Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Doporučovací systémy a jejich aplikace

2015

Bc. Lukáš Novák

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Lukáš Novák
Název práce:	Doporučovací systémy a jejich aplikace
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	60
Přílohy:	1 CD
Jazyk práce:	český

Bibliographic info

Author:	Bc. Lukáš Novák
Title:	Recommendation systems and their application
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	60
Supplements:	1 CD
Thesis language:	Czech

Anotace

Doporučováním vhodných položek může být v mnoha oblastech zajištěna lepší interakce uživatele s informačním systémem. Tím nejen stoupá účinnost informačních systémů, ale v oblasti komerčních systémů i zisk. Práce se zabývá popisem známých metod pro doporučování vhodných položek. Tyto metody byly použity pro sestavení doporučovacího systému ve specifické oblasti objednávání jídel. Následně je pro tuto oblast implementován doporučovací systém založený na formální konceptuální analýze.

Synopsis

The interaction between users and certain information systems increases with recommendation of appropriate items. There is both better efficiency and higher profit, especially in e-commerce. This thesis provides the description of well known approaches to recommending items. These approaches were used in order to implement a recommender systems operating in domain of meal ordering. Thesis also shows approach of meals recommending based on a formal concept analysis.

Klíčová slova: doporučovací systémy; doporučování jídel; formální konceptuální analýza

Keywords: recommendation systems; recommending meals; collaborative; content-based; formal concept analysis

Děkuji Mgr. Petru Krajčovi, Ph.D., za vedení této diplomové práce a Mgr. Leoši Juráskovi, MBA, za poskytnutí testovacích dat o objednávkách v univerzitní menze.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Vývoj doporučovacích systémů je v zájmu vědců v oblasti zpracování velkého množství dat zejména od 90. let 20. století. V té době díky rozšíření internetu mezi miliony lidí začali mít zástupci internetových obchodů k dispozici velké množství dat o chování zákazníků. Tato data však byla zřídka využitá a možnosti, jak je využít ve prospěch obchodníků se dostávaly do popředí.

Doporučovací systémy se nejen v oblasti internetu využívaly stále častěji. Mezinárodní korporace Google, Inc. nebo Amazon.com, Inc. začaly vyvíjet služby, jejichž úspěšnost závisí na kvalitě doporučování vhodných položek. V případě společnosti Amazon vlastnící internetový obchod Amazon.com jde o doporučení zboží, které si uživatel může koupit. Díky velikosti a síle doporučovacího systému může Amazon.com často velmi přesně zobrazit zboží, které uživatele opravdu zajímá.

Metod, jak toho docílit je popsána celá řada. Liší se způsoby samotného výběru vhodné položky, ale i oblasti použití. Jakou metodu doporučovacího systému použít závisí právě na oblasti použití, ale také na typu vstupních dat. Pro doporučení vhodných hudebních nosičů v internetovém obchodě zřejmě budou vhodné jiné metody než pro doporučení novinového článku na zpravodajském serveru. V knihách a vědeckých článcích jsou popsány metody, které pro doporučení využívají znalosti o uživatelích s podobnými preferencemi, ale také ty, které uživateli doporučí položku pouze se znalostí jeho historie.

V práci se budeme zabývat vytvořením vhodného doporučovacího systému pro doporučování jídel, konkrétně jídel z univerzitní menzy Univerzity Palackého. Ta pro tyto účely poskytla anonymizovaná data o uživatelích a jejich objednávkách. Jak si později ukážeme, tato data jsou velmi specifická. V práci je popsáno sestavení doporučovacích systémů využívající běžné metody. Mimo tyto metody je sestaven doporučovací systém s využitím formální konceptuální analýzy, metody analýzy dat, jenž umožňuje získat jiný pohled na data.

1 Doporučovací systémy

V této kapitole si popíšeme základní definici a vlastnosti doporučovacích systémů. Následně si vysvětlíme základní principy několika metod, které se v této oblasti ustálily.

Jsou to například doporučovací systémy, které doporučují dle vlastností položek. Tyto systémy většinou uchovávají profil uživatele (například oblíbené filmové žánry) a doporučují položky, které se s profilem shodují. Této metody doporučení využívá rodina doporučovacích systémů nazvané Content-Based. Tyto budeme označovat jako *doporučovací systémy se znalostí obsahu*.

Další metodou je doporučovat na základě znalosti uživatelů. Tyto systémy využívají nalezení uživatelů s podobnými preferencemi a doporučují položky na základě chování „davu“. Tyto systémy se obecně nazývají Collaborative filtering[1], dále v tomto textu budou označovány jako *doporučovací systémy se znalostí uživatelů*.

Dále je možnost provádět hybridizaci výše uvedených metod. *Hybridní doporučovací systémy* kombinuje dvě nebo více technik a využívá tak vlastností více přístupů.

1.1 Definice a vlastnosti

Jednoduše řečeno, doporučovací systém je algoritmus, který na základě informací o uživateli na vstupu vrátí doporučenou položku. Obecně doporučovací systém vrací uspořádaný seznam položek dle ohodnocení. Protože výsledek závisí na uživateli, jsou někdy doporučovací systémy popisované v tomto textu označovány jako personalizované. V některých oblastech se využívá doporučovací systém, který doporučuje položky nezávisle na uživateli, například jde o seznamy nejčastěji nakupovaných položek v internetových obchodech. Takové systémy však nejsou vzhledem k implementaci a obecnému použití příliš zajímavé, proto se jimi v tomto textu nebudeme zabývat.

Doporučovací systém navenek působí jako *černá skříňka*, která pro určitého uživatele vrátí „nějaký“ výstup. Takový systém může být zahrnut v internetovém obchodu, mobilní aplikaci nebo být dostupný jako veřejné API. Můžeme si ho představit pomocí obrázku 1, kde U je množina všech uživatelů a I_u je množina všech doporučení pro uživatele u na základě komponenty *data*. Ta může být různá napříč oblastmi použití nebo určená metodou doporučování. Ve většině případů je to kombinace následujících dat:

- data o uživatelích (preference uživatelů)
- data o položkách (vlastnosti položek)
- vztah mezi uživateli a položkami

[1] Resnick a Varian [1] označují tento termín jako zavádějící a doporučují jen pro tyto metody používat obecnější termín *Recommender systems*.

Další základní vlastností doporučovacího systému je přítomnost modelu, nad kterým sestavujeme doporučení. Doporučovací systémy mohou být děleny do kategorií:

Nevyužívající model
Systémy obecně využívají databázi uživatelů a položek, nad kterou přímo sestavují jednotlivá doporučení. Tyto systémy jsou označovány *Memory-based*.

Využívající model
Systémy obecně využívají databázi pro výpočet *modelu*, na základě kterého následně sestavují doporučení. To je vhodné zejména v případech, kdy databáze obsahuje tolik dat, že je neefektivní provádět každé doporučení přímo nad nimi. Model doporučovacím systémům slouží především k nalezení skrytých informací v datech. Také slouží k odstranění redundantních informací. Takové systémy mohou být řádově efektivnější. Jsou označovány *Model-based*.

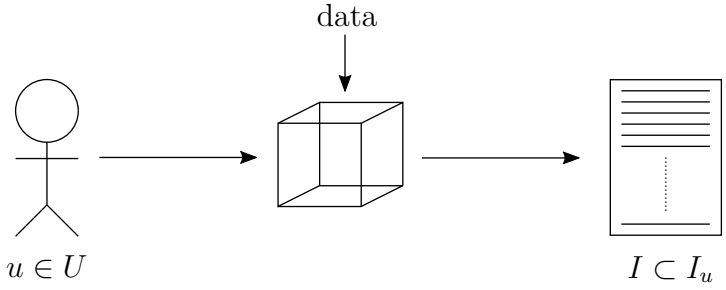
Podle dat, na základě kterých doporučovací systém sestavuje doporučení, se dělí doporučovací systémy do skupin, které si popíšeme v následujících kapitolách.

