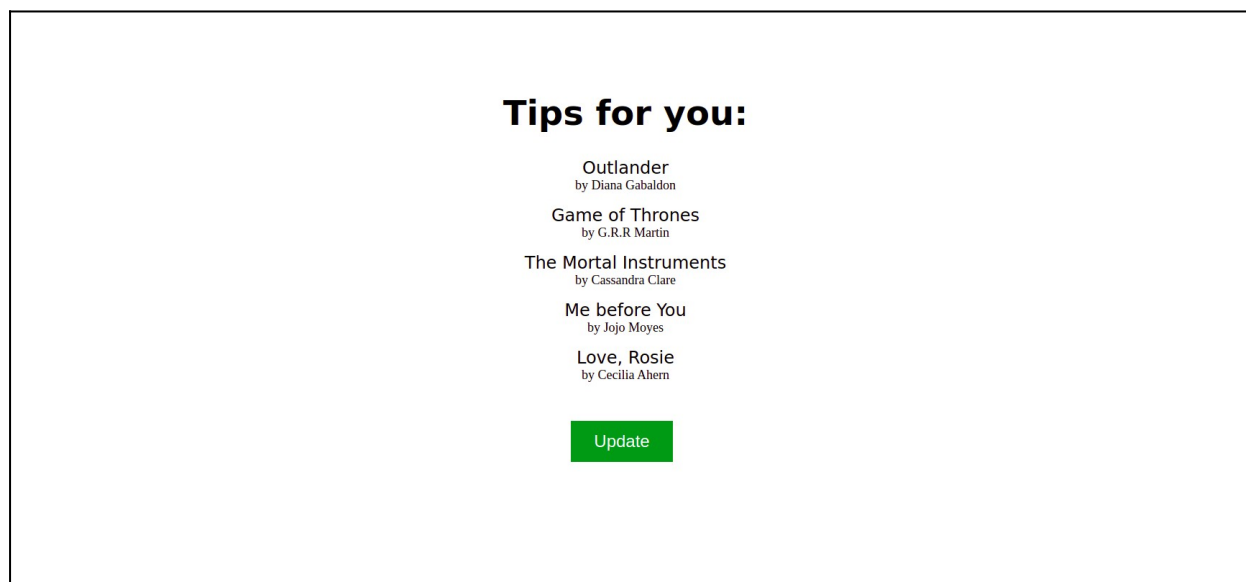
Click [here](#) to view recommendations

Log Out

Obrázek č.6: Kliknout na update doporučení



Obrázek č.7: Doporučené knihy



Experimenty

Vzhledem k tomu, že náš výpočet má tři hlavní parametry, tak jsme na simulovaném projektu zkoušeli, jak jednotlivé zvyšování parametrů ovlivní rychlost algoritmu. Časovým hodnotám odpovídá následující tabulka:

	count	count	count	count
Rating	1000	10000	1000	1000
Book	1000	1000	10000	1000
User	1000	1000	1000	10000
time	0.2s	1.52s	3.94s	4.79s

Z tabulky vyplývá, že přidávání hodnocení nemá příliš velký vliv na rychlost výpočtu, kdežto pokud by se zvyšoval počet uživatelů, či knih, tak je algoritmus poměrně zdlouhavý.

Diskuze

Naše řešení rozhodně není to nejškálovatelnější a určitě by sneslo mnoho úprav jak v designu, tak v implementaci. První důležitý bod je, že pokud bychom měli reálnou aplikaci, tak bychom uživateli nedávali přímý nástroj, jak zahltit server výpočty, ale řešilo by se to nějakým časovačem, který by to uživatelům vypočítal. Další velký problém vidím ve škálovatelnosti našeho řešení, protože počty knih jsou “hard-coded” což by znamenalo nutný update kódu. V reálu by také bylo riziko zkreslení dat různými boty, kteří by se snažili zdiskreditovat naši aplikaci, to je však bezpečností otázka a problém bychom řešili důmyslnější autentizací. Uživatelská přívětivost by měla být v reálu lepší a lidé by v ideálním případě měli ještě algoritmus na vyhledávání knih a celková database by byla mnohonásobně větší.

Závěr

Na druhou stranu jsme přesvědčeni, že dobře předvádíme fungování a principy kolaborativního filtrování na malém množství dat, která nejsou nijak zkreslená. Díky projektu jsme se naučili spoustu nových věcí a problém jsme v nějaké zjednodušené formě vyřešili a s výsledky jsme spokojeni.