Recommendation engine pro IPTV – doporučování TV obsahu pro uživatele

Bc. Ondřej Janota

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej Janota**
Osobní číslo:	**A18410**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Recommendation engine pro IPTV – doporučování TV obsahu pro uživatele**
Téma práce anglicky:	**An engine for IPTV – recommending TV content to users**

Zásady pro vypracování

1. Vypracujte literární rešerši.
2. Analyzujte současný stav doporučování (mediálního) obsahu uživatelům v systému.
3. Navrhněte strukturu recommendation systému.
4. Vypracujte praktické řešení s cílem doporučení obsahu podle navržené struktury a pravidel.
5. Zvažte možnosti nasazení metod z paradigmatu A.I. s ohledem na systémové požadavky, výkon a přesnost aplikace.
6. Zhodnoťte přínosy navrženého systému a navrhněte další možná vylepšení.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. VARGHESE, Shiju, 2015. Web Development with Go: Building Scalable Web Apps and RESTful Services. 1. Apress. ISBN 9781484210529.
2. FATI, Suliman Mohamed, Saiful AZAD a Al-Sakib Khan PATHAN. *IPTV delivery networks: next generation architectures for live and video-on-demand services*. Hoboken, NJ, USA: Wiley, 2018. ISBN 9781119397915.
3. GRUS, Joel. Data science from scratch. Sebastopol: O'Reilly, 2015, xvi, 311 s. ISBN 9781491901427.
4. *Data science & big data analytics: discovering, analyzing, visualizing and presenting data*. Indianapolis: Wiley, [2015], xviii, 410 s. ISBN 9781118876138.
5. GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. Beijing: O'Reilly, [2017], xx, 545 s. ISBN 9781491962299.

Vedoucí diplomové práce: **doc. Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **28. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně .....

podpis autora

ABSTRAKT

Hlavním cílem této diplomové práce je navrhnout a vytvořit doporučovací systém pro IPTV službu. Program RecTV vytvořený v jazyce Go bude sloužit pro získávání podobnosti TV obsahu. Dále bude doporučovat TV obsah uživatelům podle toho, co za poslední dobu zhlédli. Systém umožňuje výpočet podobnosti obsahu na základě požadovaných pravidel. V teoretické části práce jsou popsány informace o IPTV službách a existujících typech systémů pro doporučování obsahu. Návrh samotného systému a pravidel pro filtrování, programové řešení, testování systému a možnosti aplikace metod umělé inteligence jsou popsány v druhé, projektové části práce. Výsledkem této práce je plně funkční a otestovaný doporučovací systém, který umožňuje získávat doporučení pro uživatele a předávat tato doporučení systémům IPTV služby pro další zpracování.

Klíčová slova: Doporučovací systém, IPTV, doporučování, TV obsah, míra podobnosti, TF-IDF

ABSTRACT

The aim of this diplomat thesis is to design and implement a recommendation system for IPTV service. The program RecTV was created with Go programming language and will be used for finding similarities of TV content. The system will recommend TV content to users according to what they had seen lately. System allows calculation of TV content similarities based on required rules. Information about IPTV services and existing types of content recommender systems is described in the theory part of this thesis. The project part describes design of such system and filtering features, software solution, system testing and possibilities of applying A.I. methods. The result of this work is fully functional and tested recommendation system that allows to procure recommendations for users and pass these recommendations to other IPTV service systems for further processing.

Keywords: Recommendation engine, IPTV, recommending, TV content, similarity measure, TF-IDF

Tímto bych chtěl poděkovat panu doc. Ing. Romanu Šenkeříkovi, Ph.D. za jeho odbornou pomoc a cenné rady, které mi při realizaci této práce poskytoval. Také bych rád poděkoval všem svým kolegům a hlavně rodině za psychickou podporu v průběhu celého studia.

OBSAH

ÚVOD ... **9**

I TEORETICKÁ ČÁST ... **10**

1 IPTV – INTERNETOVÁ TELEVIZE ... **11**

1.1 Živé vysílání ... 12

1.1.1 Multicast a protokol IGMP ... 12

1.2 Zpětné přehrávání ... 12

1.3 Nahrávky ... 13

1.4 VOD ... 13

1.5 Programový průvodce ... 14

2 PŘEHLED EXISTUJÍCÍCH ŘEŠENÍ ... **15**

2.1 Collaborative filtering ... 15

2.1.1 Memory-based metody – metody nejbližšího souseda ... 15

2.1.2 Model-based metody ... 17

2.2 Content-based metody ... 18

2.2.1 TF-IDF ... 19

2.2.2 Míra podobnosti a vzdálenosti ... 20

2.2.3 Profil uživatele ... 20

2.3 Hybridní metody ... 21

2.4 Zhodnocení metod doporučování ... 22

II PROJEKTOVÁ ČÁST ... **24**

3 POUŽITÉ TECHNOLOGIE ... **25**

3.1 Programovací jazyk Go ... 25

3.1.1 Rutiny, kanály a mutexy ... 25

3.1.2 Worker a WaitGroup ... 26

3.1.3 Další výhody jazyka ... 26

3.2 MariaDB ... 26

3.3 Docker a docker-compose ... 27

4 NÁVRH DOPORUČOVACÍHO SYSTÉMU ... **28**

4.1 Původní stav doporučování ... 28

4.2 Pravidla pro doporučování ... 29

4.2.1 Žánry ... 29

4.2.2 Herci ... 29

4.2.3 Režiséři ... 30

4.2.4 Scénáristé ... 30
4.2.5 Krajiny původu ... 30
4.2.6 Klíčová slova ... 30
4.3 Příprava databáze ... 30
4.4 Architektura ... 31
4.5 Problém s dostupností obsahu ... 32
5 VLASTNÍ IMPLEMENTACE ... 33
5.1 Práce s databází ... 33
5.2 Úpravy před výpočtem podobností ... 34
5.3 Provedené optimalizace ... 35
5.4 Program RecTV ... 36
5.4.1 Spuštění RecTV ... 36
5.4.2 Výpočet podobnosti ... 37
5.4.3 Získávání doporučení ... 40
5.4.4 Serverová část ... 40
5.5 Zabezpečení systému ... 42
6 TESTOVÁNÍ ... 44
6.1 Unit testy ... 44
6.2 Cross validation ... 45
6.3 Výsledky testování ... 46
7 MOŽNOSTI APLIKACE METOD A.I. ... 48
ZÁVĚR ... 49
SEZNAM POUŽITÉ LITERATURY ... 50
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK ... 55
SEZNAM OBRÁZKŮ ... 56
SEZNAM PŘÍLOH ... 57

ÚVOD

Doporučovací systémy mají za úkol předpovídat preference nebo ohodnocení určitého uživatele, a to pomocí filtrování informací, které by pro daného uživatele nebyly přímo žádoucí. Mnoho webových, mobilních či smart TV aplikací již využívají nějakého doporučovacího systému, které doporučují například filmy, hudbu, obrázky apod. Také se můžeme setkat se systémy, které nám dokáží podle preferencí doporučit např. restaurace, hotely, ale také osoby, které by nás mohly zajímat. Ve světě se uchytily zejména dva typy doporučovacích systémů, a to založené buď na kolaborativním nebo obsahově-založeném filtrování. Tato práce se zabývá druhým typem s přihlédnutím na výhody a nevýhody prvního typu doporučovacích systémů.

Cílem této práce je získat přehled o existujících doporučovacích systémech a po analýze současného stavu doporučování obsahu ve společnosti, pro kterou pracuji, navrhnout a implementovat nové řešení doporučovacího systému. Firma se zabývá transkódováním a distribucí živého televizního vysílání prostřednictvím internetu pomocí interně vyvinutých aplikací (služba IPTV).

Práce se zabývá analýzou informací získávaných z televizního obsahu a specifikací pravidel, na základě kterých se bude filtrovat doporučovaný obsah. Podle požadavků společnosti je navržena architektura systému a vybrány technologie pro tvorbu systému RecTV.

V první části tohoto dokumentu je popsána problematická oblast, pro kterou je vytvářen doporučovací systém. Je také zaměřena na existující řešení doporučování obsahu uživatelům a postupy, které se používají při tvorbě doporučovacích systémů.

V projektové části je uveden podrobný popis využitých technologií při tvorbě RecTV, návrh celého systému a pravidel, podle jakých jsou filtrována doporučení uživatelům. V této části je také popsáno programové řešení, metody testování, jež byly při vývoji využity, a možnosti aplikace metod umělé inteligence.

Analýza současného stavu doporučování je v této práci popsána ze dvou pohledů – v teoretické části jako přehled aktuálních metod (viz. kapitola 2) a v praktické části jako popis stávajícího skutečného stavu systému (viz. podkapitola 4.1).

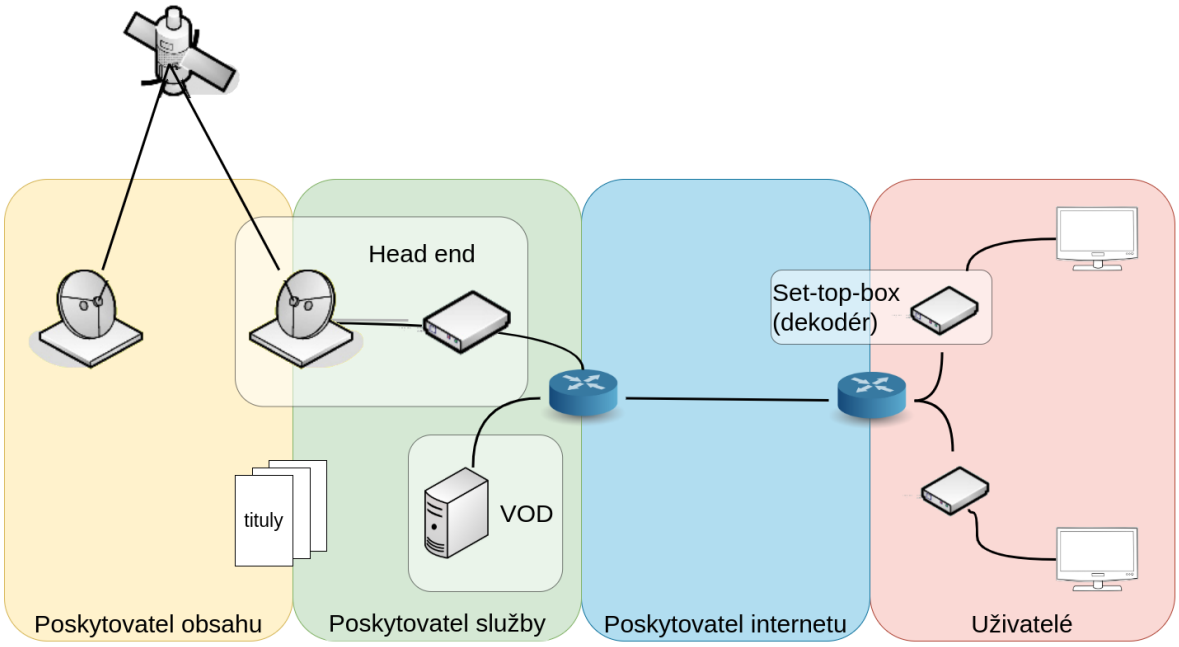
I. TEORETICKÁ ČÁST

1 IPTV – internetová televize

Od prvního objevu televize proběhlo mnoho změn, přes šíření analogového vysílání až k tomu digitálnímu. V 50. letech 20. století se objevovala první barevná vysílání. Z pohledu poskytování tohoto vysílání se televizní služby šířily pouze prostřednictvím pozemního vysílání, později se přidalo vysílání kabelové. V dnešní době mají tato vysílání stále značný podíl na trhu ve velkém množství zemí. S digitalizací se začala televize šířit i pomocí satelitního vysílání, navíc se mohla zlepšit kvalita vysílaných kanálů, přidat pomocná data jako např. titulky nebo televizní program.[1]

Jelikož ve světě dochází k obřímu nárůstu širokopásmových sítí, je dnešním trendem vše proměnit podle konceptu *all-over-IP*. Jediným cílem poskytovatelů internetu již není pouze vysokorychlostní internet, ale také doprovodné služby jako VOIP (telefonní služba přes internetový protokol [2]), IP dohled, chytré domácnosti nebo právě IPTV (televize přes internetový protokol [3]). Počátkem 21. století začaly vysílat první komerční IPTV služby.[1]

IPTV můžeme chápat jako proces přenášení a vysílání televizních programů přes internetový protokol IP – od poskytovatele obsahu až k uživatelům (znázorněno na obrázku 1.1). Tato služba je pro uživatele v porovnání s klasickým televizním přenosem mnohem komfortnější. Hlavní výhodou IPTV se stal přenos přes širokopásmové připojení, jelikož je tento typ přenosu mnohem efektivnější. Proto, abychom mohli sledovat klasický přenos, musíme přijímat všechna vysílání kanálů současně. V případě internetové televize přijímáme pouze vysílání požadovaného kanálu.[3]



Obr. 1.1 Architektura IPTV - přeloženo a upraveno z [4]

Sledující uživatel při přenosu skoro nerozezná rozdíl od klasického televizního vysílání. Jelikož je IPTV založen na obousměrném přenosu (dokáže vysílat data uživateli, ale také získávat informace od uživatele), může na rozdíl od klasického přenosu uživatelům poskytnout řadu jedinečných funkcí. Hlavní funkcí zůstává živé vysílání, přidává se ale také služba TimeShift, díky které uživatel může např. pozastavit přehrávání pořadu a vrátit se zpět na předchozí pořad. Další výhodnou funkcí se stala služba PVR, která umožňuje uživatelům nahrát některý z pořadů a podívat se na něj později. A v neposlední řadě také služba VOD, jež obsahuje řadu titulů, které můžeme sledovat kdykoli.[4]

V České Republice existuje hned několik takových IPTV služeb. Mezi nejznámější služby patří ty, které jsou poskytovány samotnými mobilními operátory – O2 se službou O2TV, T-Mobile s T-Mobile TV a Vodafone(přes UPC) s Horizon Go. Dalšími velice známými IPTV službami jsou např. KUKI, SledováníTV, Lepší.TV či 4NET.TV.[6]

1.1 Živé vysílání

Internetová televize by nemohla být televizí, pokud by se nedokázala vypořádat s živým vysíláním. Toto živé vysílání se musí zpracovat, zašifrovat a poslat dále k uživatelům, vše musí být provedeno s minimální latencí, aby uživatel téměř nerozeznal rozdíl oproti klasickému vysílání. Živé vysílání je přenášeno přes IGMPv2 protokol [3]. V rámci živého vysílání může být poskytnuta interaktivita, která umožňuje dané vysílání například zastavit a případně později pokračovat se službou TimeShift [4].

1.1.1 Multicast a protokol IGMP

Díky protokolu IGMP (*Internet Group Management Protocol*) je umožněno komunikovat pomocí multicastu. Příjemci multicastu se pomocí tohoto protokolu nejdříve musí přihlásit do multicastové skupiny u nejbližšího směrovače, poté od tohoto směrovače začnou přijímat data. Pokud již příjemce nechce data přijímat, odhlásí se z multicastové skupiny u směrovače. Směrovač se v určitých intervalech doptává, zda-li ještě existuje alespoň jeden příjemce. Pokud by již žádný neexistoval, je uzavřena multicastová skupina a přestanou se šířit data.[7]

1.2 Zpětné přehrávání

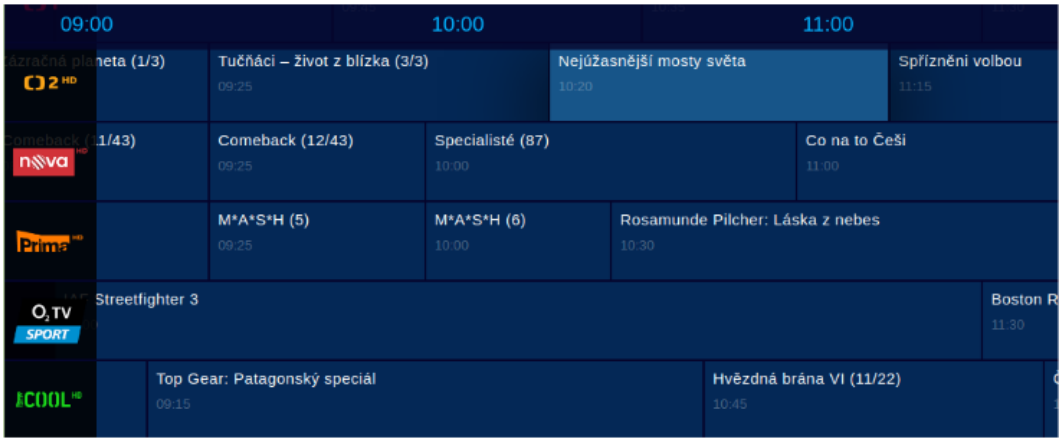
Pomocí digitálního videorekordéru může uživatel sledovat televizní pořady vysílané v minulosti několik dnů nazpět [3]. Můžeme se také setkat s názvy TimeShift [4] či Catch-up [5]. Data vysílaného pořadu nejsou uložena přímo pro jednoho uživatele, ale pro všechny uživatele. Poskytovatel IPTV ukládá pro levnější a účinější řešení pořady z minulosti do DVR systému [3].