1.2 Metoda založená na znalosti obsahu

Metody založené na znalosti obsahu doporučují podle vlastností položek nebo uživatelů. V oblasti filmů mohou být doporučovány položky například podle žánru nebo režiséra. Podobně lze využít jako vlastnost uživatele například demografická data. U těchto metod rozlišujeme mezi samotným obsahem a atributovými daty:

Obsah
Obsahem je většinou myšlen například nestrukturovaný text popisu položky. Často jde o text v přirozeném jazyce (novinový článek), ze kterého je třeba získat nejdůležitější informace. Tak budeme obsah položek chápat my.

Atributová data
To jsou například již zmíněná data o žánru nebo režisérovi. Metody, které



Obrázek 1: Znázornění doporučovacího systému

pracují právě s takovým „obsahem“, se nazývají *Knowledge-based*. Někteří autoři uvádějí metody založené na znalosti atributových dat jako podmnožinu metod založených na znalosti obsahu, běžně se však tyto metody uvádějí samostatně [2].

Doporučovací systémy čistě založené na této metodě používají výhradně historii uživatele, pro kterého se doporučení sestavuje. Je tedy možné použít tuto metodu i pokud nejsou k dispozici data o ohodnocení položek ostatními uživateli, nebo dokonce pokud je v systému pouze jeden uživatel.

1.2.1 Počet výskytů klíčových slov

Na každou položku se můžeme dívat jako na vektor, jehož každá složka reprezentuje klíčové slovo v obsahu položky. Za hodnoty složek vektoru je zvolen počet výskytů odpovídajícího slova v obsahu. Vytvářením vztahů mezi uživateli a položkami, například koupí položky při použití implicitního hodnocení, je uživatelům sestavován profil. Takový profil sestává ze slov, které zřejmě uživatele nějakým způsobem zajímají.

Zbývá však určit klíčová slova. V případě internetového obchodu si taková slova může určit uživatel sám nebo je systém odhadne na základě jeho historie. Potom je možné hledat položky vhodné k doporučení pomocí shody mezi uživatelským profilem a jednotlivými položkami.

1.2.2 Podobnost položek

Protože položky jsou reprezentovány vektory v n-dimenzionálním eukleidovském prostoru, můžeme mezi těmito vektory vypočítat úhel. Ten nám udává podobnost mezi položkami. Pro úhel θ mezi vektory $\vec{a}$ a $\vec{b}$ platí:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}.$$

Jako míru podobnosti můžeme použít právě kosinus úhlu. Ten nabývá hodnot od 0 do 1, přičemž 0 značí nejméně podobné položky, 1 nejvíce podobné položky. Doporučovat položky pak můžeme výpočtem podobností mezi vhodnými položkami k doporučení a položkami, které si uživatel zakoupil.

Vektory nemusíme použít jen pro vybraná klíčová slova, ale pro všechna slova v obsahu položky. Pokud ale využijeme ve složkách vektorů počet výskytů slov v obsahu, znamená to, že čím častěji se slova v obsahu vyskytují, tím mají větší váhu. V případě, že jde o nestrukturovaný text přirozeného jazyka nemůžeme předpokládat, že všechna slova vyskytující se v obsahu jsou stejně důležitá. Proto se často namísto počtu výskytů používá váhové schéma TF-IDF, které popisuje například Chakrabarti v [3].

TF *Term frequency*
Hodnota $\mathrm{TF}(w, c)$ nám udává četnost slova w vzhledem k ostatním slovům

v obsahu c. Nechť $n(w, c)$ je počet výskytů slova w v obsahu c a $\max_c$ je nejvyšší počet výskytů ze všech ostatních slov v obsahu c. Potom platí:

$$\mathrm{TF}(w, c) = \frac{n(w, c)}{\max_c}.$$

TF(w, c) nabývá hodnot z $\langle 0; 1\rangle \in \mathbb{R}$. Můžeme říci, že čím se slovo w v obsahu vyskytuje častěji, tím je hodnota TF(w, c) blíže 1.

IDF *Inverse document frequency*
Hodnota IDF(w) značí převrácenou četnost slova ve všech položkách. Nechť C je množina všech obsahů položek a C_w je množina obsahů, které obsahují slovo w. Potom platí:

$$\mathrm{IDF}(w) = \log \frac{|C|}{|C_w|}.$$

IDF(w) nabývá hodnot z $\mathbb{R}$. Platí, že hodnota IDF(w) je vyšší, pokud se slovo w vyskytuje v obsahu malého množství položek a naopak.

Potom můžeme pro slovo w v obsahu c použít hodnotu TF-IDF(w, c), kterou vypočítáme:

$$\text{TF-IDF}(w, c) = \mathrm{TF}(w, c) \cdot \mathrm{IDF}(w).$$

Můžeme prohlásit, že slovo w má v obsahu c položky největší hodnotu TF-IDF, pokud je pro tuto položku důležitější a zároveň se v obsahu této položky vyskytuje častěji než v jiných položkách. Potom můžeme namísto jednoduchého počtu výskytů slov v položce použít právě hodnotu TF-IDF.

1.2.3 Vlastnosti metody

Metoda je obecně méně přesná než jiné metody, protože pracuje pouze s historií uživatele, pro kterého chceme doporučit položky. Pokud má uživatel malé množství položek v historii, nemusí být možné sestavit kvalitní uživatelský profil a následně pomocí něj doporučovat. Otázka však je, zda je vhodné takové doporučení sestavovat, v takovém případě zřejmě nebude příliš přesné.

Pokud chceme pracovat s obsahem v přirozeném jazyce, mohou zde také nastat problémy s detekcí skloňování slov nebo přítomností *stop slov*[2]. V případě zpracování textu v přirozeném jazyce bychom také měli dbát na význam slov, zejména záporná slova jako: *žádný*, *nikdy* a podobně. Jako příklad špatného doporučení můžeme použít doporučování restaurací a steakovou restauraci s popisem: „Restaurace, ve které si žádný vegetarián jídlo neobjedná“. Pokud by systém neznal význam slova „žádný“, mohl by takovou restauraci doporučit uživateli, v jehož profilu se nachází slovo „vegetarián“. Takové doporučení však bezpochyby není správné.

[2]Slova, která nenesou sama o sobě žádný význam. Jde především o předložky, spojky a další slova.

1.3 Metoda založená na znalosti uživatelů

Další využívaná metoda doporučuje na rozdíl od předchozí metody na základě historie více uživatelů. Pokud se vrátíme k obrázku 1, doporučovací systém zde používá jako *data* vztah mezi jednotlivými uživateli a položkami. Ten je v internetovém obchodu určen například tím, jak položky uživatelé hodnotí. Metoda, kterou poprvé popsali Goldberg a kol. [4], je založena na principu doporučení položek, které hodnotili kladně uživatelé, se kterými má uživatel, kterému doporučení sestavujeme, podobné preference.

Chceme-li doporučit uživateli položku, najdeme pro tohoto uživatele takové uživatele, kteří s ním mají podobné preference. Kolik takových uživatelů bude závisí na implementaci, obecně však můžeme říct, že čím větší je tento počet, tím je doporučení přesnější. Pokud by doporučovací systém nalezl uživatele, který má preference shodné, pak by bylo doporučení správné[3]. Takový případ však v praxi často nenastává, proto je výsledné doporučení kombinací preferencí většího počtu uživatelů. Na druhou stranu, jak uvádí Resnick a Varian [1], je výhodnější mít hodně uživatelů s lišícími se preferencemi, kteří se navzájem neznají, nikoliv uživatele, kteří mají preference podobné. Tedy mít takovou skupinu uživatelů, aby byly zastoupené různé preference a doporučovací systém byl tak připraven na doporučení *jakémukoli* uživateli.

1.3.1 Způsob ohodnocení položek

Doporučovací systémy však nemusí vždy hledat podobné uživatele podle hodnocení položek. Základní způsoby hodnocení jsou následující:

Explicitní hodnocení
: Hodnocení, které je získáno od uživatele (zejména se jedná o hodnocení položek, které si uživatel koupil). Takové hodnocení může používat stupnici 0-1 (ne/ano), 1-5, nebo jinou. Čím větší stupnice pro hodnocení je, tím má doporučovací systém přesnější data o preferencích uživatele. Každý uživatel však může hodnotit v jiném rozmezí, proto je nutné použít metodu, která bere ohled na tuto odlišnost (kap. 1.3.2).

Implicitní hodnocení
: Hodnocení je automaticky přiřazeno položkám na základě chování uživatele. Může to být například zakoupením položky nebo v případě internetového obchodu pouhým prohlédnutím informační stránce o položce. Systém využívající takové hodnocení má obecně o uživatelích více dat, ty však nemusí být natolik přesné jako u explicitního hodnocení.

Tabulka 1: Příklad: explicitní hodnocení položek

	p_1	p_2	p_3	p_4	p_5
u_1	5	4	3	5	4
u_2	1	2	5	4	1
u_3	4	2	1	3	**?**

1.3.2 Pearsonův korelační koeficient

Předpokládejme množinu uživatelů U a uživatele $\{u_1, u_2, u_3\} \subset 2^U$, množinu položek P, položky $\{p_1, p_2, p_3, p_4, p_5\} \subset 2^P$ a hodnocení položek těmito uživateli v tabulce 1. K výpočtu odhadu ohodnocení položky p_5 uživatelem u_3 použijeme Pearsonův korelační koeficient, který upravili pro použití s ohodnocením Resnick a kol. [5]:

$$\mathrm{sim}(a,b) = \frac{\sum_{p \in P} \left(r_{a,p} - \overline{r_a}\right)\left(r_{b,p} - \overline{r_b}\right)}{\sqrt{\sum_{p \in P} \left(r_{a,p} - \overline{r_a}\right)^2} \cdot \sqrt{\sum_{p \in P} \left(r_{b,p} - \overline{r_b}\right)^2}},$$

kde $\overline{r_a}$ je průměr hodnocení uživatele a, $r_{a,p}$ je hodnocení položky p uživatelem a a $\mathrm{sim}(a,b)$ je korelace mezi uživateli a a b. Pearsonův korelační koeficient je hodnota v intervalu $\langle -1, 1 \rangle$, která při našem použití popisuje míru shody v hodnocení. Čím je korelace bližší hodnotě 1, tím je shoda větší, naopak čím bližší je k -1, tím je shoda menší.

Nejprve spočítáme průměry hodnocení pro všechny uživatele:

$$\overline{r_{u_1}} = 4,2 \qquad \overline{r_{u_2}} = 2,6 \qquad \overline{r_{u_3}} = 2,5$$

a následně vypočítáme korelaci uživatele u_3 s uživatelem u_1 a u_2:

$$\mathrm{sim}(u_1, u_3) = 0,94 \qquad \mathrm{sim}(u_2, u_3) = -0,47.$$

Vypočítali jsme, že uživatel u_3 má větší shodu s uživatelem u_1 než s uživatelem u_2, a to i přesto, že tito dva nehodnotili na stupnici stejně. Všimněme si však vztahu mezi jejich hodnocením. Uživatel u_3 hodnotil všechny položky o jeden až dva body na stupnici méně než uživatel u_1. Korelace vypočtená pomocí Pearsonova korelačního koeficientu nám tedy řeší situaci, kdy každý uživatel používá hodnocení na stupnici jiným způsobem.

1.3.3 Vlastnosti metody

Doporučovací systémy založené čistě na této metodě mají malou úspěšnost, pokud uživatel neohodnotil žádnou nebo jen malé množství položek. V takovém případě nemohou být nalezeni blízcí uživatelé. Tato vlastnost se nazývá *cold-start* a poprvé ji v oblasti doporučovacích systémů definovali Schein a kol. [6].

[3] Většinou však nemůžeme říci, že je doporučení správné. Samotný výběr položek uživatelem ovlivňuje mnoho faktorů a my se většinou jen můžeme domnívat, že systém doporučil správně.

Nevýhoda těchto systémů se může projevit i v případě, že hodnocení některých položek má větší váhu než hodnocení ostatních položek. Jsou to zejména ty, mezi jejichž hodnocením není shoda (můžeme říci, že jsou kontroverzní). Herlocker a kol. [7] navrhli řešení ve formě úpravy Pearsonova korelačního koeficientu, která zvyšuje vliv položkám s velkou odchylkou v hodnocení mezi uživateli a naopak snižuje vliv položkám, kde je odchylka v hodnocení malá.

1.4 Hybridní doporučování

Obě popsané metody mají své nevýhody. U metody založené na znalosti obsahu to může být nízká úspěšnost kvůli špatnému zpracování významu položek. Metoda založená na znalosti uživatelů se zase vyznačuje nemožností doporučovat, pokud je v systému malé množství uživatelů. Nabízí se tyto metody kombinovat s cílem snížit nevýhody samostatných metod.

Vraťme se k příkladu v kapitole 1.2.3, tedy doporučování restaurace uživateli, který je, dle uživatelského profilu, vegetariánem. V takovém případě využijeme znalosti uživatelů a nalezneme podle uživatelských profilů všechny uživatele, kteří jsou vegetariáni. V ideálním případě žádný z nich zmíněnou restauraci neohodnotí kladně. Toho můžeme využít, doporučení zkombinovat a ve výsledku takovou restauraci nedoporučit.

2 Doporučování jídel

Sekce se zabývá daty, která byla pro tuto diplomovou práci použita. Popíšeme si zde vlastnosti dat a rozdíly mezi těmito daty a daty, která se běžně používají v literatuře týkající se oblasti doporučovacích systémů.

2.1 Získání dat

Pro účely sestavení doporučovacího systému v oblasti doporučování jídel bylo potřeba získat data o jídlech a objednávkách v univerzitní menze. Tato data poskytlo vedení Správy kolejí a menz Univerzity Palackého, se kterým autor textu spolupracoval již při tvorbě bakalářské práce „*Aplikace pro objednávání jídel pro platformu Android*“.

Získaná data jsou ve formátu, který znázorňuje tabulka 2. Tato data však bylo nutné transformovat do podoby vhodné k sestavení databáze pro doporučování. To znamenalo mimo jiné detekovat stejná a podobná jídla, čemuž se budeme věnovat v následující kapitole.

2.2 Specifika doporučování jídel

Často se pro výuku doporučovacích systémů používá oblast internetového obchodu. V této oblasti jde o doporučení položky (knihy, filmu, hudebního nosiče či jiného zboží). Ať je použito hodnocení explicitní (manuální přiřazení hodnocení uživatelem) či implicitní (například počet navštívení stránky), téměř vždy je přítomna podmínka, která značí, že nesmí být doporučena položka, kterou uživatel již vlastní. V případě internetového obchodu má taková podmínka smysl, běžně nemá uživatel zájem koupit si knihu, kterou již vlastní.

Při doporučování jídel je třeba použít jiný přístup. I přesto, že mohou existovat strávníci, kteří chtějí vyzkoušet co nejvíce jídel, zpravidla si bude strávník chtít objednat jídlo, které si již v minulosti objednal nebo jídlo, které je jim nějakým

Tabulka 2: Ukázka zdrojových dat z univerzitní menzy

Datum	Strávník	Popis jídla
2014-08-12	133	400g Boloňské špagety s vepř. masem, nápoj
2014-08-12	1196	Přírodní kuřecí plátek, brambory, nápoj
2014-08-13	2916	VEG 415g Ovocné knedlíky sypané mákem, nápoj
2014-08-13	1123	140g Sekaná pečeně, bramborová kaše, nápoj
2014-08-13	245	120g Vepřový vrabec, dušený špenát, houskový knedlík, nápoj
2014-08-14	112	100g Hovězí maso vařené, rajská omáčka, houskový knedlík, nápoj

způsobem podobné. Může tak činit na základě předchozí zkušenosti, nebo proto, že jídlo patří do skupiny těch, která mu chutnají.

Ve známých doporučovacích systémech jde zpravidla o doporučení z tisíců položek, ze kterých doporučovací systém sestaví uspořádaný seznam. Při doporučování jídel, alespoň při použití v univerzitní menze, však nelze doporučit jakékoliv jídlo z databáze. Jídla, která je možné doporučit, totiž určuje denní nabídka, která obsahuje zpravidla pět jídel. Zde se nabízí otázka, zda uživateli vždy doporučit jídlo z nabídky nebo nedoporučit, pokud zjistíme malou pravděpodobnost úspěchu.