1.3 Nahrávky

Aby uživatel mohl sledovat pořady také mimo službu TimeShift, která bývá časově omezována, existují nahrávky (PVR), které uživateli umožní nahrát a uložit pořady na serveru pro pozdější zhlédnutí [4]. Nahrávky jsou stejně jako zpětné přehrávání uloženy v DVR systémech [3].

1.4 VOD

Touto zkratkou označujeme videa, která uživatel může sledovat kdykoliv, kdy si o ně požádá [1, 3]. Ze seznamu dostupných videí si uživatel může vybrat jakékoli video a spustit ho i vícekrát. Tento seznam můžeme nazývat videotékou či katalogem videí. Pro přehrávání tohoto typu vysílání se používá unicast (opak broadcastu) přes RTSP protokol [3].



Obr. 1.2 Porovnání teletextového zobrazení ČT a TV Programu IPTV služby O2 TV [8, 9]

1.5 Programový průvodce

Výhodou IPTV je také programový průvodce (EPG), který bývá součástí nejen webových, ale také mobilních i smart TV aplikací. Oproti normálnímu teletextovému programu je zde zobrazeno mnohem více informací – u klasického teletextu například získáváme informace pouze k jedinému kanálu (případně skupin kanálů [8]). Na obrázku 1.2 je znázorněno teletextové zobrazení z webu ČT [8] a část TV programu služby O2 TV [9].

2 Přehled existujících řešení

Kvůli zkvalitňování uživatelských zážitků nebo například pro reklamní účely se v posledních desetiletích začaly uchycovat systémy, které dokážou uživatelům doporučit něco, co by se jim mohlo líbit. Všeobecně tyto systémy můžeme považovat za algoritmy, jež se zaměřují na návrhy žádoucích položek uživatelům [10]. Mezi hlavní důvody, proč společnosti tyto systémy začaly vytvářet, patří odlišení se od konkurence nebo zvýšení množství příjmů [4, 10]. Prodejem více rozličných předmětů, zvýšením spokojenosti či věrnosti zákazníků a také lepší znalostí uživatelových preferencí zvyšuje pravděpodobnost dosáhnutí obou hlavních cílů [4].

V podkapitole 2.1 je popsána metoda kolaborativního filtrování, sekce 2.2 je zaměřena na metodu založenou na podobnosti obsahu a kombinace dvou předchozích metod je popsána v podkapitole 2.3. Na konci této kapitoly jsou popsány hlavní výhody a nevýhody používání zmíněných metod.

2.1 Collaborative filtering

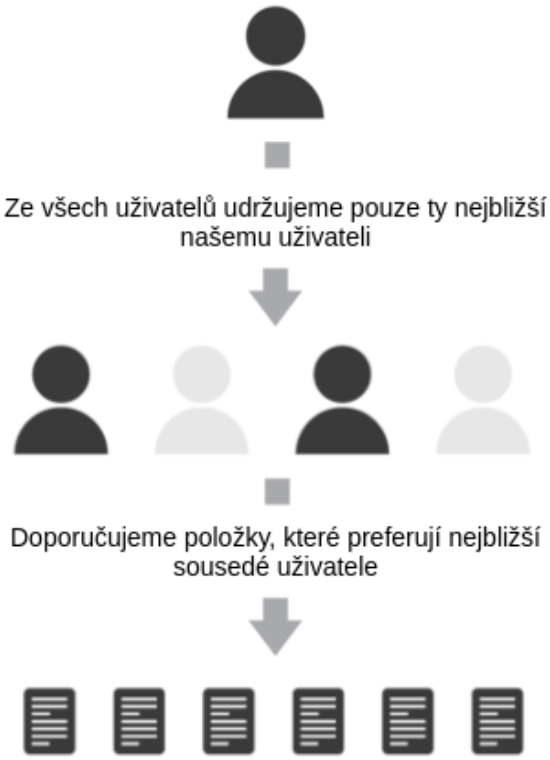
V rámci doporučovacích systémů jsou kolaborativní metody založeny na interakcích, které proběhly v minulosti. Abychom dokázali vytvářet nová doporučení, jsou zaznamenávány interakce mezi uživateli a položkami např. do matice interakcí s uživatelskými položkami [10, 11]. Za předpokladu, že jsou tyto interakce dostatečné k detekci podobnosti mezi uživateli či položkami, můžeme na základě podobností vytvářet předpovědi [10]. Kolaborativní filtrování je nejpoužívanější a nejpopulárnější metodou při implementaci doporučovacích systémů [4].

V podkapitole 2.1.1 jsou popsány metody založené na paměti, které jsou většinou založeny na hledání nejbližších sousedů [11]. Metody založené na modelu jsou popsány v podkapitole 2.1.2.

2.1.1 Memory-based metody – metody nejbližšího souseda

Mezi metody, jež jsou založeny na paměti, patří user-user a item-item přístupy. Pro tyto přístupy není potřeba vytvářet žádný model, který by vytvářel nová doporučení. Všechny informace se získávají pouze z matice interakcí mezi uživatelem a položkou [10]. Největšími výhodami těchto metod jsou jednoduchost implementace, možnost vysvětlit na základě čeho byly položky doporučeny, účinnost a stabilita při přidávání nových uživatelů či položek [4].

User-user přístupy se snaží identifikovat uživatele s nejpodobnějším profilem interakcí k navržení položek, které jsou mezi těmito sousedy nejoblíbenější a zároveň jsou

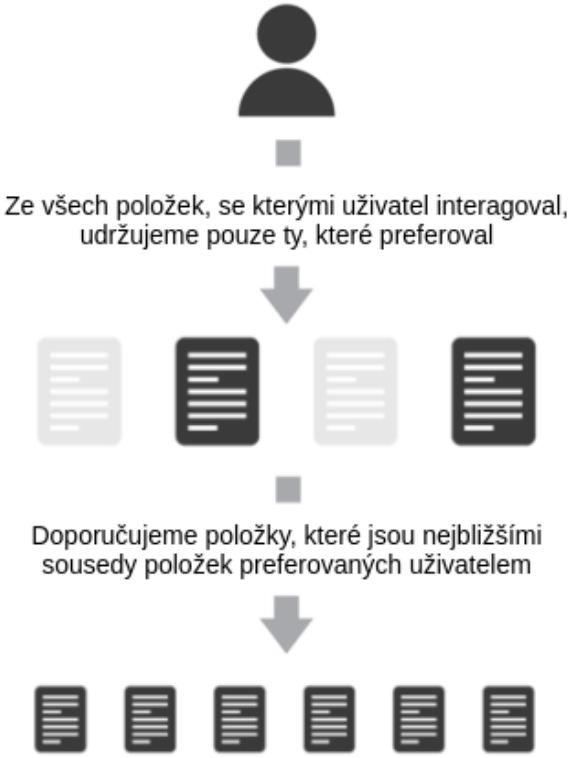


Obr. 2.1 Doporučení položek přístupem user-user – přeloženo z [10]

také novými pro tohoto uživatele (viz. obrázek 2.1) [10]. Na základě míry podobnosti se vyhodnocuje vzdálenost mezi uživateli. Mezi nejpoužívanější metody pro získání vzdálenosti pomocí míry podobnosti patří cosine similarity (kosinová podobnost) [11]. Na základě této vzdálenosti dokážeme získat nejbližší sousedy a doporučovat tak poté položky těchto sousedů. Abychom mohli uživateli něco doporučit, nesmíme najít úplně stejného uživatele, ale pouze takového, jenž má většinu stejných interakcí s položkami [10].

Item-item přístupy pracují s podobností položek, se kterými uživatel interagoval. Pokud většina uživatelů interaguje s některou položkou stejným způsobem (například formou líbí/nelíbí), je tato položka považována za podobnou. Stejným způsobem, jako u přístupu user-user, nalezneme nejbližší sousedy preferovaných položek uživatele a ty mu doporučíme (viz. obrázek 2.2).[10]

Porovnání user-user a item-item přístupů Metoda user-user je velice citlivá na zaznamenané interakce, jelikož se hledá podobnost mezi uživateli, kteří interagovali pouze s několika položkami, vzniká tak velký rozptyl. Výhodou tohoto přístupu je získávání více personalizovaných výsledků díky doporučením založených pouze na interakcích



Obr. 2.2 Doporučení položek přístupem item-item – přeloženo z [10]

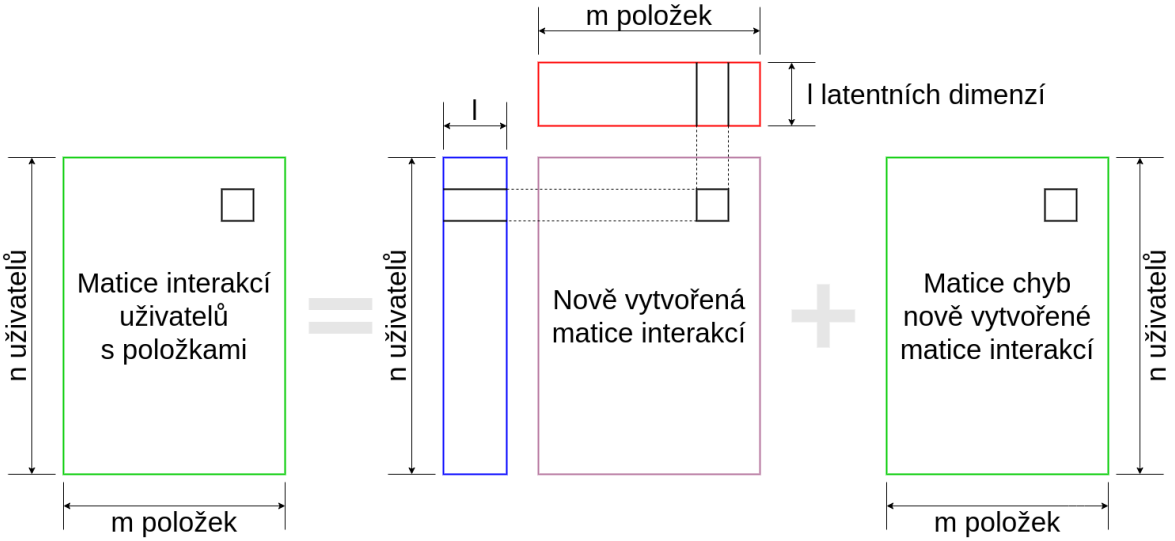
zaznamenaných podobnými uživateli, je zde tedy nízká zaujatost. V případě přístupu item-item již metoda není tak citlivá na zaznamenané interakce, ale výsledky doporučení mohou být velmi zkreslené, jelikož se pracuje s interakcemi všech typů uživatelů, a to i těch, kteří si nejsou vůbec podobní.[10]

2.1.2 Model-based metody

V metodách založených na modelu se předpokládá vytvoření latentního modelu, který by vysvětlil interakce s uživatelskými položkami. Aby se snížil šum a zrychlil se celkový výpočet, tento latentní model vytváří nový, zmenšený prostor funkcí původní matice uživatele či položky [4, 10, 11]. Tyto metody se tedy používají pro snižování počtu dimenzí [12]. Pro vytvoření takového modelu se používají dvě metody – matrix factorization a metody hlubokého učení.

Matrix factorization je metodou pro snižování počtu dimenzí prostoru [12]. Získání nízkorozměrného latentního prostoru funkcí je hlavním předpokladem při použití faktorizace matic. V tomto prostoru můžeme reprezentovat jak uživatele, tak i položky. Výpočtem skalárního součinu odpovídajících vektorů v prostoru můžeme získat interakce mezi uživatelem a položkou [10]. Při výpočtu snižujeme počet dimenzí a je tak

více než jasné, že budou vznikat nějaké chyby. Výsledným součtem nově vytvořené matice a matice chyb přibližně získáme původní matici (viz. obrázek 2.3).



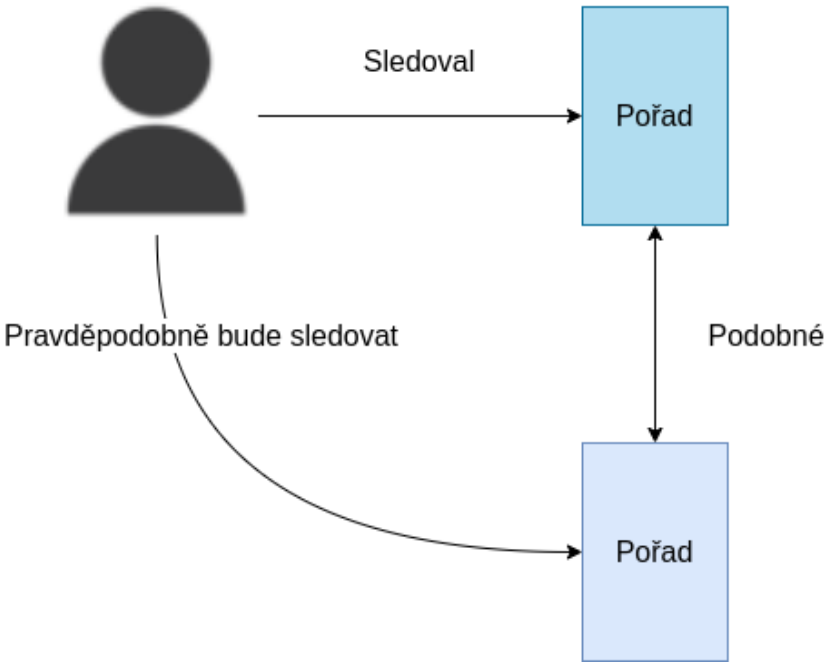
Obr. 2.3 Faktorizace matice – přeloženo z [10]

Deep-learning je více flexibilní oproti matrix factorization. Díky flexibilitě vstupní vrstvy sítě je možné zvyšovat relevanci doporučení zahrnutím vedlejších pravidel, která lépe vyjadřují zájmy uživatele. U těchto metod je však těžší trénování a také náročnější dotazování se na doporučení (vedlejší pravidla je vždy nutné zpracovávat při dotazu) [13].

2.2 Content-based metody

Content-based metody se na rozdíl od kolaborativního filtrování nezaměřují na interakce všech uživatelů s položkami. U těchto metod se snažíme získat podobnost jednotlivých položek a na základě této podobnosti je dále doporučit uživateli (viz. obrázek 2.4). Při použití této metody tedy vytváříme profily položek (obsahu) [14]. Tyto položky jsou definovány množinou známých atributů, kterou jsou jasně charakterizovány. Obvykle se nejprve snažíme vyextrahovat veškeré atributy položek a poté určujeme vhodnost tohoto atributu pro účely doporučování. Atributy položek jsou většinou prezentovány jako vektory s nějakou váhou, získanou například metodou TF-IDF [4, 14]. Pro získávání podobných položek se využívá například míry podobnosti či vzdálenosti.

U těchto metod se můžeme setkat s klasifikačními problémy, kdy se snažíme zjistit, zda-li se položka uživateli líbí či nikoliv. Nebo také regresními problémy, při kterých se snažíme předvídat, jaké hodnocení by uživatel položce přiřadil. Tyto problémy jsou popsány v podpodkapitole 2.2.3.



Obr. 2.4 Logika doporučení u content-based metod

2.2.1 TF-IDF

Tato metoda se používá pro zpracovávání přirozeného jazyka [15], ale také pro extrakci informací. Dalo by se říct, že tato metoda získává váhu či důležitost prvku v dokumentu. Prvně se vypočítá frekvence výskytu prvku v daném dokumentu ke všem obsaženým prvkům v dokumentu [4, 14] – tento výpočet je popsán vztahem 2.1.

$$Tf\,(t) = \frac{cetnost\_vyskytu\_prvku\_t\_v\_dokumentu}{celkovy\_pocet\_prvku\_v\_dokumentu} \tag{2.1}$$

Dále je potřeba získat inverzní četnost dokumentu (IDF), ta vyjadřuje, jak moc je daný prvek v daném dokumentu vzácný. Vypočítá se jako celkový počet dokumentů k četnosti dokumentů obsahujících daný prvek [4, 14] – výpočet je popsán vztahem 2.2.

$$Idf\,(t) = log_{10}\left(\frac{celkovy\_pocet\_dokumentu}{pocet\_dokumentu\_obsahujici\_prvek\_t}\right) \tag{2.2}$$

Celkový výsledek TF-IDF je součinem těchto dvou hodnot [4, 14] a je dán vztahem 2.3.

$$TFIDF\,(t) = Tf\,(t) \cdot Idf\,(t) \tag{2.3}$$

Abychom získali lépe měřitelnou hodnotu, můžeme výsledné váhy ještě normalizovat [16], a to například pomocí kosinu [4].

2.2.2 Míra podobnosti a vzdálenosti

Abychom získali podobnost jednotlivých položek na základě vypočítaných vah, můžeme k tomu využít míry podobnosti. Mezi nejpoužívanější varianty patří kosinová podobnost, euklidovská vzdálenost nebo Pearsonova korelace.