Vzhledem k tomu, že v databázi nejsou informace o hodnocení jednotlivých jídel uživateli, nelze použít explicitní hodnocení. Máme však informaci o počtu objednávek, které můžeme využít jako implicitní hodnocení jídel. Čím více provedl uživatel objednávek stejného jídla, tím je pravděpodobnější, že bude mít jídlo raději. Této skutečnosti využijeme k sestavení doporučovacího systému se znalostí obsahu (kap. 3.5) i doporučovacího systému se znalostí uživatelů (kap. 3.6).

Lze však doporučovat i jiným způsobem než použitím pouze implicitního hodnocení. Jedinou informaci, kterou máme o vlastnostech jídla, je jeho popis. Jednotlivá slova v popisu nám dávají informaci o složení jídla či jednotlivých ingrediencích. Jídla se však mohou lišit například pouze typem přílohy. Můžeme však považovat jídlo s jinou přílohou za jiné jídlo? Kroky vedoucí k jednoznačnému určení jídla pomocí detekce stejných a podobných jídel jsou popsány v následujících kapitolách. Veškerá rozhodnutí v těchto kapitolách byla ověřena sestavením konfigurací a následným vyhodnocením. Některé rozdíly v úspěšnosti doporučování mezi konfiguracemi jsou popsány v kapitole 4.

2.3 Detekce stejných jídel

- **Vstup**: Textový řetězec, například `100g Hovězí maso vařené, rajská omáčka, houskový knedlík, A 1 3 5`
- **Výstup**: Řetězec klíčových slov, pro daný vstup: `hovězí, maso, vařené, rajská, omáčka`

Jednotlivé objednávky jsou určeny informací o datu provedení, identifikátorem uživatele a popisem jídla. Sestavení doporučovacího systému mírně komplikuje skutečnost, že jídla nejsou určena jednoznačným identifikátorem. Analýzou bylo zjištěno, že se v datech vyskytují na základě popisu velmi podobná jídla, například jídla v tabulce 3.

Je zřejmé, že s těmito jídly by měl doporučovací systém pracovat jako se stejnými. Pokud bychom přiřadili těmto jídlům různé identifikátory, doporučovací systém by nám například na základě předchozí objednávky jídla s identifikátorem 1 nedoporučil jídlo s identifikátorem 2. Proto použijeme několik transformací popisu tak, aby doporučovací systém považoval jídla v tomto a dalších podobných případech za stejná.

Tabulka 3: Ukázka stejných jídel ve zdrojových datech

ID	Popis
1	VEG 350g Těstoviny penne s nivovým přelivem, nápoj
2	VEG 300g Těstoviny penne s nivovým přelivem, nápoj
3	VEG 350g Těstoviny Penne s nivovým přelivem, nápoj
4	VEG 350g Těstoviny penne s nivovým přelivem, nápoj A 1, 7
5	STUDENTSKÝ TIP: VEG 350g Těstoviny penne s nivovým přelivem, nápoj

2.3.1 Základní transformace

Cílem transformací je získat stejný řetězec klíčových slov pro všechna jídla, která má doporučovací systém považovat za stejná. Pokud se vrátíme k tabulce 3, jídla v ní obsažená by měla být transformována do stejného řetězce. Tento řetězec bude pro další kroky sloužit jako jednoznačné určení jídla. Všem jídlům se stejným řetězcem následně přiřadíme jednoznačný identifikátor.

Mezi základní transformace řadíme především odstranění hmotnosti porce, seznamu alergenů, označení `STUDENTSKÝ TIP` a záměnu velkých písmen za malá písmena. Také zde odstraňujeme veškeré slova s délkou do dvou znaků. Tím vyloučíme, že budou předložky či spojky použité jako klíčová slova.

2.3.2 Detekce skloňování

V mnoha jídlech se objevují slova různě skloňovaná, například `vepřové` a `vepřová`. K tomu, aby systém považoval tato slova za stejná, můžeme využit regulární výrazy a definovat nahrazení tak, jak je znázorněno v tabulce 4. Toto řešení má ale zřejmé nevýhody, je totiž potřeba sestavit náhrady manuálně pro každou databázi jídel. I tak se ale nemusí podařit zahrnout všechny potřebné náhrady.

Tabulka 4: Nahrazení slov různě skloňovaných pomocí regulárních výrazů

Regulární výraz	Náhrada
`vep[^ ]*`	vepř
`hov[^ ]*`	hov
`soj[^ ]*`	soj
⋮	⋮
`smaž[^ ]*`	smaž

Vhodnější pro detekci skloňování a překlepů je použít Levenshteinovu vzdále-

Tabulka 5: Aplikace Levenshteinovy vzdálenosti

a	b	$\text{lev}_{a,b}(\lvert a\rvert, \lvert b\rvert)$
vepřové	vepřový	1
nivovou	nivovým	2
sýrovým	nivovým	4
sojová	sojovou	2

nost[4]. Pro dvě slova je to počet vložení, smazání a nahrazení jednoho znaku, který je potřeba pro transformaci z jednoho slova na druhé. Definice Levenshteinovy vzdálenosti $\text{lev}_{a,b}$ mezi slovy a a b je:

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{když } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1 \\ \text{lev}_{a,b}(i,j-1) + 1 \\ \text{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{jinak,} \end{cases}$$

$$1_{(a_i \neq b_j)} = \begin{cases} 0 & \text{když } a_i = b_i \\ 1 & \text{jinak,} \end{cases}$$

kde $i = |a|$ a $j = |b|$. Příklady použití Levenshteinovy vzdálenosti jsou znázorněny v tabulce 5. Většina slov v jídlech, která můžeme považovat za stejná má Levenshteinovu vzdálenost menší nebo rovnu dvěma. Proto postupně projdeme všechna jídla a pokud v jídle nalezneme slovo, pro které s některým již zpracovaným slovem platí tato podmínka, provedeme náhradu za již zpracované slovo.

2.3.3 Odstranění přílohy

Množství jídel v databázi lze považovat za stejná, přesto však i přes předcházející kroky za stejná považována nejsou. Jde o jídla, která se liší pouze typem přílohy, případně přítomností oblohy nebo nápoje.

Pokud však nechceme manuálně odstraňovat možné přílohy, můžeme využít skutečnosti, že příloha se téměř vždy nachází v poslední či předposlední skupině slov oddělených čárkami. Důležitá slova reprezentující jídlo jsou naopak v první nebo druhé skupině. Pro potřeby sestavení doporučovacího systému použijeme řešení, které je nastíněné ve zdrojovém kódu 1. To bylo po množství vyhodnocení nejspolehlivější.

[4]Levenshteinova vzdálenost je pojmenována po Vladimiru Levenshteinovi, který ji definoval v roce 1965 v článku [8].

```
1 REMOVE_SIDE_DISH(MEAL)
2   KEYWORDS ← ∅
3   GROUPS ← SPLIT(DESC(MEAL), ',')
4   if (SIZE(GROUPS) <= 3)
5     ADD(KEYWORDS, GROUPS[0])
6   else
7     ADD(KEYWORDS, GROUPS[1])
8     ADD(KEYWORDS, GROUPS[2])
```

Zdrojový kód 1: Odstranění přílohy

2.4 Detekce podobných jídel

- **Vstup**: Množina dvojic, kde první složka je identifikátor a druhá množina klíčových slov, například {⟨1,{krupicová, kaše, sypaná, čokoládou}⟩, ⟨2,{krupicová, kaše, sypaná, kakaem}⟩, . . . }
- **Výstup**: Množina dvojic identifikátorů jídel, která jsou si navzájem podobná, pro daný vstup: {⟨1,2⟩, . . . }

Po eliminaci vytvoření odlišných identifikátorů pro stejná jídla nám však v databázi stále zbývají jídla, která můžeme označit jako podobná. I po provedení kroků v předcházejících kapitolách jsou například položky v tabulce 6 považovány za rozdílná jídla.

O skupinách jídel označených identifikátory 1-3 a 6-7 lze bezesporu říci, že by měla být chápána jako stejná. S touto skutečností musíme při návrhu doporučovacího systému počítat. V našem případě budeme uchovávat informaci o tom, která jídla jsou podobná jiným jídlům. Nejprve však tato podobná jídla musíme detekovat. V následujícím textu jsou popsány metody, které k této detekci můžeme využít.

2.4.1 Porovnání klíčových slov

Nejjednodušší metodou je využití porovnání množin klíčových slov a za podobná jídla považovat ta, která mají shodná klíčová slova. Tím získáme jako podobná jídla ta z tabulky 6 s identifikátory 1 a 2. I přesto, že jídlo s identifikátorem 3 se zdá být předchozím za podobné, jako podobné považováno nebude.

Další možnou metodou je zmírnění podmínky stejných klíčových slov. Pokud bude průnik klíčových slov menší nebo roven dvěma, pokud se tedy jídla liší ve dvou slovech, budeme je považovat za podobná. Tím budou za podobná považována jídla s identifikátory 4,5 a 6,7.

2.4.2 Dice koeficient

V případě jídel 4 a 5 by však zřejmě nemělo jít o podobná jídla. Metoda průniku totiž vrací hodnotu nezávisle na celkovém počtu klíčových slov. I přes stejnou

Tabulka 6: Ukázka podobných jídel ve zdrojových datech

ID	Popis Klíčová slova
1	150g Kuřecí steak s broskví a sýrem, brambory, obloha {broskví, brambory, steak, kuřecí, sýrem}
2	150g Kuřecí steak se sýrem a broskví, brambory, obloha {broskví, brambory, steak, kuřecí, sýrem}
3	100g Kuřecí plátek se sýrem a broskví, brambory, nápoj {broskví, brambory, plátek, kuřecí, sýrem}
4	VEG, BEZLEP 350g Mexické zeleninové rizoto sypané sýrem (čínská zelenina, kečup), steril. okurek, nápoj {zel, bezlep, mexické, sypané, veg, sýrem, kečup, čínská, rizoto}
5	STUDENTSKÝ TIP: BEZLEP 100g Ďábelská pochoutka z vepř. masa (pórek, kečup, chilli, solamyl), žampionová rýže, nápoj {masa, bezlep, ďábelská, pochoutka, solamyl, kečup, vepř, pórek, chilli}
6	100g Přírodní kuřecí řízek, rýže, nápoj {přírodní, kuřecí, řízek}
7	100g Přírodní kuřecí plátek, rýže, nápoj {přírodní, kuřecí, plátek}

Tabulka 7: Metody výpočtu vzdáleností mezi jídly

a	b	intersection (a, b)	dice (a, b)
4	5	2	0.22
6	7	2	0.66

hodnotu průniku klíčových slov jsou zpravidla podobnější ta jídla, která obsahují klíčových slov méně, tedy ta, která jsou kratší. Jako metodu, která počítá s celkovým počtem klíčových slov, můžeme použít výpočet Dice koeficientu[5]. Pro dvě množiny slov A a B je Dice koeficient $d(A, B)$ definován:

$$d(A, B) = \frac{2|A \cap B|}{|A| + |B|}.$$

Označme si intersection(a, b) a dice(a, b) jako míru podobnosti mezi jídly a a b. V tabulce 7 si všimněme, že pro skupiny jídel 4,5 a 6,7, pro které je počet prvků průniku stejný, je hodnota Dice koeficientu odlišná. Pokud zvolíme hranici Dice koeficientu 0,5, pak budou jídla s identifikátory 6,7 označena jako podobná a naopak jídla 5,7 nikoliv. Tato metoda, konkrétně s hranicí 0,55 byla využita pro sestavení doporučovacího systému. Část skupin podobných jídel, které byly vytvořeny za pomocí této metody jsou znázorněné v tabulce 34.

[5]Statistická metoda pro porovnávání statistických souborů, kterou definoval Lee Raymond Dice v [9].

3 Popis implementace MealRecommender

V této sekci je popsáno sestavení doporučovacího systému pro použití v oblasti jídel. Jsou zde podrobnosti o implementaci a informace k jeho použití. Pro účely porovnání různých metod a doporučovacích systémů bylo nutné sestavit celou aplikaci tak, aby bylo možné jednotlivé komponenty nahradit, provádět konfiguraci aplikace a tyto změny vyhodnocovat.

3.1 Způsob doporučení

Doporučování je prováděno pomocí instancí potomků třídy `BaseRecommender`. Tito potomci musí implementovat metodou `recommendForUser`, které je v parametru předán identifikátor uživatele, jemuž chceme doporučit jídlo. Tato metoda vrací uspořádaný seznam doporučených jídel typu `ScoreMeal`. `ScoreMeal` reprezentuje jídlo, společně s informací o ohodnocení doporučení. V případě, že doporučovací systém nemá dostatek dat pro doporučení nebo je ohodnocení doporučení jídel velmi malé, je seznam doporučení prázdný.

Třídy, které dědí z `BaseRecommender` jsou:

- `ContentBasedRecommender` – doporučovací systém se znalostí obsahu (popsán v kapitole 3.5),
- `CollaborativeRecommender` – doporučovací systém se znalostí uživatelů (popsán v kapitole 3.6),
- `HybridPriorityRecommender` a `HybridAgreementRecommender` – hybridní doporučovací systémy, které kombinují výše uvedené systémy (popsány v kapitole 3.8).

3.2 Způsob vyhodnocování

Abychom byli schopni jednotlivé doporučovací metody porovnávat, potřebujeme doporučovací systém naučit na části databáze a oproti druhé části otestovat. Toho docílíme rozdělením objednávek po dnech a tyto dny následně podle určitého poměru. Pokud zvolíme poměr 7:3, doporučovací systém naučíme na 70 % dní a otestujeme oproti 30 % dnů.

Vyhodnocování probíhá iterací přes všechny dny v testovací databázi a následně ty uživatele, kteří si v daný den objednali minimálně jedno jídlo. Doporučovacímu systému je předaná denní nabídka a získán seznam doporučených jídel. Výsledek doporučení můžeme rozdělit na:

- *Správně*, pokud doporučovací systém vrátí seznam doporučení a první jídlo v tomto seznamu odpovídá jídlu, které si uživatel daný den objednal.
- *Špatně*, pokud doporučovací systém vrátí seznam doporučení a první jídlo v tomto seznamu neodpovídá jídlu, které si uživatel daný den objednal.

Doporučení je započítáno do sloupce nezávisle na tom, zda se jídlo, které si uživatel daný den objednal, v seznamu doporučení vyskytuje či nikoliv.

- *Nedoporučeno*, pokud doporučovací systém vrátí prázdný seznam doporučení. To je zpravidla z důvodu, že systém nemá dostatek dat pro sestavení doporučení.

Poměr správných doporučení ke špatným doporučením a nedoporučením nám určuje úspěšnost doporučovacího systému.

3.3 Konfigurace

Pro konfiguraci systému slouží instance třídy `Config`, pomocí které lze nastavit jak obecné parametry pro připravení a transformaci databáze, tak i specifické parametry pro různé doporučovací systémy. Tyto parametry jsou:

- `inputPath`
  - cesta ke vstupní databázi
  - text
- `wholeDbPath`
  - cesta k zápisu transformované databáze
  - text
- `trainDbPath`
  - cesta k zápisu transformované trénovací části databáze
  - text
- `testDbPath`
  - cesta k zápisu transformované testovací částí databáze
  - text
- `splitFactor`
  - poměr rozdělení databáze na trénovací a testovací část
  - číslo v intervalu $(0; 1) \in \mathbb{R}$
- `reduceKeywordGroup`
  - určuje zda využít odstranění přílohy popsané v 2.3.3
  - pravdivostní hodnota
- `recommenderMethod`
  - typ doporučovacího systému
  - jedna z konstant
    - `RecommenderMethod.CONTENT_BASED`
    - `RecommenderMethod.COLLABORATIVE`
    - `RecommenderMethod.HYBRID_PRIORITY`
    - `RecommenderMethod.HYBRID_AGREEMENT`
- `mealSimilarityMethod`
  - typ metody pro detekci podobných jídel popsané v kapitole 2.4
  - jedna z konstant
    - `MealSimilarityMethod.OFF`
    - `MealSimilarityMethod.KEYWORD_EQUALS`

        * `MealSimilarityMethod.INTERSECTION`
        * `MealSimilarityMethod.DICE`
- `mealSimilarityValue`
  - hodnota určující práh podobnosti jídel
  - číslo, hodnoty se liší podle zvolené `mealSimilarityMethod`
- `mealNormalizeMethod`
  - typ metody pro detekci skloňování jídel popsané v kapitole 2.3.2
  - jedna z konstant
    * `MealNormalizeMethod.OFF`
    * `MealNormalizeMethod.REGEX`
    * `MealNormalizeMethod.LEVEN`

3.4 Testovací rozhraní

Pro jednodušší testování můžeme využít řádkové rozhraní `MealRecommender`, pomocí kterého lze jednoduše provádět změny v databázi a porovnávat seznamy doporučení různých konfigurací. Užitečnou funkcí rozhraní je práce s uživatelem, jenž má zpočátku prázdnou historii objednávek. Tomuto uživateli je možné následně vytvořit objednávky a sestavit seznam doporučených jídel. Funkce testovací utility jsou mimo jiné:

- objednání určitého jídla vybraným uživatelem,
- zobrazení objednávek určitého uživatele,
- přidání určitého jídla do dnešní nabídky,
- změna konfigurace doporučovacího systému,
- sestavení doporučení pro uživatele dle denní nabídky.