Cosine similarity je jednou z nejčastěji používaných metod pro výpočet podobnosti [4, 14]. Vypočítá velikost úhlu mezi dvěma vektory a je dána vztahem 2.4.

$$cos\left(\phi\right) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.4}$$

.

Euklidovská vzdálenost je nejjednodušší a nejpoužívanější metodou pro výpočet vzdálenosti mezi vektory a je dána vztahem 2.5.

$$d(A, B) = \sqrt{\sum_{i=1}^{n} \left(A_i - B_i\right)^2} \tag{2.5}$$

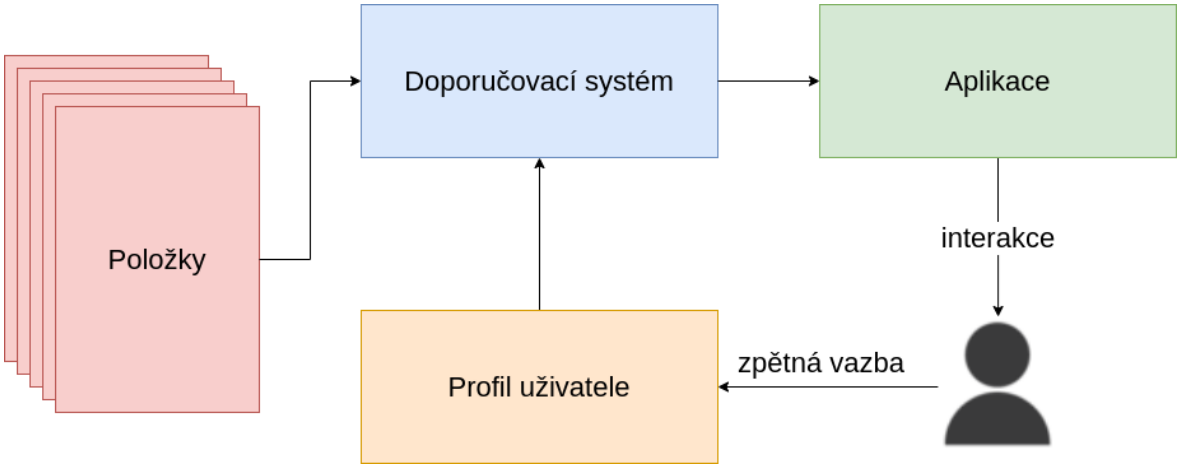
Pearsonův korelační koeficient se používá spíše pro podobnosti mezi uživateli v kolaborativním filtrování, ale lze použít pro výpočet podobnosti u content-based metod. Pearsonův korelační koeficient je dán vztahem 2.6.

$$\rho_{A,B} = \frac{cov\left(A, B\right)}{\sigma_A \cdot \sigma_B} \approx \frac{\sum_{i=1}^{n} \left(A_i - \bar{A}\right) \cdot \left(B_i - \bar{B}\right)}{\sqrt{\sum_{i=1}^{n} \left(A_i - \bar{A}\right)^2} \cdot \sqrt{\sum_{i=1}^{n} \left(B_i - \bar{B}\right)^2}} \tag{2.6}$$

2.2.3 Profil uživatele

Pokud dokážeme získávat zpětnou vazbu od uživatelů, je vhodné využít složitějších metod pro získávání doporučení. Na základě těchto zpětných vazeb dokážeme vytvořit profil či model pro uživatele a získávat tak lepší doporučení podle jeho preferencí (viz. obrázek 2.5). Abychom dokázali predikovat, zda-li se uživateli nějaká položka líbí, je vhodné využít například naivního bayesovského klasifikátoru, který bude zaměřený na položky. Pro každou z položek bychom chtěli natrénovat naivní bayesovský klasifikátor se vstupy vektoru uživatele a výstupy v podobě líbivosti položky uživatelem. Snažíme se tedy získat pravděpodobnost hodnoty *líbí* a *nelíbí* pro danou položku na základě uži-

vatelského vektoru. Druhou možností je predikování uživatelského hodnocení položek. Pro každého uživatele chceme vytvořit lineární regresivní model, který na vstupech přijímá vektor položky a výstupem bude hodnocení pro danou položku.[10]

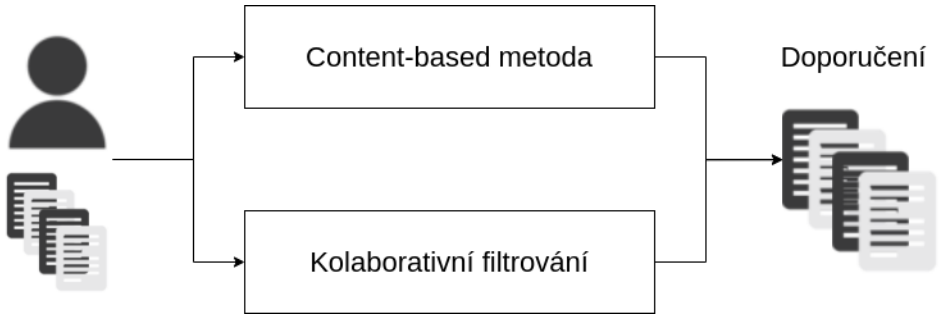


Obr. 2.5 Způsob doporučování se zpětnou vazbou

2.3 Hybridní metody

Jelikož každá z předchozích metod má své plusy i mínusy, využívá se hybridních metod, které se snaží (například kombinací metod) eliminovat všechny nedostatky. Deep learning modely mohou být velmi efektivní při kombinaci kolaborativního filtrování s content-based metodou [11]. Existuje však mnoho způsobů jak kombinaci těchto dvou metod zpracovat.

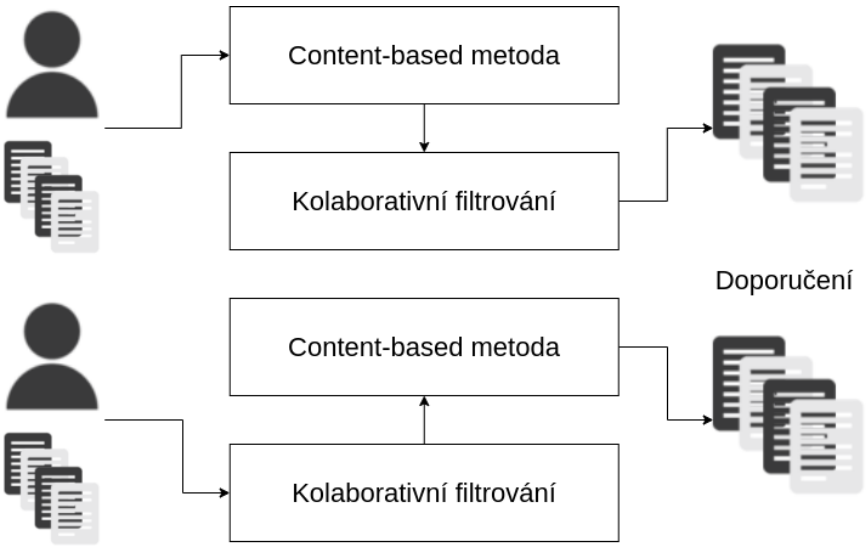
Jednou z možností může být využití content-based metody pouze v případě, kdy pro kolaborativní filtrování neexistuje dostatečné množství interakcí uživatele s položkami [11] a doporučit například top n nejdoporučovanějších a jim podobných položek.



Obr. 2.6 Kombinace content-based metody a kolaborativního filtrování – přeloženo z [17]

Kolaborativní filtrování dokáže doporučovat lepší, personalizované výsledky uživatelům, ale nedokáže říct, na základě čeho byla doporučení získána, kromě toho, že je

systém doporučil na základě podobného uživatele. To zase lépe dokáže content-based metoda, která přesně ví, na základě čeho bylo něco doporučeno.



Obr. 2.7 Použití výsledků jedné metody jako vstupy další metody – přeloženo z [17]

Můžeme tedy vytvořit hybridní systém, který bude kombinovat doporučení z výsledků content-based metody a výsledků kolaborativního filtrování (viz. obrázek 2.6). Další možností jak získat lepší doporučení je využití doporučení jedné metody jako vstupy do metody druhé (viz. obrázek 2.7) [17].

2.4 Zhodnocení metod doporučování

Největší nevýhodou většiny metod je problém se studeným startem [4, 10, 11]. Je to stav, kdy nemáme dostatek informací o uživateli, abychom takovému uživateli mohli něco doporučit (stejný problém může nastat u nových položek). U kolaborativního filtrování nám z počátku chybí dostatek interakcí uživatele s položkami. Takovému uživateli nelze doporučovat a nelze doporučovat ani jiným uživatelům na základě tohoto nového uživatele. V takovém případě je nutné zavádět doprovodné algoritmy, jež budou doporučovat například nové položky nejaktivnějším či náhodným uživatelům a nebo náhodné či nejoblíbenější položky novým uživatelům. U content-based metod je problém pouze s absolutním startem nového uživatele, kdy mu nelze na základě jeho historie (žádnou nemá) něco doporučit. Pokud však uživatel začne alespoň nějak interagovat s položkami, jsou mu ihned doporučeny podobné položky.

Nevýhodou content-based metod je vytváření bublin, které vznikají tím, že nedoporučujeme již nic zajímavého, ale pouze položky, které již byly například koupeny, zhlédnuty [15] apod. Jelikož doporučují tyto metody podle podobnosti obsahu, vzniká problém s položkami, které nemají dostatek informací, protože bez těchto informací

nelze podobnost položek vypočítat.

Výhodou kolaborativního filtrování v porovnání s content-based je, že nepotřebují znát žádné informace o položkách ani uživatelích a lze je tak využívat v mnoha situacích. Efektivita systému se zvyšuje novými interakcemi uživatelů s položkami a platí tedy, že čím více uživatelů s položkami interaguje, tím lépe [10]. Oproti tomu se zase u metod založených na podobnosti obsahu nesetkáme s problémem přidávání nové položky a odpadá tak problém studeného startu. S novou položkou není třeba nijak interagovat velkým množstvím uživatelů [15]. Výhodou je také nezávislost uživatele, kterému dokážeme doporučovat cokoliv pouze na základě jeho vlastní historie, a také jednoduché vytvětlování důvodů doporučení položek.

Nejlepších výsledků dosáhneme spojením content-based metody s kolaborativním filtrováním, kdy se částečně eliminují nevýhody obou metod. Kombinací získáme systém, který nebude mít velké problémy se studeným startem, také dokáže doporučovat více personalizované výsledky i s odůvodněním doporučení pro uživatele. Tímto způsobem se také odstraní problém s uzavíráním uživatele do vlastní bubliny.

II. PROJEKTOVÁ ČÁST

3 Použité technologie

Společnost vlastní velmi obsáhlou databázi, která běží v clusteru na několika serverech. Tyto MariaDB databáze jsou velice rychlé a lze s nimi pracovat stejně dobře jako s databázemi MySQL. Aby bylo možné předejít komplikacím, byla využita taktéž databáze MariaDB, která je spustitelná v docker kontejneru pomocí *docker-compose*. Jelikož společnost využívá pro svá řešení programovacího jazyku Go, byl při tvorbě RecTV použit právě tento jazyk. Go je velmi dobře škálovatelný, také velice rychlý a efektivní v porovnání např. s jazykem Python [18]. Pomocí metody *runtime.GOMAXPROCS()* a Go rutin [19] lze také lehce spouštět paralelní výpočty. Jazyk Go má pro mne také výhodu v striktní syntaxi, bohaté knihovně balíčků a vestavěném testovacím nástroji.

3.1 Programovací jazyk Go

V listopadu 2009 vyšla první verze programovacího jazyku Go (logo na obrázku 3.1), na níž pracovali Robert Griesemer, Rob Pike a Ken Thompson od roku 2007 [20]. Je to poměrně nový jazyk, má však velmi kvalitní dokumentaci a také si našel již velké množství obdivovatelů. Tento staticky typovaný kompilovaný jazyk svým zpracováním připomíná dynamicky typovaný interpretovaný jazyk. Výhodou tohoto jazyka je kompilace zdrojových kódů do jednoho spustitelného binárního souboru.

Obr. 3.1 Logo Go [21]

3.1.1 Rutiny, kanály a mutexy

Go routines jsou svým způsobem odlehčené verze vláken [22]. Běh funkce (metody) spuštěné v rutině probíhá asynchronně, je využíváno stejného adresního prostoru, a proto je potřeba využívat synchronizačního balíčku pro přistupování k paměti. Použití mutexů [23], které při přístupu k proměnné tuto proměnnou zablokují a ostatní procesy musí počkat, dokud nebude proměnná odblokována. V běhu rutin lze také využívat kanálů [24], které lze v jednom procesu naplňovat objekty a v druhém rovnou tyto objekty vytahovat a zpracovávat. Vše musí být dokonale sladěno, aby nenastalo vytahování objektů z kanálu dříve, než bude něčím naplněn.

3.1.2 Worker a WaitGroup

Jelikož lze pomocí rutin paralelně zpracovávat metody, které nemusí dokončit svou práci před určitým bodem, kdy je nutné, aby byla metoda již zpracována, používá se tzv. WaitGroup [25]. Tato synchronizační pomůcka zabrání spuštění dalšího kódu, dokud nebude vše potřebné zpracováno. Proces probíhá tak, že se před spuštěním rutiny přidá worker zpracovávající spuštěnou rutinu [26], pomocí metody `Add()`, a rutině se předá ukazatel na WaitGroup. Po dokončení rutiny se nad touto skupinou zavolá metoda `Done()`. V původním vlákně se pomocí metody `Wait()` čeká, dokud všichni přidaní workeři nedokončí svou práci.

3.1.3 Další výhody jazyka

Metoda `Init()`, která je spuštěna již při načítání balíčků, se může hodit v případě, kdy chceme např. něco zpracovat před samotným během hlavního programu. Nemusíme se tak starat o volání požadované funkce, program se o vše postará sám.

V Go lze využívat soubory s příponou *.gob* [27], které jsou beze snahy čitelné pouze pro programy psané v Go. Tyto soubory lze nejen ukládat do souborového systému, ale také přeposílat přes síť pomocí kodéru a číst v jiné aplikaci pomocí dekodéru. Dokážeme tak tímto způsobem synchronizovat i velké složité struktury mezi jednotlivými servery či instancemi programu.

Velkou výhodou pro mne byla existence ukazatelů, které jsem využíval kvůli nižší paměťové náročnosti, ale také například při výměně dat v cachi (v rámci proměnné). Pokud je potřeba v cachi udržovat velké množství informací, které se občas musí za běhu vypočítávat znovu, lze pouhým přehozením ukazatele na nově vytvořenou paměť nepozorovaně vyměnit stará data za nová.

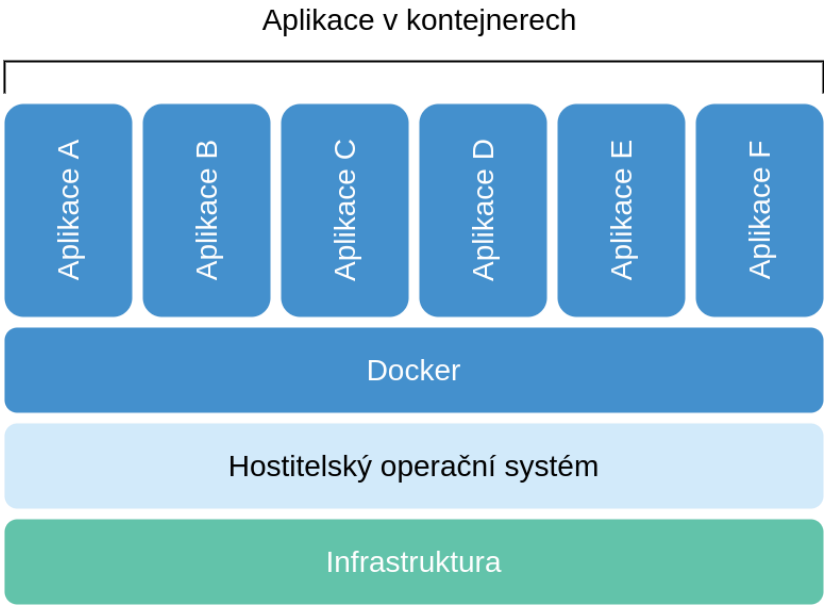
3.2 MariaDB

Původními vývojáři MySQL byla vytvořena relační databáze MariaDB. Poté, co Oracle zakoupil MySQL, již nebylo garantováno, že MySQL zůstane open source. Proto byla založena MariaDB Foundation [29], která garantuje, že MariaDB již bude navždy open source projektem. Díky své rychlosti, robustnosti a škálovatelnosti je MariaDB využívána v mnoha světových společnostech jako je Google nebo Wikipedie. Samotná nadace je podporována velkými společnostmi jako jsou například Microsoft, Booking.com či IBM [29]. Do verze 5.5 obsahovala MariaDB stejné funkce jako MySQL, pak již MariaDB pokračovala verzí 10, od kdy již nemusí obsahovat všechny funkce původní MySQL. Zajímavostí obou databází je, že jejich vývojář Michael „Monty“ Widenius pojmenoval tyto databáze po svých vlastních dcerách – My [30] a Maria [31].