Testovací rozhraní je podrobněji popsáno v příloze B.

3.5 Implementace metody založené na znalosti obsahu

V této kapitole si popíšeme sestavení doporučovacího systému, jenž pro doporučení využívá pouze historii uživatele, kterému chceme sestavit doporučení. Nejprve sestavíme uživatelský profil ze všech klíčových slov jídel, které si uživatel objednal, a poté tento profil porovnáme s jednotlivými jídly.

3.5.1 Jednoduché vyhledání jídel

Uživatelský profil reprezentujeme výskyty jednotlivých klíčových slov v jeho historii. Předpokládejme množinu jídel M a jídla $\{m_{kpl}, m_{vpl}, m_{spg}\} \subset 2^M$, zobrazení $P : W \to \mathbb{Z}$ přiřazující slovu $w \in W$ ohodnocení slova v uživatelském profilu a zobrazení $W : M \to 2^W$ přiřazující jídlu $m \in W$ množinu klíčových

Tabulka 8: Příklad: Historie objednávek uživatele

m	$P(m)$	$W(m)$	Popis
m_{kpl}	3×	{kuřecí, plátek}	100g Kuřecí plátek, rýže
m_{vpl}	1×	{vepřový, plátek}	100g Vepřový plátek, rýže
m_{spg}	2×	{špagety, tuňákem}	400g Špagety s tuňákem

Tabulka 9: Příklad: Uživatelský profil

w	Slovo	$P(w)$
w_1	kuřecí	3
w_2	plátek	4
w_3	vepřový	1
w_4	špagety	2
w_5	tuňákem	2

slov $w \in W$, které jídlo m obsahuje. Označme si jednotlivá slova w_1=kuřecí, w_2=plátek, w_3=vepřový, w_4=špagety, w_5=tuňákem. Potom ohodnocení $P(w)$ slova w v uživatelském profilu je:

$$P(w) = \sum_{m \in M} M(m, w),$$

kde

$$M(m, w) = \begin{cases} 1 & \text{když } w \in W(m), \\ 0 & \text{jinak.} \end{cases}$$

Předpokládejme objednávky uživatele znázorněné v tabulce 8. Aplikací funkce $P(w)$ na všechna slova $w \in W$ získáme uživatelský profil, jenž je znázorněn v tabulce 9.

Pro ohodnocení jídel získáme definujme funkci $R : M \to \mathbb{Z}$, která přiřazuje jídlu hodnocení:

$$R(m) = \sum_{w \in W(m)} P(w),$$

pak ohodnocení jídel $m_{kpl}, m_{vpl}, m_{spg}$ je:

$$\mathbf{R(m_{kpl}) = 7}, \qquad R(m_{vpl}) = 5, \qquad R(m_{spg}) = 4.$$

Poté, co jídla seřadíme sestupně podle ohodnocení, takový seznam prohlásíme za doporučení. Pokud by tedy v dnešní nabídce bylo jídlo m_{kpl}, mohli bychom toto jídlo uživateli doporučit.

Tabulka 10: Příklad: IDF pro slova v jídlech

w	Slovo	$P(w)$
w_1	kuřecí	1,82
w_3	vepřový	2,10
w_2	plátek	2,77
w_4	špagety	4,07
w_5	tuňákem	4,76

3.5.2 Vyhledání jídel s využitím TF-IDF

Metoda popsaná v předchozí kapitole předpokládá, že jsou všechna jídla v rámci celé databáze stejně důležitá. Upravme nyní funkci tak, aby využívala váhové schéma TF-IDF, které jsme popsali v kapitole 1.2.2. Protože četnost slova v jídle pro nás není příliš zajímavá, z TF-IDF nevyužijeme obě složky, nýbrž pouze IDF.

IDF použijeme tak, že vypočítáme IDF koeficient pro každé slovo, tedy získáme pro každé slovo jeho převrácenou četnost ve všech jídlech. Pak budeme mít hodnotu IDF větší pro slova, která se vyskytují v malém počtu jídel a menší pro slova, která se vyskytují v mnoha jídlech. Funkci $\mathrm{IDF}_W : W \to \mathbb{R}^+$ si definujeme:

$$\mathrm{IDF}_W(w) = \log \frac{|M|}{\sum_{m \in M} M(m, w)}.$$

V tabulce 10 jsou vypočítány hodnoty IDF_W v reálné databázi. Všimněme si, že slova jako *veg* nebo *kuřecí* mají malou hodnotu IDF_W, protože jsou obsažena ve více jídlech. Naopak slovo *gyros* je obsaženo v méně jídlech. Se znalostí IDF_W pro všechna slova můžeme upravit funkci pro výpočet ohodnocení následovně:

$$R_{\mathrm{IDF}_W}(m) = \sum_{w \in W(m)} (P(w) \cdot \mathrm{IDF}_W(w)).$$

To způsobí, že klíčová slova, která se vyskytují ve více jídlech budou mít menší váhu než slova, která jsou ojedinělá. S použitím funkce R_{IDF_W} vypočítáme hodnocení jídel:

$$\begin{aligned}
R_{\mathrm{IDF}_W}(m_{kpl}) &= P(w_1) \cdot \mathrm{IDF_W}(w_1) + P(w_2) \cdot \mathrm{IDF_W}(w_2) = 16,54, \\
R_{\mathrm{IDF}_W}(m_{vpl}) &= P(w_3) \cdot \mathrm{IDF_W}(w_3) + P(w_2) \cdot \mathrm{IDF_W}(w_2) = 13,18, \\
R_{\mathrm{IDF}_W}(m_{spg}) &= P(w_4) \cdot \mathrm{IDF_W}(w_4) + P(w_5) \cdot \mathrm{IDF_W}(w_5) = \mathbf{17,66}.
\end{aligned}$$

Všimněme si, že na rozdíl od předchozího hodnocení, kdy jsme použili funkci $\mathrm{R_{IDF_W}}$, nyní získalo nejlepší hodnocení jídlo m_{spg}. To je dáno tím, že slova *špagety* a *tuňákem* se vyskytují v méně jídlech a mají tedy ve výpočtech větší váhu.

3.5.3 Konfigurace

Kromě základních parametrů v `Config`, jsou navíc v `ContentBasedConfig` definované speciální parametry pro nastavení doporučovacího systému na základě znalosti obsahu. Mezi tyto parametry patří:

- `distanceMethod`
  - typ metody pro hledání vhodných jídel
  - jedna z konstant
    - `DistanceMethod.COUNT`
    - `DistanceMethod.WORD_IDF`
    - `DistanceMethod.WORD_AND_MEAL_IDF` (popsané v kapitole 3.8.2)
- `minimumDifferentMeals`
  - počet různých jídel, který musí mít uživatel v historii, aby systém sestavil doporučení
  - číslo z oboru $\mathbb{Z}_0^+$
- `nearestMealsCount`
  - počet objednávek, které musí jídlo mít, aby byla klíčová slova tohoto jídla zařazena do uživatelského profilu
  - číslo z oboru $\mathbb{Z}_0^+$

3.6 Implementace metody založené na znalosti uživatelů

V této kapitole si popíšeme implementaci doporučovacího systému, který je založený na znalosti uživatelů. Jak jsme si vysvětlili v kapitole 2, máme k dispozici data o objednávkách a každou objednávku identifikuje datum, strávník a popis jídla.