Pro práci s databází jsem použil nástroj phpMyAdmin, který je sice vytvořen pro MySQL, ale MariaDB udržuje kompatibilní prostředky ke komunikaci s tímto nástrojem. S phpMyAdminem lze jednoduše zadávat SQL dotazy, vyhledávat v tabulkách a upravovat veškerá data. Oproti příkazové řádce, kde lze vykonávat stejné operace, je phpMyAdmin přehlednější a také obsahuje předem vytvořené zkratky pro nejpoužívanější příkazy (vyhledávání, vkládání, úprava, smazání, apod.).

3.3 Docker a docker-compose

Díky dockeru dokážeme vytvářet kontejnery, které poběží na všech zařízeních, které prostředí Docker podporuje (Linux, Windows, atd.). Dockerem vytvořené kontejnery jsou sice izolované stejně jako virtuální stroje, ale na rozdíl od nich tyto kontejnery využívají hostitelský operační systém (viz. obrázek 3.2). Právě kvůli tomu lze sdílet služby mezi jednotlivými kontejnery a mohou tak mezi sebou komunikovat.[32]



Obr. 3.2 Architektura Docker kontejnerů – přeloženo z [36]

Docker Hub je službou, v níž lze vyhledávat a sdílet obrazy kontejnerů [33]. Díky této službě je pak spouštění kontejnerů velmi jednoduchou záležitostí. V rámci RecTV jsem vytvořil kontejnery z obrazů *MariaDB* a *phpmyadmin/phpmyadmin*, konfigurace je obsažena na přiloženém CD v souboru *database/docker-compose.yml*. Aby nebyla ztracena data při restartování kontejneru, můžeme využít atributu *volumes*, v němž namapujeme cestu k souboru v našem systému na cestu k souborům uvnitř kontejneru (pro MariaDB i MySQL jsou databázové soubory uloženy v adresáři */var/lib/mysql*). Pokud je konfigurační soubor správně nastaven, lze pak kontejnery spustit jednoduchým příkazem `docker-compose up` [34] a poté ukončit příkazem `docker-compose stop` [35].

4 Návrh doporučovacího systému

Před samotnou implementací musel být velmi dobře promyšlen návrh systému. Jelikož se jedná o doporučování obsahu pro IPTV službu, vyskytuje se tu závažný problém s dostupností obsahu (vysvětleno v sekci 4.5). Aby se s tímto problémem systém RecTV vypořádal, musí být dostatečně flexibilní.

Požadavkem společnosti bylo získávat doporučení pouze na základě podobnosti obsahu. Nebylo tak prozatím nutné zahrnovat techniky založené na umělé inteligenci, avšak v případě rozšiřování systému na hybridní doporučovací systém by již bylo vhodné využít AI algoritmů pro získávání potencionálně lepších výsledků. Existují však stále možnosti využití složitějších statistických metod, které by postačily k vytvoření hybridního systému.

V podkapitole 4.4 je popsáno, jakým způsobem byl systém navržen a jak se z pouhých dat o obsahu vytvoří doporučení pro uživatele. Také je zde podrobný popis pravidel, podle kterých bude RecTV vypočítávat podobnost dvojic titulů. Nejdříve si však řekneme něco o nynějším stavu doporučování ve společnosti, pro kterou je RecTV navržen.

4.1 Původní stav doporučování

Ve společnosti prozatím neexistoval žádný automatizovaný doporučovací systém. Dalo by se říct, že nynější systém funguje jako poloautomat. V první řadě je nutné ručně v administraci vybírat tituly, které se budou uživatelům zobrazovat, v druhé řadě pak musí systém překontrolovat vybrané tituly. Nastává zde stejný problém s dostupností těchto pořadů, jako tomu je u RecTV.

Jedním z nedostatků původního stavu doporučování je existence pracovníka, který se stará o zadávání potencionálně dostupných pořadů do systému. Dalším nedostatkem, spjatým s předchozím, je výskyt problému s již vyplněnými seznamy, které se časem musí ručně kontrolovat a případně měnit například pořadí zadaných pořadů.

Hlavním, největším nedostatkem je, že doporučení nejsou nijak vázána na uživatele. Jsou to doporučení pouze obecná, protože nijak nevyužívají historii žádného uživatele. Všechna doporučení jsou vybírána náhodně např. podle žebříčku nejlepšího TV obsahu tohoto týdne apod. Systém jako poloautomat samozřejmě nenechává vše pouze na schopnostech zadavatele, zapojil se v podobě kontroly dostupnosti zadaných pořadů a žádnému uživateli se poté nezobrazí nedostupná doporučení.

Nejen to, že se nynější systém nedokázal trefit do uživatelovy obliby, vedlo ke vzniku nápadu na vytvoření automatizovaného systému RecTV, který dokáže na základě podobností obsahu doporučit tituly přímo podle uživatelem zhlédnutých pořadů a tím

potencionálně zvýšit oblibu doporučování pořadů.

4.2 Pravidla pro doporučování

Proto, abychom mohli doporučovat obsah uživateli, musíme nejprve nalézt podobnost mezi jednotlivými tituly. Jelikož o titulech udržujeme mnoho informací, je důležité vybrat pouze důležité atributy, podle kterých podobnosti získáme. Nedostatek takových pravidel by mohl způsobit, že budeme získávat i tituly, jež nejsou na první pohled podobné. Z toho vyplývá, že musíme zkombinovat vícero důležitých atributů.

Důležitým, avšak ne nejdůležitějším, pravidlem jsou žánry. Získáme tak tituly, které mají společné znaky z hlediska tématu, o kterém pojednávají.

K tomu, abychom našli pro uživatele podobné filmy či pořady, je nutné přidat také logiku na straně významnosti osob v jednotlivých titulech. Ve filmech se objevují stejní herci, moderátoři a hosté. Některé tituly jsou také známé podle svého režiséra, který natáčí filmy podobného typu a obsazuje skupinu svých oblíbených herců (typické např. pro Quentina Tarantina). Existují však i trilogie a filmy na pokračování, které již nejsou natočeny stejným režisérem, ale obsahují náměty stejného scénáristy. Velkou roli u scénářů hrají např. předlohy podle knih (jako je tomu u série filmů o Harry Potterovi podle knih J. K. Rowling), tuto informaci však o titulech neudržujeme, a proto si budeme muset vystačit pouze se scénáristy.

Většina nejznámějších filmů je natočena v USA, můžeme ale také nalézt úspěšné filmy ze severní Evropy či Nového Zélandu. Proto je také dobré dát slovo krajinám původu. Podle klíčových slov bychom zase dokázali nalézt nejlepší shody mezi jednotlivými tituly. Bližší důvody vybrání těchto atributů popíšeme v následujících podkapitolách.

4.2.1 Žánry

Seznam žánrů nám dokáže získat podobné tituly podle obsahu. Tímto atributem dokážeme nalézt podobnosti nejen mezi filmy, ale také mezi magazíny a zajímavými dokumenty, které při doporučování bývají často opomínány. Uživatel může sledovat určité typy pořadů a RecTV při doporučení tyto typy zohlední.

4.2.2 Herci

Herec je velice důležitým atributem, který RecTV využívá. Díky tomuto pravidlu můžeme nalézt pořady, v nichž hraje např. Brad Pitt nebo naopak pořady s herci, kteří nejsou natolik světoznámí.

Pokud bychom navíc chtěli doporučené tituly sázet do kategorií podle herců, bude se tento atribut velice hodit – v podstatě by se kategorie vytvořila a naplnila sama již

při výpočtu podobností.

4.2.3 Režiséři

Režisér je také důležitým atributem, podle něhož dokážeme dohledat velmi podobné tituly. Režiséři mají určitý vkus a většina známých titulů lze podle režisérů značně odlišit. Toto pravidlo je použito spíše pro doladění celého doporučení, protože (až na výjimky) lépe a přesněji funguje ve spojení s herci a scénáristy.

4.2.4 Scénáristé

Díky scénáristům dokážeme nejlépe naleznout další pokračování titulů. Nemusí vždy platit to, že pokračování nějakého filmu bude natáčet stejný režisér a už vůbec zde nemusí hrát stejní herci. Z většiny však scénář takových pokračování píše stejný scénárista. Také se stává, že scénárista zpracovává předlohy knih jednoho spisovatele, pak se propojí také filmy, které mají podobný děj, ale nemají vícero společných znaků.

4.2.5 Krajiny původu

Tento atribut není příliš důležitý, ale dokáže nám přidat hodnotu u filmů, které nejsou přímo světoznámé. Uživatelé si mohou oblíbit např. české filmy, tímto atributem získáváme výhodu a dokážeme vyhledat právě tituly, jež jsou z této krásné země. Nelze se však vztahovat pouze na Českou Republiku, tento atribut nám pomůže nalézt obsah, který je specifický pro danou zemi působení služby IPTV.

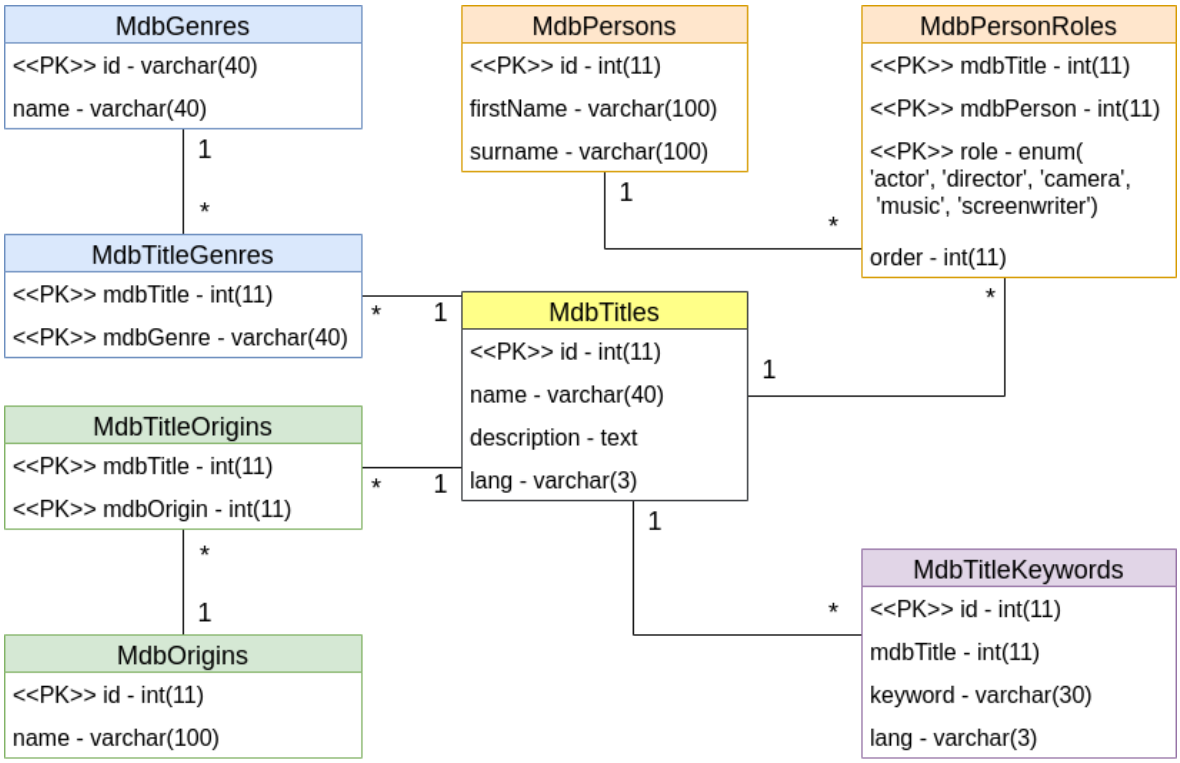
4.2.6 Klíčová slova

Pokud by byla klíčová slova vyextrahována přesně a striktně, bylo by možné díky tomuto pravidlu nalézt obsahově stejné tituly. Největším problémem v získávání klíčových slov jsou však odlišně dlouhé popisy filmů a pořadů. Získání opravdu přesných klíčových slov tak není vůbec jednoduché a v této práci se extrakcí nebudeme zabývat. Pro účely doporučení by však bylo potřeba klíčová slova přesněji vyextrahovat, proto jsou tyto slova použita, bohužel, pouze jako druhotné pravidlo. I přesto nám klíčová slova dokážou některé pořady dobře propojit a plní tak alespoň trochu svůj účel.

4.3 Příprava databáze

V systému společnosti je používána obsáhlá databáze s velkým množstvím entit. Většina těchto entit není pro RecTV důležitá a bylo nutné zaměřit se pouze na potřebná data. V ER diagramu na obr. 4.1 můžeme vidět, jak vypadá (pro RecTV) důležitá část obsáhlé databáze a jaké vazby jsou mezi entitami. Pro běh recommendation enginu

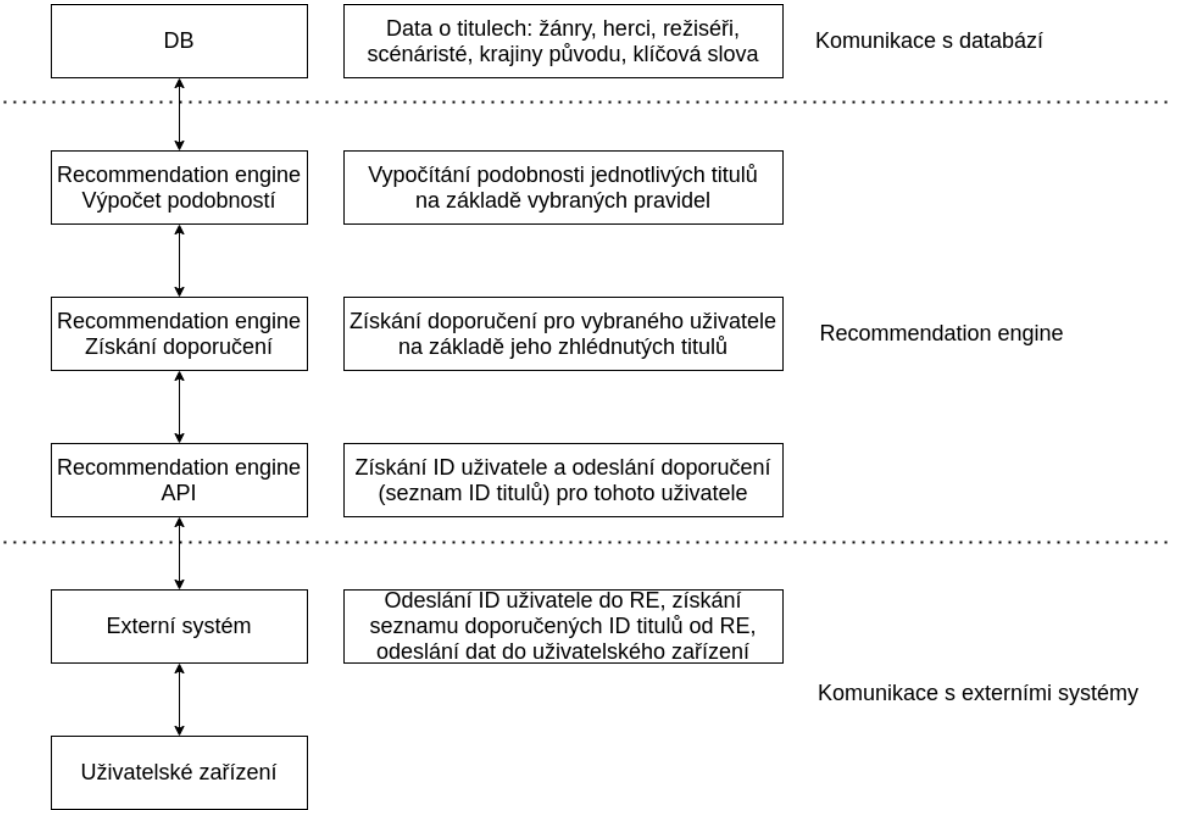
bylo třeba použít a přetransformovat pouze entity vyjadřující vztah mezi pravidly a tituly, tj. *MdbTitleGenres*, *MdbTitleOrigins*, *MdbPersonRoles* a *MdbTitleKeywords*. Při transformaci jsem zanedbal nepotřebné atributy původních entit a vytvořil tak 4 nové entity. Pro databázi RecTV není potřeba udržovat vztahy mezi entitami, jelikož se z databáze bude pouze číst. Proto bylo možné vynechat tabulku *MdbTitles* a vztahy mezi původními entitami udržovat pouze pomyslně (u každé nově z vytvořených entit stále udržujeme atribut *mdbTitle*, díky němuž známe původní propojení pravidla s titulem).



Obr. 4.1 ER diagram věnované části databáze IPTV systému

4.4 Architektura

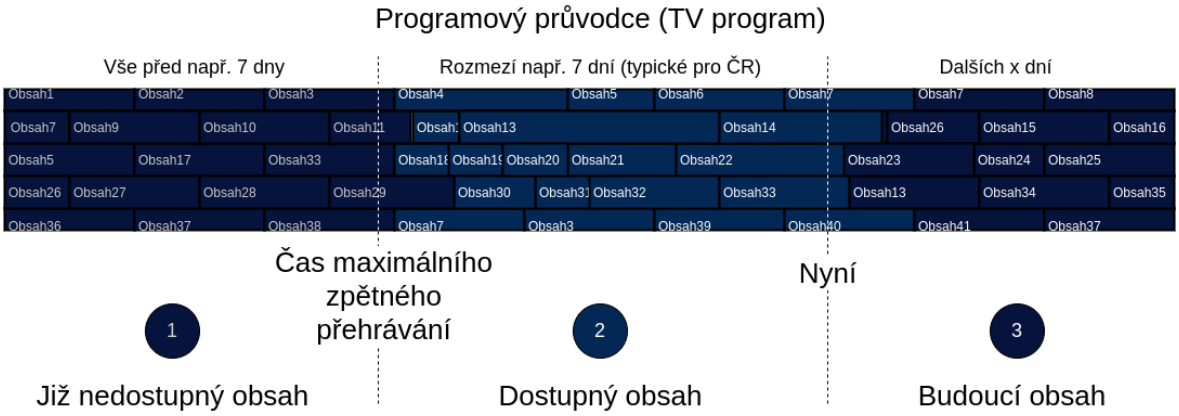
Aby doporučovací systém mohl získávat doporučení, musí nejprve vypočítat podobnosti mezi jednotlivými tituly. K tomu však potřebuje důležitá data, nad kterými bude výpočet probíhat. Je nutné komunikovat s databází, která tato data uchovává. Po získání důležitých informací bude první část recommendation enginu vypočítávat podobnost titulů na základě navrhovaných pravidel. Druhá část systému se zaměří na získání doporučení pro uživatele podle jeho zhlédnutého obsahu. Nesmíme však zapomenout na to, že je potřeba tato doporučení poskytovat dalším systémům. Proto je nutné vytvořit API komunikující s dalšími servery služby IPTV. Schéma architektury můžeme vidět na obr. 4.2.



Obr. 4.2 Architektura recommendation enginu

4.5 Problém s dostupností obsahu

IPTV služby, jak již bylo zmíněno v kapitole 1, poskytují obsah statický (VOD), ale také dynamický (DVR / živé vysílání). Kvůli tomu, že tituly v rámci vysílání nezůstávají v nabídce navždy, je potřeba vyřešit problém s tímto unikajícím obsahem a nedoporučovat pro uživatele již nepřehratelné tituly. V systému bylo navrženo řešení, které je popsáno v podkapitole 5.3.



Obr. 4.3 Dostupnost obsahu v rámci IPTV vysílání – zobrazení v TV programu

5 Vlastní implementace

Celý systém je naprogramován tak, aby byl do budoucna lehce škálovatelný. Dokáže sám komunikovat s jinou instancí recommendation enginu a z této instance získávat výsledky výpočtů. V rámci implementace proběhlo několik iterací, které se projevovali přidanými optimalizacemi. Systém RecTV má za úkol načíst data z databáze a získat doporučení pro uživatele na základě podobnosti obsahu, také musí být schopen komunikovat s externími servery za účelem doporučení uživateli. V podkapitole 5.1 je popsáno, jaká data byla převzata z původní databáze. Také je zde uvedeno, jakým způsobem RecTV komunikuje s touto databází a získává z ní důležitá data, která dále zpracovává. Důležitou částí RecTV je výpočet podobností mezi jednotlivými tituly. To, podle čeho se podobnost počítá, je přiblíženo v sekci 5.2. V předposlední podkapitole je popsáno, jak samotný program RecTV funguje a jak probíhá proces přeměny dat na výsledná doporučení uživateli. Zabezpečení serveru a přístupů do databáze je popsáno v poslední podkapitole.

5.1 Práce s databází

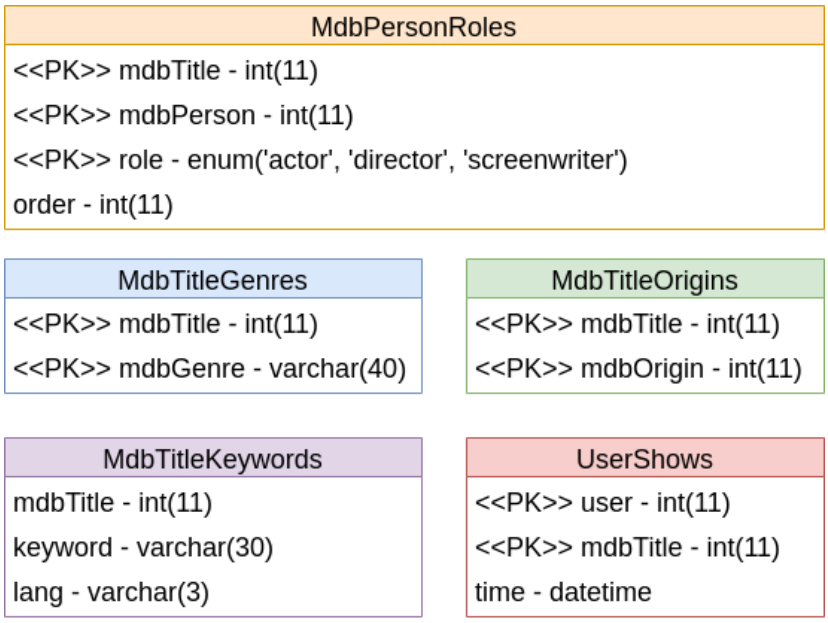
Databáze MariaDB, která obsahuje veškerá data o jednotlivých titulech, je zpracována z velké databáze společnosti. Z původního výřezu databáze (obr. 4.1) jsem vytvořil tabulky *MdbTitleGenres* pro žánry, *MdbTitleOrigins* pro krajiny původu, *MdbTitleKeywords* pro klíčová slova a *MdbPersonRoles* pro herce, režiséry a scénáristy. Pomocí ovladače pro MySQL a balíčku GORM pro ORM v Go je k databázi přistupováno přímo z recommendation enginu. Konfigurace k přístupu do databáze je uložena v souboru *src/config.yml*. Cesta ke konfiguračnímu souboru je povinným parametrem hlavního programu. Příklad spuštění programu je uveden v sekci 5.4.1.

V původních datech byl problém s duplikáty osobností a bylo tak potřeba provést konverzi duplikovaných dat, jelikož by kvůli duplikátům nemohl na základě tří pravidel RecTV přesně vypočítat podobnosti. V tabulce *MdbPersons* existovaly osoby, které byly nalezeny ve více než 1 záznamu. Bylo potřeba získat všechny identifikátory těchto duplikátů, vybrat např. první z nich a nahradit výskyty ostatních identifikátorů v záznamech tabulky *MdbPersonRoles* vybraným identifikátorem. Tato operace byla časově velice náročná, pro každý záznam osoby musel být vytvořen vlastní SQL dotaz *UPDATE*, který byl vytvářen ve for cyklu. Po přidání INDEXu na sloupec *mdbPerson* již aktualizace velkého množství duplikátů proběhla v rámci několika jednotek minut. Tato konverze byla provedena se zanedbáním výskytu 2 či více osob se stejným jménem a příjmením.

Pro výpočet podobností bylo jednodušší používat číselné hodnoty než zpracovávat textové řetězce (např. spojování příjmení a jména apod.). Proto RecTV používá pi-

votové tabulky, které vyjadřují vazby titul-osobnost a titul-krajina původu. Jelikož klíčová slova a žánry jsou vyjadřovány jako jedno slovo, nebylo potřeba je nijak dále zpracovávat a pro výpočty mohly být použity textové řetězce. Výsledná databáze, kterou můžeme vidět na obrázku 5.1, neobsahuje žádné vazby mezi entitami, protože jsou vazby s tituly jednoznačně definovány obsaženými atributy.

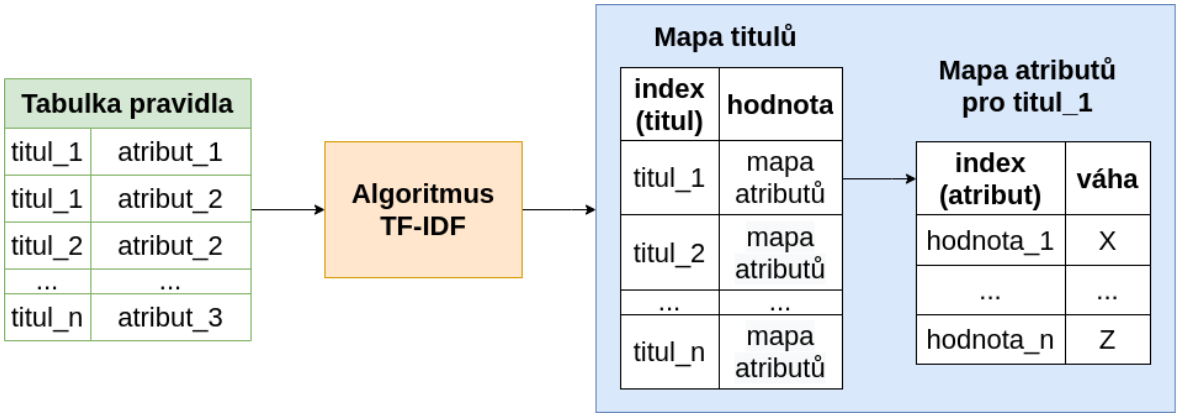
Proto, abychom mohli systém RecTV otestovat, bylo potřeba vytvořit tabulku, jež obsahuje všechny uživateli zhlédnuté pořady – *UserShows*. Data v tabulce jsou kopií ostré verze databáze sledovanosti, identifikátory uživatelů nelze nijak spojit s žádnou konkrétní osobou. RecTV by za žádných okolností neměl znát důvěrné informace o uživateli.



Obr. 5.1 Tabulky databáze RecTV

5.2 Úpravy před výpočtem podobností

Jak již bylo zmíněno v kapitole 4.2, bylo nutné získat důležité atributy, podle kterých se vypočítá podobnost jednotlivých titulů. V každém z případů bylo nutné získat nějakou váhu každého výskytu daného pravidla. Tato váha byla vypočítána pomocí algoritmu TF-IDF, jež je popsán v sekci 2.2.1. Pomocí takto získaných číselných hodnot (vah) bylo konečně možné získat podobnost pomocí algoritmu míry podobnosti. Pro tento systém byla využita kosinová podobnost (cosine similarity). Tato metoda dokáže získat podobnost výpočtem kosinu úhlu vektorů vytvořených z atributů daného titulu. Všechny vektory jsou však rozděleny podle jednotlivých pravidel, abychom mohli získávat nejen celkovou podobnost mezi tituly, ale také podobnost podle jednotlivých pravidel (např. kvůli kategorizaci).



Obr. 5.2 Transformace dat s použitím algoritmu TF-IDF pro výpočet vah

Na obrázku 5.2 můžeme vidět, jak se např. z tabulky *MdbTitleGenres* vytvoří mapa (hash tabulka s klíčem a hodnotou) titulů, kde klíčem je titul a hodnotou je mapa atributů (v tomto případě žánrů). Tato mapa atributů je přípravou pro výpočet podobnosti algoritmem cosine similarity – každý titul má svůj seznam atributů s vypočítanou váhou.

5.3 Provedené optimalizace

Jelikož jsem chtěl získat podobnost všech titulů, bylo nutné vypočítat kombinace titulů každý s každým. Tato kombinace je vyjádřena vztahem 5.1.

$$x = n \cdot (n-1)\,, \tag{5.1}$$

kde x je počet kombinací a n je počet všech titulů.

Po hlubším zamyšlení jsem si uvědomil, že při výpočtu podobnosti nezáleží na pořadí, podobnost titulu A s B je stejná jako podobnost titulu B s A. Po upravení vztahu kosinu úhlu ϕ, který svírají 2 vektory, získáme pouze operace násobení. Tyto operace jsou komutativní, protože platí rovnost popsána vztahem 5.2.

$$cos\,(\phi) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} = \frac{\sum_{i=1}^{n} B_i \cdot A_i}{\sqrt{\sum_{i=1}^{n} B_i^2} \cdot \sqrt{\sum_{i=1}^{n} A_i^2}} = \frac{B \cdot A}{\|B\|\|A\|} \tag{5.2}$$

Protože nezáleží na pořadí operandů, lze využít kombinací ze vztahu z kombinatoriky. Tento výpočet je vyjádřen vztahem 5.3.

$$C\,(k,n) = \frac{V\,(k,n)}{k!} = \frac{n!}{(n-k)! \cdot k!}, \tag{5.3}$$

kde k vyjadřuje počet členů kombinace a n počet prvků. Jelikož hledáme pouze dvoučlenné kombinace titulů, můžeme za k dosadit číslo *2* ve vztahu 5.4.

$$C\left(2,n\right)=\frac{n!}{(n-2)!\cdot 2!)}=\frac{n\cdot(n-1)\cdot(n-2)!}{2\cdot(n-2)!}=\frac{n\cdot(n-1)}{2} \tag{5.4}$$

Zjišťujeme, že jsme získali pouze polovinu kombinací oproti předchozímu případu.

Abychom se však přiblížili k problematice vysílání IPTV služby, můžeme výpočet podobnosti ještě o něco zoptimalizovat a vyřešit tak také problém s dostupností pořadů. Úkolem RecTV však není zjišťovat, jaké tituly jsou či nejsou dostupné uživateli, a proto byla tato logika vysunuta mimo tento systém. Pomocí konfigurace však lze nastavit endpoint k serveru, který vrátí seznam dostupných titulů. Pokud je tento endpoint nastaven, RecTV před samotným výpočtem požádá tento server o dostupné pořady. Odpověď serveru musí být přesná podle definice, jinak bude ignorována a bude počítáno se všemi tituly. Pokud budeme počítat s tím, že počet všech titulů je prozatím cca 130 tisíc a počet dostupných pořadů v průměru 15 tisíc, snížíme výsledný počet kombinací skoro na pětinu. Lze to dokázat výpočtem 5.5.

$$\frac{130000\cdot 129999}{2}\doteq 8.45{\cdot}10^9\approx 4.6\cdot 1.84{\cdot}10^9\doteq 4.6\cdot(130000\cdot 15000)-\sum_{i=1}^{15000}i, \tag{5.5}$$

kde levá strana je dosazena do vztahu kombinací a pravá do analyticky vytvořené rovnice.

RecTV tedy dokáže vypočítat podobnosti všech dvojic titulů bez zbytečných výpočtů navíc. Pokud nebudeme chtít používat poslední optimalizaci s dostupnými pořady, stačí vynechat konfiguraci endpointu k serveru pro získání seznamu identifikátorů (nebo ze serveru špatně odpovídat).

5.4 Program RecTV

Implementace programu je rozdělena na 3 hlavní části – výpočet podobnosti, získávání doporučení a server s API. V následujících podpodkapitolách je popsáno spuštění programu RecTV, jaké procesy po startu programu probíhají a jak jsou řešeny jednotlivé hlavní části.

5.4.1 Spuštění RecTV

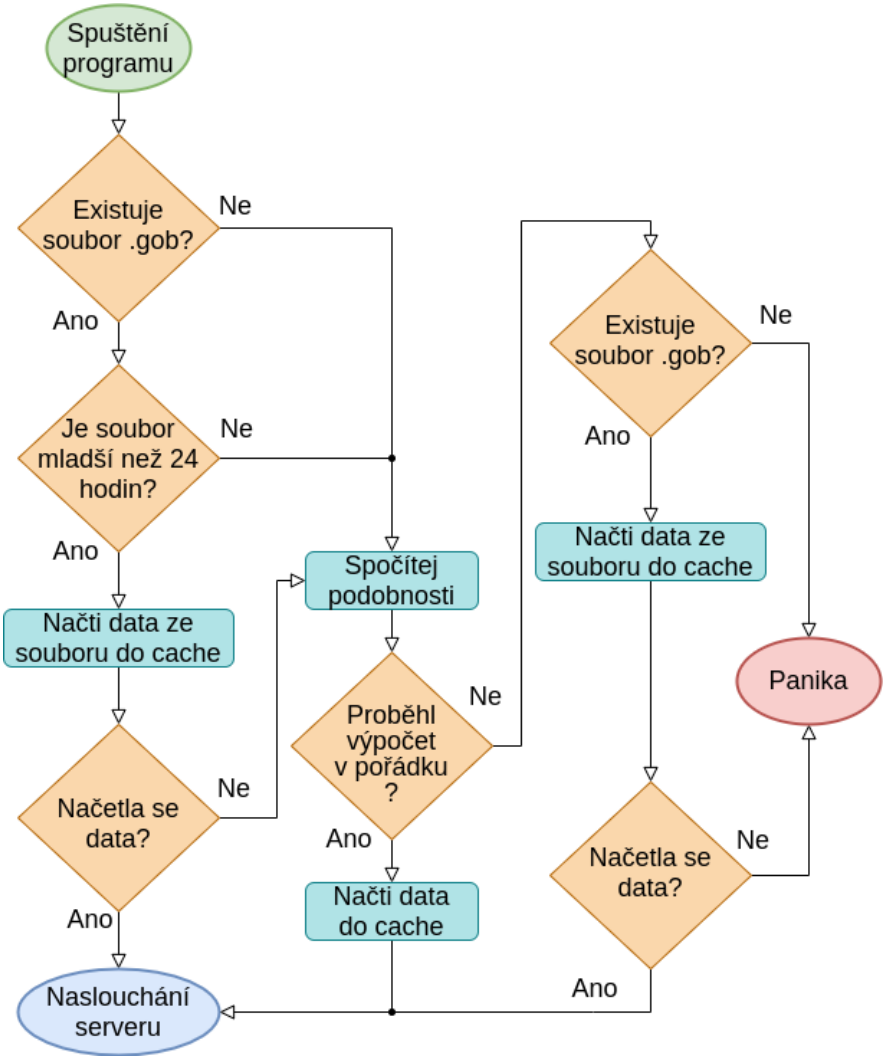
Před samotným spuštěním programu je nutné vytvořit konfiguraci ve formátu YAML (v této práci je již vytvořena), jelikož je cesta k souboru s touto konfigurací povinným parametrem spustitelného programu. V konfiguračním souboru jsou totiž informace k přístupu k databázi, kde jsou uložena data o titulech. Samotný program poté můžeme

spustit příkazem `./recTV -config="config.yml"`.