V kapitole 1.3 jsme si vysvětlili, že doporučování uživateli na základě ostatních uživatelů je postaveno na principu doporučení položek od těch, kteří mají s daným uživatelem podobné preference. V případě doporučování jídel jsou to ti uživatelé, kteří si v minulosti objednávali stejná či podobná jídla. Můžeme také prohlásit, že to jsou ti uživatelé, kteří mají podobné chuťové preference.

V této kapitole budeme označovat uživatele, kterému hledáme doporučení, jako Alici a její blízké uživatele jako Bob, Cyril a David.

3.6.1 Sestavení dvojic uživatelů

Pro nalezení nejbližších uživatelů vypočítáme vzdálenosti mezi Alicí a ostatními uživateli v databázi. V následujícím textu jsou popsány metody, jak toho docílit.

Jednoduchý přístup je procházet jednotlivé uživatele a každé dvojici přiřazovat ohodnocení podle počtu objednaných jídel. Přiřazování ohodnocení je možné implementovat několika způsoby. O výpočet se stará funkce, která přiřadí ohodnocení dvěma počtům objednávek stejného jídla. Část implementací, které byly pro testování využity jsou popsány zdrojovými kódy 2, 3 a 4. Pomocí jedné z těchto funkcí je následně získáno ohodnocení dvojice sečtením hodnot pro všechna jídla, která

si objednala Alice. Hodnoty `aliceThreshold` a `bobThreshold` u metody 6 jsou počty objednávek jídla, které jsou potřeba pro nenulové ohodnocení. Tyto hodnoty se však ukázaly jako nejvhodnější nulové, ohodnocení je tedy nenulové, pokud si Alice i Bob objednali jídlo alespoň jednou.

```
1 COMPUTE_MEAL_SCORE_1(ALICE_COUNT, BOB_COUNT)
2   return ALICE_COUNT + BOB_COUNT
```

Zdrojový kód 2: Výpočet ohodnocení dvojice uživatelů, metoda 1

```
1 COMPUTE_MEAL_SCORE_2(ALICE_COUNT, BOB_COUNT)
2   if (ALICE_COUNT < BOB_COUNT)
3     return ALICE_COUNT
4   else
5     return BOB_COUNT
6 }
```

Zdrojový kód 3: Výpočet ohodnocení dvojice uživatelů, metoda 2

```
1 COMPUTE_MEAL_SCORE_6(ALICE_COUNT, ALICE_THRESHOLD, BOB_COUNT,
     BOB_THRESHOLD)
2   if (ALICE_COUNT > ALICE_THRESHOLD ∧ BOB_COUNT > BOB_THRESHOLD)
3     if (ALICE_COUNT > BOB_COUNT)
4       return ALICE_COUNT
5     else
6       return BOB_COUNT
7   else
8     return 0
```

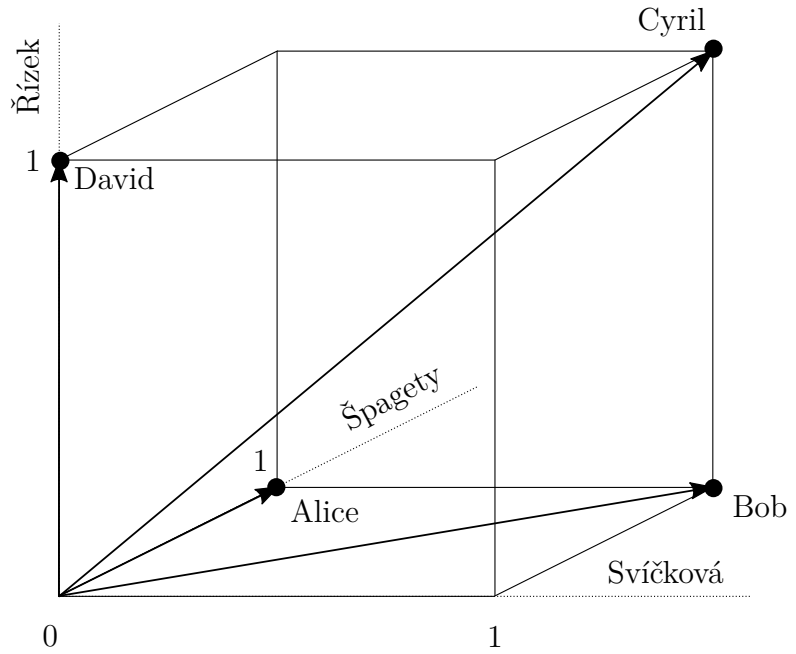
Zdrojový kód 4: Výpočet ohodnocení dvojice uživatelů, metoda 6

Jiný přístup spočívá ve využití eukleidovského prostoru. Předpokládejme tři jídla v databázi a čtyři uživatele, jejichž objednávky jsou určeny tabulkou 11. Protože máme v databázi tři jídla, dimenze eukleidovského prostoru je tři. Takový eukleidovský prostor si můžeme představit jako krychli (obr. 2). Každý uživatel je tak určen vektorem, jehož složky jsou 1, pokud si uživatel objednal jídlo odpovídající dimenzi, jinak 0. Vektor Alice je $\vec{a} = (0, 0, 1)$, vektor Boba $\vec{b} = (1, 0, 1)$, vektor Cyrila $\vec{c} = (1, 1, 1)$ a vektor Davida $\vec{d} = (0, 1, 0)$.

Úhly pro všechny dvojice uživatelů jsou vypočítány v tabulce 12. Všimněme si, že hodnoty odpovídají chápání úhlů mezi vektory v krychli, například, že Alice má mezi Bobem menší úhel než mezi Davidem. To odpovídá shodě historie uživatelů, tedy Alice má s Bobem větší shodu než s Davidem. Pokud by Alice

Tabulka 11: Příklad: Data objednávek pro výpočet úhlů mezi uživateli

Jídlo	Alice	Bob	Cyril	David
Svíčková	0	1	1	0
Řízek	0	0	1	1
Špagety	1	1	1	0



Obrázek 2: Příklad: Znázornění uživatelů v eukleidovském prostoru

Tabulka 12: Příklad: Úhly mezi vektory v eukleidovském prostoru (°)

	$\vec{a}$	$\vec{b}$	$\vec{c}$	$\vec{d}$
$\vec{a}$	$0,0$	$45,0$	$\approx 54,7$	$90,0$
$\vec{b}$	$45,0$	$0,0$	$\approx 35,2$	$90,0$
$\vec{c}$	$\approx 54,7$	$\approx 35,2$	$0,0$	$\approx 54,7$
$\vec{d}$	$90,0$	$90,0$	$\approx 54,7$	$0,0$

měla v historii objednanou navíc svíčkovou, její vektor a vektor Boba by byl stejný. Můžeme tedy prohlásit, že čím větší je shoda v historii, tím menší je úhel mezi vektory, kterými jsou uživatelé reprezentováni.

Vypočítáním úhlů mezi Alicí a všemi ostatními uživateli tedy získáme informaci o tom, který uživatel je Alici nejpodobnější. Jako míra podobnosti nám poslouží kosinus úhlu, tedy hodnota v intervalu $\langle 0; 1\rangle$, blížící se k 1 se zmenšující se vzdáleností mezi uživateli.