Poté, co se program spustí a správně se načte konfigurace, začne probíhat výpočet podobnosti, který je popsán v další podsekci. Po získání podobností je nastartován server, který naslouchá na požadovaném portu (taktéž obsaženo v konfiguračním souboru), tento proces můžeme vidět na obrázku 5.3.

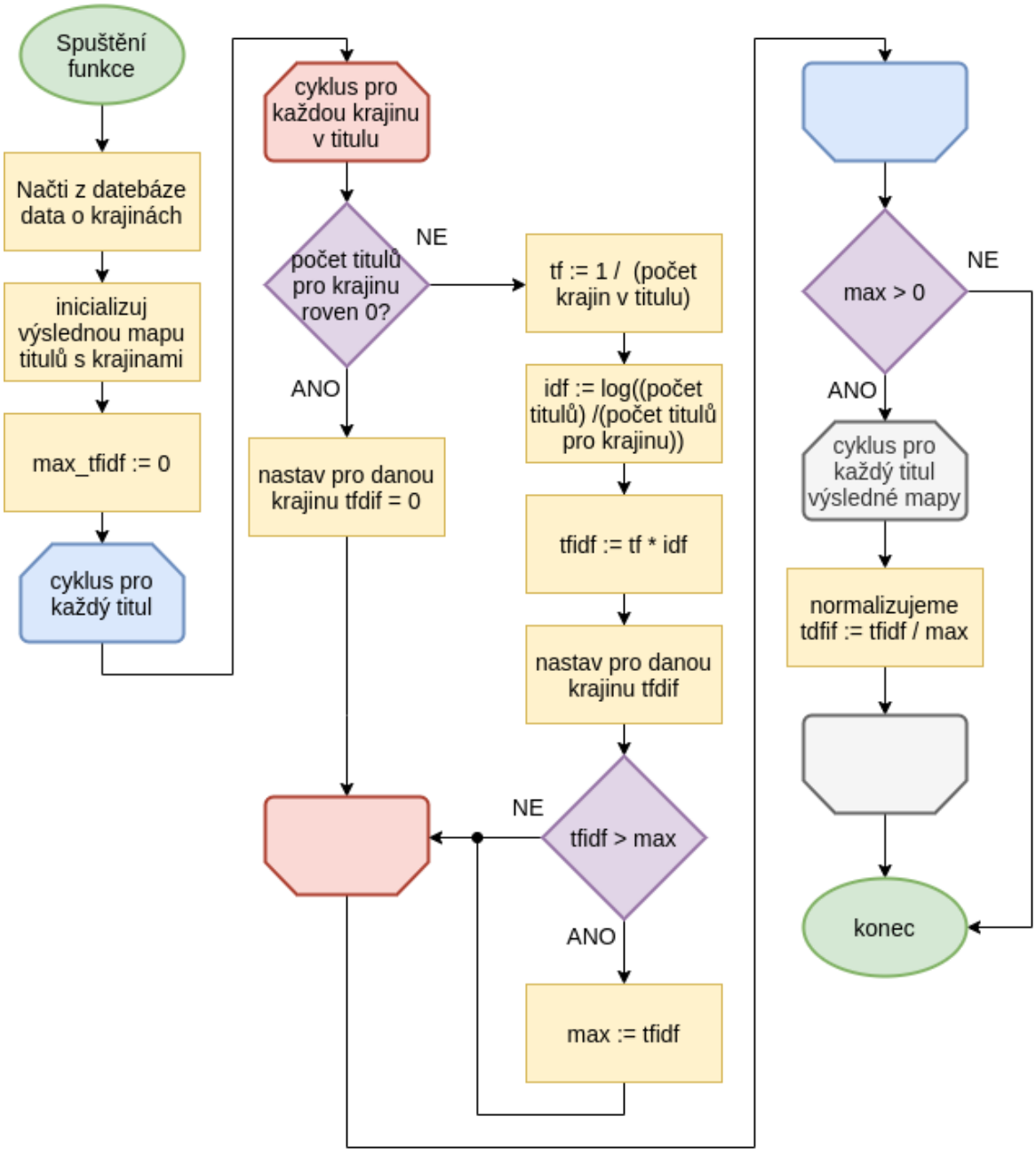
5.4.2 Výpočet podobnosti

Při spuštění programu se zjišťuje, zda-li existuje uložený GOB soubor, který obsahuje výsledky výpočtů podobnosti. Tento soubor nesmí být starší jednoho dne, jinak se spouští nový výpočet. Uložení dat do GOB souboru probíhá ihned po vypočítání poslední podobnosti titulů. Soubor je tu pro případ, kdy nastane nějaká chyba a bude restartován celý program. Výpočet podobností je časově náročný a RecTV by nemohl doporučovat ihned po restartu.



Obr. 5.3 Vývojový diagram výpočtu podobnosti

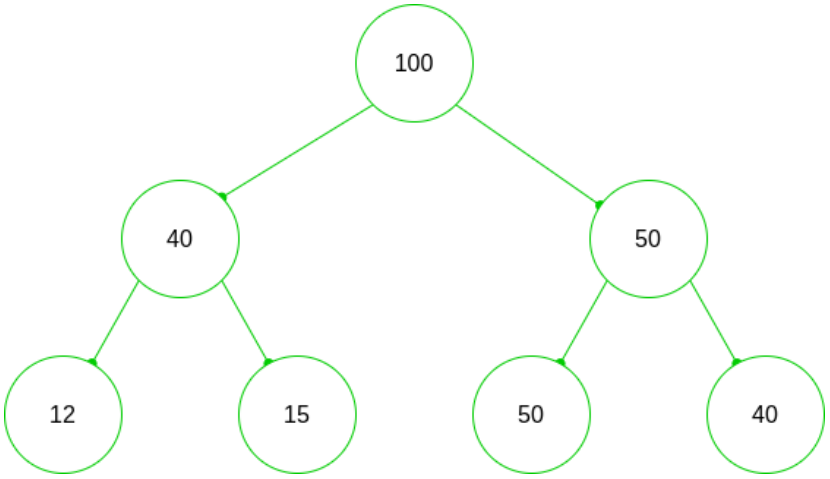
V případě, kdy se spustí výpočet podobnosti, získá se nejprve seznam všech identifikátorů titulů, pro které podobnost chceme vypočítat. Poté se tento seznam předá do metody `Process` v balíčku *src/matrix*, která pomocí Go kanálů [24] postupně naplňuje seznam pro zpracování a zároveň zpracovává již naplněné seznamy. Vše pomocí Go rutin [22] probíhá paralelně. Také můžeme ze vstupních parametrů nastavit, kolik workerů [26] bude zpracovávat tento výpočet. Defaultní hodnota počtu pracovníků je podle počtu procesorových jader. Aby se veškeré výpočty dokončily, než se spustí ukládání, musí existovat tzv. WaitGroup [25], která zabrání v pokračování programu, dokud všichni pracovníci v rutinách nedokončí svou práci.



Obr. 5.4 Vývojový diagram výpočtu TF-IDF pro krajiny

Jelikož se výpočet podobnosti rozdělil na části podle pravidel, bylo nutné vytvořit rozhraní, která jsou implementována v balíčku *similarityHandlers*. Pro každý z těchto handlerů lze nakonfigurovat proměnnou, která funguje jako spouštěč/vypínač daného pravidla. Pokud je dané pravidlo zapnuté, před samotným výpočtem se spustí načítání souvisejících dat o titulech. Tato data jsou vytažena z databáze a je nad nimi proveden algoritmus TF-IDF, který ohodnotí relevanci hodnoty atributu. Tento proces probíhá v balíčcích pojmenovaných podle pravidel (např. *genres*). Data jsou po tomto procesu uložena do map s jednotlivými tituly a je možné s nimi dále počítat. Jak výpočet pomocí TF-IDF probíhá můžeme vidět na vývojovém diagramu na obrázku 5.4.

Po ohodnocení atributů lze vypočítat podobnost pomocí cosine similarity. Prvně se vytvoří pomocná data (vektory), kde se vyplní chybějící položky obou zpracovávaných titulů. Poté se vypočítává střední hodnota obou vektorů a v případě, že je kladná, vypočítá se skalární součin těchto vektorů. Po vydělení skalárního součinu střední hodnotou získáme výslednou podobnost v rozmezí od 0 do 1. Abychom neudržovali výsledek jako číslo s plovoucí desetinnou čárkou, vynásobíme jej číslem 100 a přetypujeme na kladné celé číslo. Zdrojový kód cosine similarity je uložen v adresáři *src/cosine*. Obsahuje metody `Pad()`, `Norm()` a `DotProduct`. Metoda `Pad` vytváří vektory 2 titulů, aby oba obsahovali stejné prvky pro jednodušší výpočet. Pokud nejsou data jednoho titulu obsažena v druhém, jsou nastavena hodnotou 0. U obou nově vytvořených vektorů je potřeba získat jejich normu (velikost), která je definována jako odmocnina skalárního součinu vektoru samého se sebou [37]. Skalárním součinem těchto norem získáme střední hodnotu, pokud je střední hodnota větší než 0, je možné vypočítat skalární součin dvojice vektorů metodou `DotProduct()`. Tato metoda v cyklu jednoduše projde oběma vektory a sčítá násobky hodnot na stejném indexu. Poté už stačí skalární součin vydělit střední hodnotou a výsledek vynásobit číslem 100.



Obr. 5.5 Příklad haldy Max-heap – upraveno z [38]

Vytvořil jsem datovou strukturu heap (halda), díky které není potřeba zpracovávat

nejpodobnější tituly, jelikož se do struktury ukládají tituly od nejvyšší po nejnižší hodnotu podobnosti (viz. příklad na obrázku 5.5). Také je zde konfigurovatelná velikost heapu, která je defaultně nastavena na 1000 titulů, aby se omezila paměťová náročnost uložených podobností.

5.4.3 Získávání doporučení

Jakmile máme v cachi načteny všechny podobnosti titulů, můžeme pomocí volání na server spustit získání doporučení pro uživatele. Tento proces probíhá tak, že se z databáze načtou všechny zhlédnuté pořady uživatele, které nejsou zhlédnuty příliš dávno (omezeno konfigurovatelným počtem).

Pro každý zhlédnutý pořad vytvoříme mapu s indexem identifikátoru podobného titulu a hodnotou 1 pro jeho relevanci – nazveme ji mapou relevancí. Navíc si vytvoříme mapu s indexy zhlédnutých pořadů, hodnotou 3 u posledních tří zhlédnutých pořadů, hodnotou 2 pro další 3 pořady a hodnotou 1 pro každý další pořad jako výsledný poměr doporučení – nazveme ji mapou poměrů.

Pokud při přidávání podobných titulů dalšího pořadu narazíme v některé z map relevancí na stejný titul, zvýšíme jeho relevanci v této mapě. V případě, že mezi zhlédnutými pořady jsou již tituly nějak podobné, zvýšíme poměr doporučení v mapě poměrů.

Zbývá již jen vytvořit výsledné pole identifikátorů doporučených titulů. Pomocí mapy poměrů získáme koeficienty, kterými budeme ořezávat mapu relevancí. Tyto koeficienty získáme hodnotou z mapy poměrů vydělenou sumou hodnot poměrů ve všech mapách poměrů a následným vynásobením délky mapy relevancí pro daný zhlédnutý pořad. Z každého pole relevancí vybereme ty tituly, které nemají hodnotu relevance rovnu 1, ty vložíme ihned do pomocné mapy doporučených titulů. Poté podle získaných poměrů pro každý zhlédnutý titul vybereme zbylé podobné tituly a vložíme je taktéž do této pomocné mapy. Pokud již v mapě některý titul je, neměníme již jeho hodnotu relevance. Po získání celé mapy doporučených titulů již pouze převedeme tuto mapu na pole a seřadíme ho podle obsažených relevancí. Nakonec již pouze vrátíme výsledné pole doporučení do serverové části.

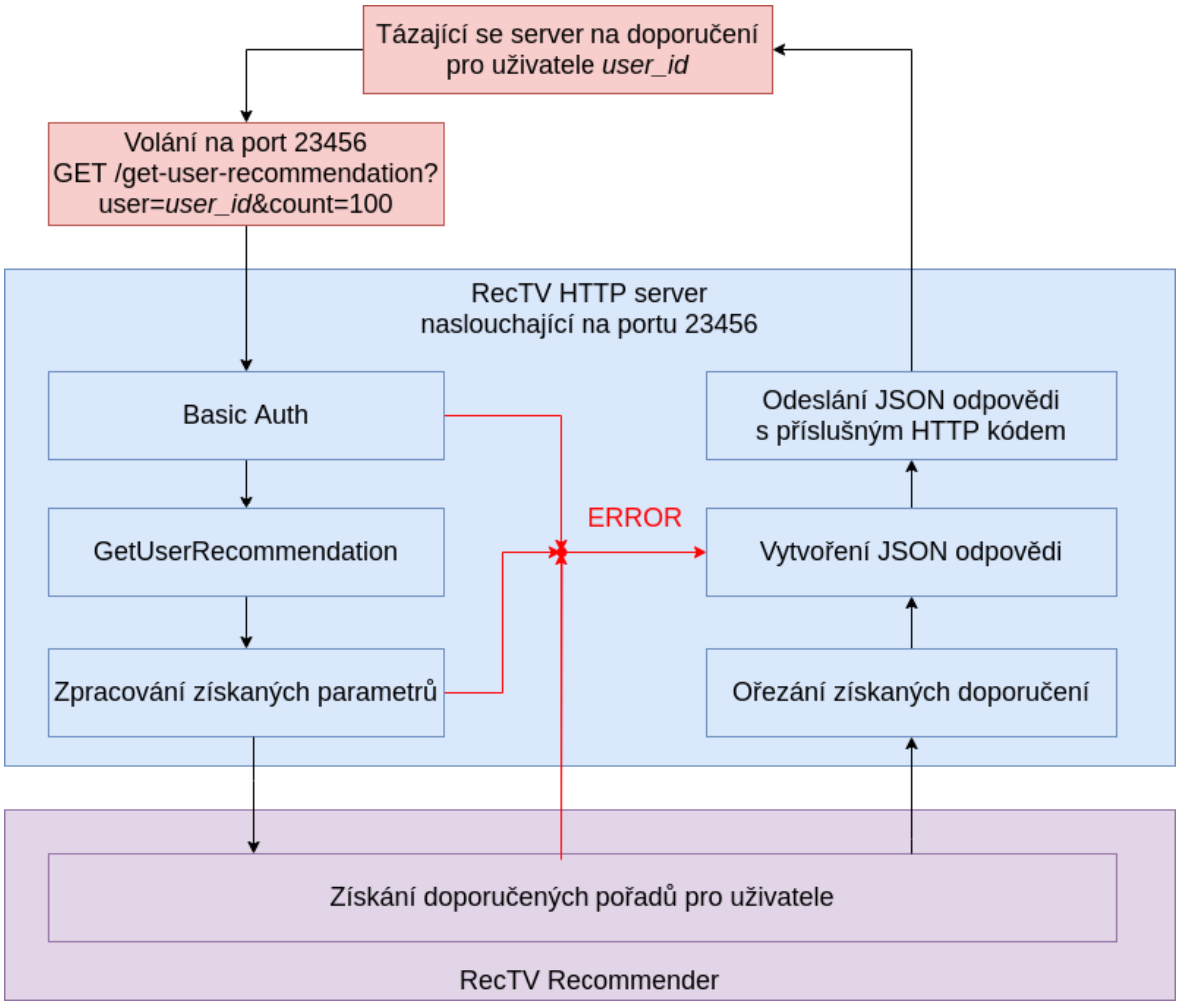
5.4.4 Serverová část

K implementaci serveru byl využit Go balíček *net/http* [28]. Je to jednoduchý HTTP server, který po spuštění naslouchá na požadovaném portu a ihned dokáže zpracovávat requesty od externích serverů. V souboru *src/server/server.go* je tento server naimplementován.

Jedinou potřebnou metodou k získání doporučení pro uživatele je `getUserRecommendation()`. Voláním endpointu `/get-user-recommendation` spustíme

provádění této metody, která zpracuje vstupní parametry requestu (typicky identifikátor uživatele a případně počet doporučení) a spustí doporučovací část programu (popsáno v podpodkapitole 5.4.3). Po získání pole identifikátorů doporučených titulů je, v případě získaného parametru počtu doporučení, nutné toto pole ještě ořezat na požadovanou délku. Poté se již vytvoří výsledný JSON, který je odeslán jako odpověď na dotaz. JSON odpověď je v následujícím formátu: `{"status": 1, "title_ids": [id_1, id_2, id_3]}`. Tento proces je znázorněn na obrázku 5.6.

V případě chyby se neodesílá atribut `title_ids`, ale místo toho přibývá atribut `error` a mění se HTTP kód odpovědi. Výsledný JSON pak vypadá následovně: `{"status": 0, "error": "Server has internal problems."}` s HTTP kódem 500 (Internal server error [39]).

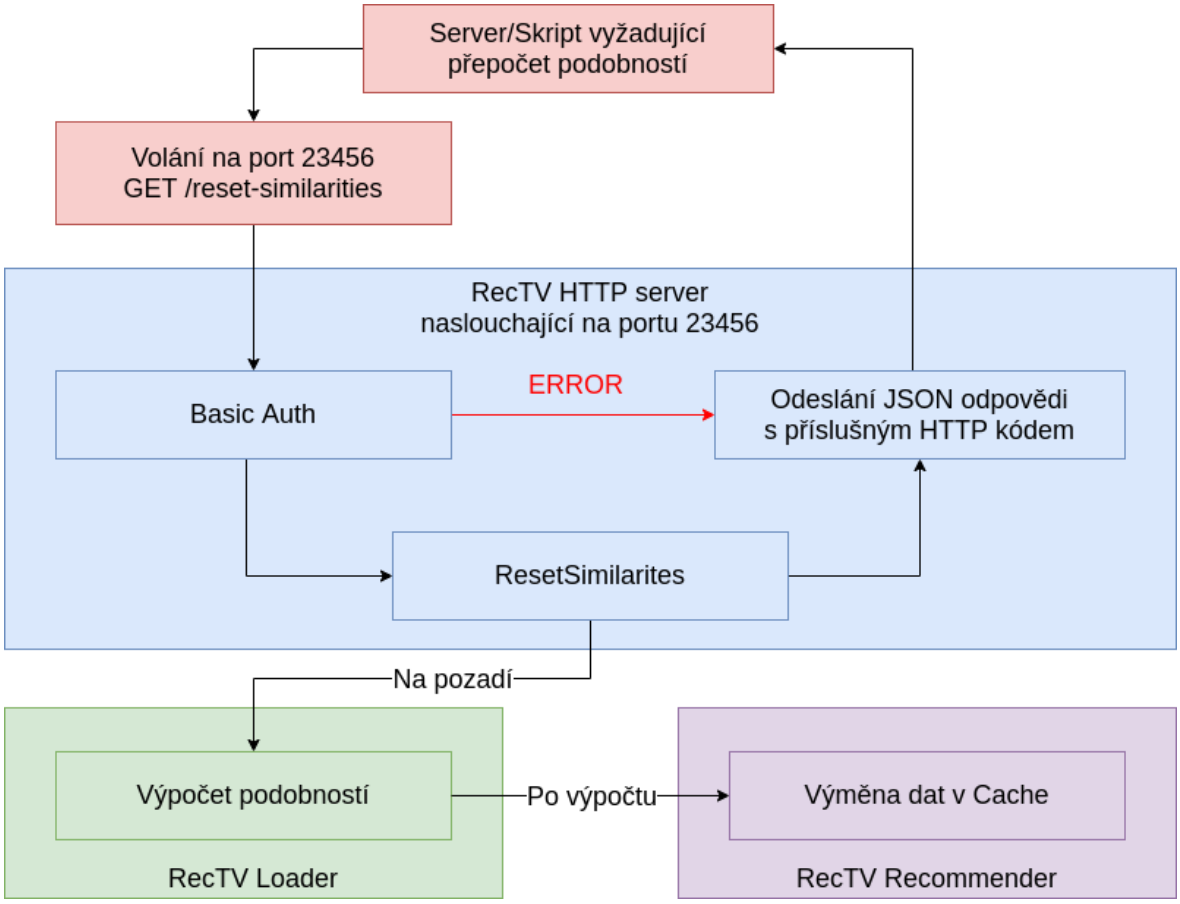


Obr. 5.6 Ukázka průběhu získání doporučení

Existuje zde také endpoint `/status`, který v případě běžícího serveru odpoví pouze *"OK"* a HTTP kódem 200 (OK [39]). Typicky je toto volání důležité pro dohledové systémy. Pokud by odpověď od serveru nezískali, musel nastat nějaký problém a je nutné program restartovat či znovu spustit.

Pro testování byl vytvořen endpoint `/test_recommendation`, který spouští metodu `getTestRecommendation()`. Tato metoda nejprve zpracuje tělo requestu a získá tak seznam identifikátorů, které znázorňují zhlédnuté pořady nějakého uživatele. Poté probíhá získávání doporučení stejným způsobem, jako je tomu u `/get-user-recommendation`.

Někdy bude potřeba znovu vypočítat podobnosti jednotlivých titulů, proto byl vytvořen endpoint `/reset-similarities`, který na pozadí spustí samotný výpočet a odpoví serveru, že se proces výpočtu spustil. Výpočet, stejně jako při spuštění programu, opět probíhá paralelně a po ukončení výpočtu jsou data vyměněna v cachi (viz. obrázek 5.7.



Obr. 5.7 Ukázka průběhu přepočítávání podobností titulů

Aby systém nebyl dostupný pro všechny, je přístup do systému zabezpečen jednoduchým ověřením (Basic Auth) – dvojice uživatel-heslo (konfigurovatelné).

5.5 Zabezpečení systému

Jelikož systém využívá databázi, mohlo by se stát, že dojde podvržením SQL dotazů k získání uložených dat. Jediná možnost, jak by se dal systému podvrhnout nějaký SQL dotaz, by bylo volání na server pro získání doporučení pro uživatele. Server přijme

parametry requestu, očekávané parametry jsou ihned po přijetí zpracovány a převedeny na celé číslo. Pokud by se tedy někdo pokusil podvrhnout zde nějaký SQL dotaz, nepodaří se mu to a server odpoví chybou. I kdyby se cokoliv pokazilo a přece jen se nějakým způsobem dostal podvržený SQL dotaz až ke zpracování dotazu do databáze, jsou tyto dotazy ošetřeny proti SQL injection pomocí escapování parametrů (využívá GORM [40]).

A jak již bylo zmíněno v podpodkapitole 5.4.4, server je zabezpečen alespoň pomocí jednoduchého ověření přístupu Basic Auth, které zastřešuje balíček *net/http* v o. Pokud je request na server zavolán se špatným přístupovým jménem a heslem, je ihned spojení ukončeno chybovou hláškou (i v případě chybějících údajů).

6 Testování

Tato kapitola se zabývá testováním programu RecTV. Každá fáze testování je velmi důležitou součástí vývoje, již z počátku vývoje můžeme díky testování zmírnit výskyt jakýchkoli chyb, které by později mohly způsobovat velké problémy. Aby bylo možné předejít řešení takových problémů, lze v programovacím jazyce Go využít integrovaného testovacího nástroje. Výhody a příklady použití tohoto nástroje jsou uvedeny v sekci 6.1. Nesmí být zapomenuto na testování celého programu jako celku. Jelikož je testování doporučovacích systémů velice složité, nejjednodušší cestou je křížová validace, která se obejde bez reálných osob, výsledky tak nejsou zkresleny a testování zůstává objektivní. Na to, jakých výsledků dosáhl RecTV a jak by se systém dal vylepšit, se podíváme v poslední sekci této kapitoly.

6.1 Unit testy

Jak již bylo uvedeno v úvodu kapitoly, programovací jazyk Go obsahuje integrovaný testovací nástroj. Vytvoření jednotlivých testů je umožněnu jednoduchým způsobem, při vytváření nového souboru *.go* je zapotřebí také vytvořit odpovídající testovací soubor se stejným názvem s přidáním řetězce *_test*. Pokud tedy vytvoříme soubor *heap.go*, je vhodné také vytvořit soubor *heap_test.go*, který bude automaticky propojen s tímto souborem. Poté již stačí v příslušném adresáři spustit pouze příkaz `go test`, který spustí všechny testy automaticky. Lze také spustit pouze chtěné testy a to pomocí příkazu `go test -run Create`, který spustí pouze testy obsahující slovo *Create* (např. *TestCreateTable*).[41]

Pro každý soubor byly vytvořeny odpovídající testovací soubory, které mají za úkol vyloučit chyby jednotlivých komponent. Nejspíše nejdůležitější testy jsou obsaženy v balíčku *matrix*, které ověřují správnost průběhu výpočtu podobnosti a uložení dat do cache. Druhou důležitou částí testování jsou testy v balíčku *cosine*, které ověřují správnost výpočtu cosine similarity. Také baliček *server* bylo nutné otestovat, abychom ověřili správnost autentizace a také odpovědí serveru.

V následujícím seznamu se můžeme podívat na jednotlivé kroky průběhu testování funkce `IntPad()` (vytvoření vektorů se stejnými indexy, aby byly stejně dlouhé) v balíčku *cosine*:

1. Start testu TestIntPad(t *Testing) v balíčku cosine
2. Vytvoříme 2 mapy integerů s hodnotou váhy, které odpovídají atributům titulů
3. Spustíme funkci IntPad() a předáme mapy jako parametry
4. Získáme 2 nově vytvořené vektory (mapy)

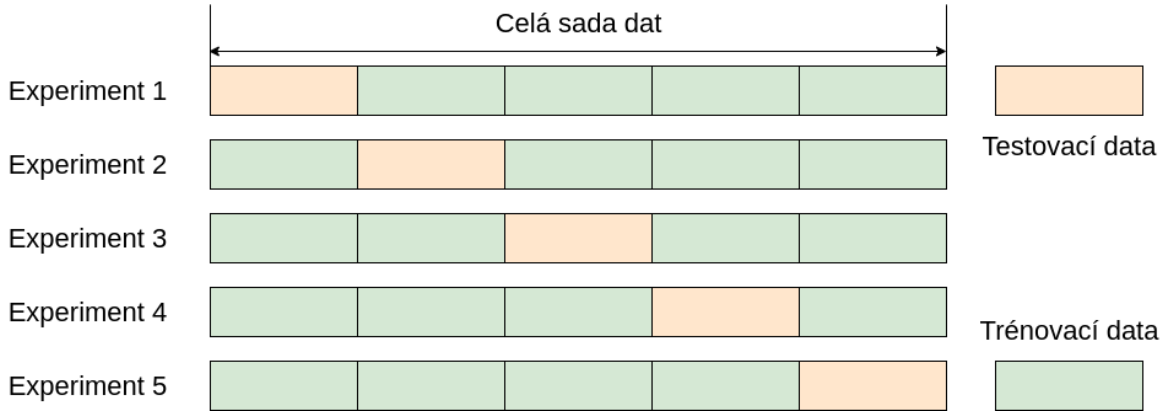
5. Porovnáme délku obou vektorů, pokud je rozdílná, vypíšeme chybu
6. Pro každý vektor zkontrolujeme výsledné hodnoty
7. Pokud hodnoty nesouhlasí, vypíšeme chybu
8. Konec testu.

Samotný test je napsán tak, aby pokryl co nejvíce možných problémů. U zobrazeného testu vidíme, že vytvořené vektory musí být stejně dlouhé a musí obsahovat správné hodnoty.

Takovým způsobem lze pokrýt velkou část chyb, které by se mohly během vývoje vyskytnout. Je však velice těžké se takovému způsobu testování přizpůsobit. V první řadě je potřeba vytvořit testy a ošetřit výstupy funkcí – po spuštění nesmí testy projít, teprve poté je možné psát samotnou funkcionalitu a tu následně otestovat. Hlavním problémem, se kterým jsem se setkal, byla změna již napsaného kódu či připsání kódu nového. Občas se stane, že člověk zapomene testy před změnami upravit a poté nemá pokrytou část svého kódu. I přes to je však dobré takový typ testování zavést, jelikož je lepší pokrytí nějakého kódu, než-li žádného.

6.2 Cross validation

Křížovou validaci jsem vybral z prostého důvodu, testovat na reálných uživatelích prozatím není možné, protože bych musel vytvořit nějaké prostředí, v němž by tito uživatelé mohly systém RecTV vyzkoušet – seznam identifikátorů titulů by jim nejspíše nic moc neřekl. Navíc společnost prozatím nemá žádný hodnoticí systém, díky kterému bychom dokázali získat zpětnou vazbu.



Obr. 6.1 Výběr testovacích dat u křížové validace – upraveno z [42]

Abychom tento typ testování mohli zavést, bylo potřeba získat nějaké záznamy zhlédnutých pořadů uživateli. Tato testovací data jsou uložena v databázi v tabulce *UserShows* a obsahují vždy uživatelem zhlédnuté pořady a přibližný čas, kdy uživatel titul

zhlédl. Při cross validaci je nutné datový set rozdělit na 2 části. První, menší částí jsou testovací data, tou druhou pak data trénovací. Nelze však po jednom spuštění testu ihned soudit, zda-li jsou výsledky dobré či nikoli. Je nutné přidat iterace (experimenty), během kterých se postupně prostřídají sady trénovacích i testovacích dat. Na obrázku 6.1 je znázorněn příklad výběru datových sad.

Kvůli tomuto testování jsem do RecTV musel přidat metodu, která místo hledání zhlédnutých pořadů v databázi využije data poslaná z testovacího programu. Získání doporučení jinak proběhne stejným způsobem, jako je tomu při normálním doporučování pro uživatele. Po získání doporučení na základě zaslaných dat bylo nutné zkontrolovat, jak dobře se recommendation engine trefil do testovacích dat.

Pro testování byl použit skript *rec-cross-checker*. Skript má 2 možnosti spuštění – první pro připravení datasetu, druhý pro validaci. Pro přípravu datasetu můžeme spustit skript pomocí příkazu `./rec-cross-checker prepare` *`csv_s_porady dataset`*, kde vstupní CSV soubor obsahuje data vytáhnutá z tabulky *UserShows* a výstupem je dataset pro testování uložený v souboru *dataset*. Druhý způsob `./rec-cross-checker validate -h=`*`url:port dataset`*, který spustí validaci recommendation enginu s daty z předaného datasetu. Výstupem v příkazové řádce je pak výsledný počet správných doporučení ze všech testovacích zhlédnutých titulů daného uživatele. Po ukončení skriptu je zobrazen průměrný výsledek doporučování v procentech.

6.3 Výsledky testování

Po provedení několika desítek nastavování parametrů pro pravidla doporučování (například váh) a otestování systému s takovým nastavením, vyšla prozatím nejvyšší průměrná úspěšnost systému na 38.17616695594002 %. Parametry, které byly při dosažení tohoto výsledku jsou nastaveny v konfiguračním souboru (*config.yml*). Určitě by se dala naleznout ještě mnohem lepší kombinace parametrů, ale jelikož výpočet všech podobností trvá několik desítek minut, není zcela možné zkoušet všechny možnosti nastavení. Dosažení takového výsledku je lepší, než bylo očekáváno, protože jak již bylo řečeno v kapitole 4.1, původní systém nebyl téměř nijak automatizován a výsledky doporučení byly absolutně nepersonalizované.

Proto, aby se RecTv velmi dobře otestoval, by bylo potřeba získávat zpětnou vazbu od uživatelů. Na základě této zpětné vazby by se dala doporučení zlepšit například tak, že budou uživatelem špatně hodnocená doporučení penalizována a v příštím doporučení by se tak podobné výkyvy nedostávaly na povrch. Také bych nejraději zavedl získávání hodnocení všech pořadů např. formou líbí/nelíbí. Takto získaná data by se dala využít pro další způsob doporučení, případně tak utvrdit způsob doporučení stávající. Mohla by se tak v implementovaném případě mapy relevancí zvyšovat relevance podobných

pořadů i na základě pořadů, které uživatel nikdy v blízké historii neviděl.

Pokud by RecTV měl zůstat navždy doporučovacím systémem založeným na content-based filtrování, bylo by možné využít jinou metodu pro výpočet podobností jako např. euklidovskou či manhattanskou vzdálenost a možná tak získat lepší výsledky. Také by bylo možné přidat některé další pravidla, která by zpřísnila výpočet podobnosti, pro tato pravidla by však musela být nejprve získána data o titulech.

7 Možnosti aplikace metod A.I.

Vytvoření této práce je také testováním Proof of Concept, kdy bylo jako produkt vytvořena implementace bez použití metod umělé inteligence. Bez provedení dlouhodobějších provozních testů nelze prozatím rozhodnout, zda-li bude řešení dostačující. Rozhodovat můžeme také na základě zátěže systému na serveru, kdy současné řešení pracuje pouze několik desítek minut v případě výpočtu podobností všech kombinací titulů (64jádrový procesor).

Pro přechod na strojové učení by pravděpodobně bylo potřeba vypořádat se složitější implementací a také s řešením na míru, aby systém doporučoval lépe. Jelikož se vývoj recommendation enginu nejspíše nikdy nezastaví, bude potřeba získávat stále lepší výsledky doporučení a také tedy potřeba vylepšovat metody pro implementaci. K tomu by šlo například použít TensorFlow, open source softwaru pro machine learning [43], ve spojení s knihovnou Keras v jazyce Python [44]. Implementace metod machine learningu by oproti současnému řešení muselo s velkou pravděpodobností provádět některé z výpočtů přímo na GPU.

ZÁVĚR

Hlavním cílem této diplomové práce bylo vytvořit doporučovací systém, který bude doporučovat pořady uživatelům v rámci IPTV služby. Z počátku bylo nutné se obecně seznámit s doporučovacími systémy, jakým způsobem pracují a jak se testují. Následně jsem se seznámil s doporučováním TV obsahu ve společnosti, pro kterou pracuji. Také jsem zanalyzoval požadavky této společnosti a vytvořil pravidla, podle kterých se vypočítává podobnost TV obsahu. Na základě těchto požadavků jsem vytvořil systém pro doporučování TV pořadů založený na filtrování podle podobnosti obsahu. Systém, který jsem nazval RecTV, splňuje veškeré požadavky společnosti a zároveň také požadavky zadání.

V první řadě jsem se seznámil s existujícími doporučovacími systémy a technikami, které se při tvorbě takových systémů používají. Po celkové analýze a zpracování požadavků společnosti jsem pro vytvoření RecTV vybral programovací jazyk Go, který vyniká hlavně ve své rychlosti a jednoduché, avšak velmi striktní syntaxi. Na základě požadavků jsem navrhl architekturu systému a rozdělil jsem jej na tři hlavní části – výpočet podobností, získávání doporučení a HTTP server. Při prozkoumání dostupných informací o TV pořadech jsem zvolil vhodná pravidla pro výpočet podobností. Po veškerých analýzách a návrzích jsem vytvořil programové řešení projektu a na samotný závěr práce jsem provedl testování a zhodnocení celého systému.

Další vývoj na tomto systému by vyžadoval lepší zpracování dat o pořadech a získávání zpětné vazby od uživatelů IPTV služby. Systém, který je nyní založen na filtrování podle podobnosti obsahu, by mohl být po doplnění chybějících informací přepracován na přístup kolaborativního filtrování. V takovém případě by již doporučení nezávisela pouze na minulosti jednoho uživatele, který o doporučení žádá, ale na podobném vkusu všech uživatelů IPTV služby. Není však předem známo, zda-li by takový krok kvůli dynamickému TV obsahu nebyl výpočetně příliš náročný. Další možností by bylo přidání pouze některých částí kolaborativního filtrování a vytvořit tak hybridní doporučovací systém.

Při práci tohoto projektu jsem získal nové zkušenosti z odvětví doporučování obsahu a jakožto PHP vývojář jsem si poprvé vyzkoušel programování v jazyce Go. Tento programovací jazyk se mi zalíbil a v budoucnu bych se v něm chtěl dále zlepšovat a získávat tak více zkušeností. Opravdu velkou zkušeností pro mne bylo vytvoření celého systému pro doporučování a budu velice rád, pokud na tomto projektu budu moci dále pracovat a vylepšovat jej.

SEZNAM POUŽITÉ LITERATURY

[1] FATI, Suliman Mohamed, Saiful AZAD a Al-Sakib Khan PATHAN. *IPTV delivery networks: next generation architectures for live and video-on-demand services.* Hoboken, NJ, USA: Wiley, 2018. ISBN 978-1-119-39790-8.

[2] What is VOIP? - VoIP-Info. *Welcome to VOIP-info: a reference guide to all things VOIP. - VoIP-Info* [online]. Copyright © 2003 [cit. 2020-08-07]. Dostupné z: https://www.voip-info.org/what-is-voip/.

[3] What is Internet Protocol Television (IPTV)? - Definition from Techopedia. *Techopedia: Educating IT Professionals To Make Smarter Decisions* [online]. Copyright © 2020 Techopedia Inc. [cit. 2020-04-26]. Dostupné z: https://www.techopedia.com/definition/24957/internet-protocol-television-iptv.

[4] RICCI, Francesco, Lior ROKACH, Bracha SHAPIRA a Paul B. KANTOR. *Recommender Systems Handbook* [online]. Boston, MA, USA: Springer, 2011 [cit. 2020-08-03]. ISBN 978-0-387-85820-3. Dostupné z: https://rd.springer.com/book/10.1007/978-0-387-85820-3.

[5] catch-up TV | meaning in the Cambridge English Dictionary. *Cambridge Dictionary | English Dictionary, Translations & Thesaurus* [online]. Copyright © Cambridge University Press [cit. 2020-04-27]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/catch-up-tv.

[6] Sledování TV přes internet aneb IPTV v Česku – aktualizováno! - Internet pro všechny. *Internet pro všechny* [online]. Copyright © 2002 [cit. 2020-04-28]. Dostupné z: http://www.internetprovsechny.cz/sledovani-tv-pres-internet-aneb-iptv-v-cesku/.

[7] IGMP: what is the Internet Group Management Protocol? - IONOS. *1&1 IONOS Inc.* [online]. Copyright © 2020 [cit. 2020-08-07]. Dostupné z: https://www.ionos.com/digitalguide/server/know-how/igmp-internet-group-management-protocol/.

[8] Teletext ČT — Teletext — Česká televize. *Česká televize* [online]. Copyright © 1996 [cit. 2020-07-23]. Dostupné z: https://www.ceskatelevize.cz/teletext/ct/?channel=CT2&page=300.

[9] O2 TV. *O2 TV* [online]. [cit. 2020-08-05]. Dostupné z: https://www.o2tv.cz/tv/program/.

[10] Introduction to recommender systems - Baptiste Rocca. *Towards Data Science* [online] [cit. 2020-05-06]. Dostupné z: `https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada`.

[11] Recommender Systems in Practice - Houtao Deng. *Towards Data Science* [online] [cit. 2020-05-07]. Dostupné z: `https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a`.

[12] Build a Recommendation Engine With Collaborative Filtering – Real Python. *Python Tutorials – Real Python* [online]. Copyright © 2012 [cit. 2020-08-06]. Dostupné z: `https://realpython.com/build-recommendation-engine-collaborative-filtering/`.

[13] Deep Neural Network Models | Recommendation Systems. *Google Developers* [online] [cit. 2020-08-04]. Dostupné z: `https://developers.google.com/machine-learning/recommendation/dnn/softmax`.

[14] ADOMAVICIUS, G. a A. TUZHILIN. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* [online]. 2005, 17(6), 734-749 [cit. 2020-08-07]. DOI: 10.1109/TKDE.2005.99. ISSN 1041-4347. Dostupné z: `http://ieeexplore.ieee.org/document/1423975/`.

[15] Introduction to TWO approaches of Content-based Recommendation System – Kevin Luk. *Towards Data Science* [online] [cit. 2020-08-06]. Dostupné z: `https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c`.

[16] Gensim - Creating TF-IDF Matrix - Tutorialspoint. *Apache HTTP Server Test Page powered by CentOS* [online]. Copyright © Copyright 2020. All Rights Reserved. [cit. 2020-08-07]. Dostupné z: `https://www.tutorialspoint.com/gensim/gensim_creating_tf_idf_matrix.htm`.

[17] SINGH, Pramod. *Machine Learning with PySpark* [online]. Berkeley, CA, USA: Apress, 2019 [cit. 2020-08-08]. ISBN 978-1-4842-4130-1. Dostupné z: `https://rd.springer.com/book/10.1007/978-1-4842-4131-8`.

[18] Golang vs. Python: Which One to Choose? - DZone Open Source. *DZone* [online] [cit. 2020-08-07]. Dostupné z: `https://dzone.com/articles/golang-vs-python-which-one-to-choose`.

[19] Concurrency and Parallelism In Golang | by Mayur Wadekar | Medium. *Medium – Get smarter about what matters to you.* [online]

[cit. 2020-08-04]. Dostupné z: https://medium.com/@mayurwadekar2/concurrency-and-parallelism-in-golang-c8327701fd94.

[20] Frequently Asked Questions (FAQ) - The Go Programming Language. *The Go Programming Language* [online] [cit. 2020-08-01]. Dostupné z: https://golang.org/doc/faq.

[21] Go's New Brand - The Go Blog. *The Go Programming Language Blog* [online] [cit. 2020-08-01]. Dostupné z: https://blog.golang.org/go-brand.

[22] Go by Example: Goroutines. *Go by Example* [online] [cit. 2020-08-01]. Dostupné z: https://gobyexample.com/goroutines.

[23] Go by Examle: Mutexes. *Go by Example* [online] [cit. 2020-08-01]. Dostupné z: https://gobyexample.com/mutexes.

[24] Go by Example: Channels. *Go by Example* [online] [cit. 2020-08-01]. Dostupné z: https://gobyexample.com/channels.

[25] Go by Example: WaitGroups. *Go by Example* [online] [cit. 2020-08-01]. Dostupné z: https://gobyexample.com/waitgroups.

[26] Go by Example: Worker Pools. *Go by Example* [online] [cit. 2020-08-01]. Dostupné z: https://gobyexample.com/worker-pools.

[27] gob - The Go Programming Language. *The Go Programming Language* [online] [cit. 2020-08-01]. Dostupné z: https://golang.org/pkg/encoding/gob/.

[28] http - The Go Programming Language. *The Go Programming Language* [online] [cit. 2020-08-01]. Dostupné z: https://golang.org/pkg/net/http/.

[29] About MariaDB Server - MariaDB.org. *MariaDB Foundation - MariaDB.org* [online] [cit. 2020-08-01]. Dostupné z: https://mariadb.org/about/.

[30] MySQL :: MySQL 8.0 Reference Manual :: 1.2.3 History of MySQL. *MySQL :: Developer Zone* [online]. Copyright © 2020, Oracle Corporation and [cit. 2020-08-01]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/history.html.

[31] Why is the Software Called MariaDB? - MariaDB Knowledge Base. *MariaDB Enterprise Open Source Database & SkySQL MariaDB Cloud | MariaDB* [online]. Copyright © 2020 MariaDB. All rights reserved. [cit. 2020-08-01]. Dostupné z: https://mariadb.com/kb/en/why-is-the-software-called-mariadb/.

[32] Co je Docker? | Microsoft Docs. *Microsoft Docs* [online]. Copyright © Microsoft 2020 [cit. 2020-08-01]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/architecture/containerized-lifecycle/what-is-docker.

[33] Docker Hub. *Docker Hub* [online] [cit. 2020-08-01]. Dostupné z: https://hub.docker.com/.

[34] docker-compose up | Docker Documentation. *Docker Documentation* [online]. Copyright © 2013 [cit. 2020-08-01]. Dostupné z: https://docs.docker.com/compose/reference/up/.

[35] docker-compose stop | Docker Documentation. *Docker Documentation* [online]. Copyright © 2013 [cit. 2020-08-01]. Dostupné z: https://docs.docker.com/compose/reference/stop/.

[36] What is a Container? | App Containerization | Docker. *Empowering App Development for Developers | Docker* [online]. Copyright © 2020 Docker Inc. All rights reserved [cit. 2020-08-01]. Dostupné z: https://www.docker.com/resources/what-container.

[37] Vector Norm – from Wolfram MathWorld. *Wolfram MathWorld: The Web's Most Extensive Mathematics Resource* [online]. Copyright © 1999 [cit. 2020-08-02]. Dostupné z: https://mathworld.wolfram.com/VectorNorm.html.

[38] Heap Data Structure - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online] [cit. 2020-08-02]. Dostupné z: https://www.geeksforgeeks.org/heap-data-structure/.

[39] HTTP/1.1: Status Code Definitions. *World Wide Web Consortium (W3C)* [online] [cit. 2020-08-02]. Dostupné z: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

[40] Security | GORM - The fantastic ORM library for Golang, aims to be developer friendly.. *GORM - The fantastic ORM library for Golang, aims to be developer friendly.* [online]. Copyright © 2013 [cit. 2020-08-08]. Dostupné z: http://v2.gorm.io/docs/security.html.

[41] testing - The Go Programming Language. *The Go Programming Language* [online] [cit. 2020-08-04]. Dostupné z: https://golang.org/pkg/testing/.

[42] Cross-Validation | Kaggle. *Kaggle: Your Machine Learning and Data Science Community* [online] [cit. 2020-08-03]. Dostupné z: https://www.kaggle.com/dansbecker/cross-validation.

[43] TensorFlow. *TensorFlow* [online] [cit. 2020-08-08]. Dostupné z: `https://www.tensorflow.org/`.

[44] Keras: the Python deep learning API. *Keras: the Python deep learning API* [online] [cit. 2020-08-08]. Dostupné z: `https://keras.io/`.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IPTV	Internet Protocol TV
VOIP	Voice over Internet Protocol
PVR	Personal Video Recordings
VOD	Video On Demand
DVR	Digital Video Recorder
RTSP	Real Time Streaming Protocol
IGMPv2	Internet Group Management Protocol verze 2
EPG	Electronic Program Guide
TF-IDF	Term Frequency - Inverse Document Frequency
API	Application Programming Interface
ORM	Object-relational mapping (objektově relační zobrazení)
GPU	Graphic Processing Unit

SEZNAM OBRÁZKŮ

Obr. 1.1 Architektura IPTV - přeloženo a upraveno z [4] 11
Obr. 1.2 Porovnání teletextového zobrazení ČT a TV Programu IPTV služby O2 TV [8, 9] 13
Obr. 2.1 Doporučení položek přístupem user-user – přeloženo z [10] 16
Obr. 2.2 Doporučení položek přístupem item-item – přeloženo z [10] 17
Obr. 2.3 Faktorizace matice – přeloženo z [10] 18
Obr. 2.4 Logika doporučení u content-based metod 19
Obr. 2.5 Způsob doporučování se zpětnou vazbou 21
Obr. 2.6 Kombinace content-based metody a kolaborativního filtrování – přeloženo z [17] 21
Obr. 2.7 Použití výsledků jedné metody jako vstupy další metody – přeloženo z [17] 22
Obr. 3.1 Logo Go [21] 25
Obr. 3.2 Architektura Docker kontejnerů – přeloženo z [36] 27
Obr. 4.1 ER diagram věnované části databáze IPTV systému 31
Obr. 4.2 Architektura recommendation enginu 32
Obr. 4.3 Dostupnost obsahu v rámci IPTV vysílání – zobrazení v TV programu 32
Obr. 5.1 Tabulky databáze RecTV 34
Obr. 5.2 Transformace dat s použitím algoritmu TF-IDF pro výpočet vah . . 35
Obr. 5.3 Vývojový diagram výpočtu podobnosti 37
Obr. 5.4 Vývojový diagram výpočtu TF-IDF pro krajiny 38
Obr. 5.5 Příklad haldy Max-heap – upraveno z [38] 39
Obr. 5.6 Ukázka průběhu získání doporučení 41
Obr. 5.7 Ukázka průběhu přepočítávání podobností titulů 42
Obr. 6.1 Výběr testovacích dat u křížové validace – upraveno z [42] 45

SEZNAM PŘÍLOH

P I. Obsah přiloženého paměťového média

PŘÍLOHA P I. OBSAH PŘILOŽENÉHO PAMĚŤOVÉHO MÉDIA

CD obsahuje následující adresářovou strukturu:

- database/ – obsahuje soubory k databázi
- src/ – obsahuje zdrojové kódy aplikace
  - client/ – Klient pro volání na server
  - config/ – Zpracování konfiguračního souboru
  - data/mysql/ – Zpracování přístupu k databázi
  - fileCache/ – Zpracování cachování v souborovém systému
  - loader/ – Zpracování spouštění výpočtu podobností
  - matrix/ – Zpracování výpočtu podobností
  - model/ – Zpracování pravidel pro výpočet podobností
  - recommender/ – Zpracování doporučení
  - server/ – HTTP server
  - similarity/ – Výpočet samotné podobnosti
  - similarityHandlers/ – Sada pravidel implementující rozhraní pro výpočet podobností
- config.yml – Konfigurační soubor
- go.* – Soubory pro balíčkování Go
- Makefile – Spouštění programu pomocí make
- recommendation-cross-checker – skript pro testování
- Readme.md