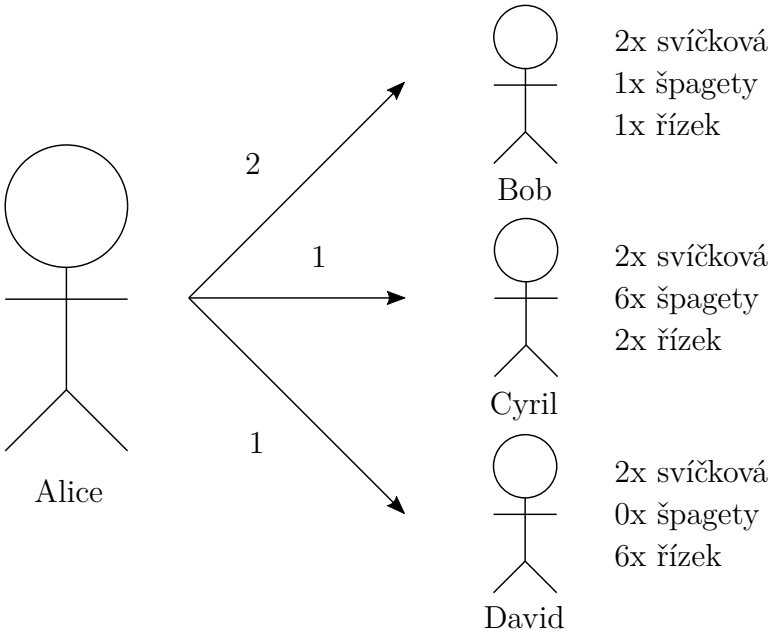
Pro potřeby našeho doporučovacího systému nám však nestačí eukleidovský prostor dimenze 3. Dimenze vektorového prostoru pro výpočet podobnosti Alice s ostatními uživateli musí odpovídat počtu jídel v databázi. Pro zmenšení dimenze však můžeme vyřadit ty složky vektorů odpovídající jídlům, která neměli oba uživatelé ani jednou. Touto optimalizací dimenzi řádově zredukujeme a výpočty tak budou méně náročné.

3.6.2 Vyhledání nejbližších uživatelů

Po sestavení všech dvojic uživatelů máme k dispozici uspořádaný seznam od toho s největší shodou (nejmenší vzdálenost) v historii po toho se shodou nejmenší (největší vzdálenost). Následně od těchto uživatelů získáme informace o historii jejich objednávek. Pokud u blízkého uživatele nalezneme jídlo, které je v dnešní nabídce, je toto jídlo jistým způsobem vhodné pro Alici.

Triviální je pracovat s objednávkami nejbližšího uživatele. Pokud z těchto jídel vybereme taková, která si uživatel objednal nejčastěji, získáme tak nejoblíbenější jídlo uživatele s nejpodobnějšími chuťovými preferencemi. Toto jídlo však ve většině případů nebude v dnešní nabídce. Více jídel získáme uspořádáním historie nejbližšího uživatele dle počtu objednávek a vyfiltrováním těch jídel, které jsou v dnešní nabídce.

Práce s uživatelem s největší shodou ale nemusí být vždy dobrou volbou. I pokud mají podobné chuťové preference, mohou v jejich historii existovat výjimky. V takové situaci by jistě bylo vhodné pro zpřesnění doporučení použít historii objednávek dalších blízkých uživatelů. Pokud mají nejbližší uživatelé stejnou vzdálenost k Alici, můžeme jednoduše použít sumu objednávek jídel všech



Obrázek 3: Příklad: Princip doporučování se znalostí chování uživatelů

takových uživatelů. Jaká jídla bychom ale měli doporučit v případě, že další uživatelé mají větší vzdálenost od Alice? Řešením je použít vzdálenost uživatelů jako váhy a následně podle nich upravit počty objednávek.

3.6.3 Sestavení seznamu doporučených jídel

Předpokládejme, že máme čtyři uživatele u_a, u_b, u_c, u_d a tři jídla m_{sv}, m_{sp}, m_r. K doporučení jídel Alici použijeme historie třech dalších uživatelů. Váha uživatelů (kterou získáme dle popsaných metod) a jídla, která si uživatelé objednali, jsou znázorněny na obrázku 3.

Sestavení doporučených jídel se tedy skládá nejméně ze dvou částí, a to z vyhledání určitého počtu uživatelů s podobnými preferencemi a následně z přiřazení ohodnocení jídlům, která mají tito uživatelé v historii. Nechť U je množina uživatelů a M je množina jídel. Potom pro doporučení uživateli $a \in U$ vypočítáme ohodnocení $S_{a,m}$ jídla $m \in M$:

$$S_{a,m} = \sum_{b \in U \setminus \{a\}} \left(\frac{d_{a,b}}{D_a} \cdot c_{m,b} \right),$$

kde $d_{a,b}$ je vzdálenost uživatele a k uživateli b, $c_{m,b}$ je počet objednávek jídla m uživatele b a D_a je suma vzdáleností všech uživatelů od uživatele a:

$$D_a = \sum_{b \in U \setminus \{a\}} d_{a,b}.$$

Pro situaci na obrázku 3 je ohodnocení jednotlivých jídel uvedeno v tabulkách 13, 14 a 15 podle počtu nejbližších uživatelů. Všimněme si, že pro každý počet

Tabulka 13: Příklad: Výpočet ohodnocení jídel pro jednoho uživatele

m	d_{u_a,u_b}	D_{u_a}	c_{m,u_b}	$S_{u_a,m}$
$\mathbf{m_{sv}}$	2	2	**2**	**2**
m_{sp}			1	1
m_r			1	1

Tabulka 14: Příklad: Výpočet ohodnocení jídel pro dva uživatele

m	d_{u_a,u_b}	d_{u_a,u_c}	D_{u_a}	c_{m,u_b}	c_{m,u_c}	$S_{u_a,m}$
m_{sv}	2	1	3	2	2	2
$\mathbf{m_{sp}}$				**1**	**6**	**≈2,6**
m_r				1	2	≈1,3

uživatelů je jídlo s nejvyšším ohodnocením vždy jiné. V případě doporučení s využitím dvou uživatelů je jídlo s nejvyšším ohodnocením m_{sp}, a to i přesto, že uživatel u_b s největší váhou si m_{sp} objednal pouze jednou. I přesto, že uživatel u_c má dvakrát menší váhu, jeho velký počet objednání m_{sp} má na výběr jídla velký vliv.

Můžeme říci, že čím jsou ostatní uživatelé blíže Alici, tím mají počty objednávek těchto uživatelů větší váhu. Obecně je tento princip nazýván *k-nearest neighbours* (kNN).

Poté, co máme seznam všech jídel seřazený vzestupně dle vypočteného ohodnocení, můžeme ho prohlásit za konečné doporučení.

3.6.4 Konfigurace

Jak bylo popsáno v kapitole 3.3, `MealRecommender` používá pro konfiguraci instance potomků třídy `Config`. Mimo základní parametry v `Config`, jsou navíc v `CollaborativeConfig` definované speciální parametry pro nastavení doporučovacího systému na základě znalosti uživatelů. Mezi tyto parametry patří:

- `nearestUsersMethod`

Tabulka 15: Příklad: Výpočet ohodnocení jídel pro tři uživatele

m	d_{u_a,u_b}	d_{u_a,u_c}	d_{u_a,u_d}	D_{u_a}	c_{m,u_b}	c_{m,u_c}	c_{m,u_d}	$S_{u_a,m}$
m_{sv}	2	1	1	4	2	2	2	2
m_{sp}					1	6	0	2
$\mathbf{m_r}$					**1**	**2**	**6**	**2,5**

  - typ metody nalezení blízkých uživatelů, popsáno v kapitole 3.6.1
  - jedna z konstant
    * `NearestUsersMethod.COUNT`
    * `NearestUsersMethod.VECTOR_SIM`
- `nearestUsersCount`
  - počet blízkých uživatelů, popsáno v kapitole 3.6.2
  - číslo z intervalu $\langle 1; |U| \rangle \subset \mathbb{Z} \cup \{-1\}$, kde U je množina uživatelů
  - pokud je -1, pak počet uživatelů není omezen
- `nearestMealsCount`
  - počet jídel blízkých uživatelů, kterým je vypočítáno ohodnocení
  - číslo z intervalu $\langle 1; |M| \rangle \subset \mathbb{Z} \cup \{-1\}$, kde M je množina jídel
  - pokud je -1, pak počet jídel není omezen
- `commonMealsThreshold`
  - minimální počet společných jídel pro nalezení blízkého uživatele
  - číslo z oboru $\mathbb{Z}_0^+$
- `addSimilar`
  - určuje, zda jsou do výsledku doporučení přidána doporučení podobných jídel
  - pravdivostní hodnota
  - více informací o tomto parametru je v kapitole 3.8.3
- `commonMealsType`
  - určuje, který typ metody pro výpočet ohodnocení dvojice uživatelů se použije, metody 1, 2 a 6 byly popsány v kapitole 3.6.1
  - číslo z intervalu $\langle 1; 7 \rangle \subset \mathbb{Z}$
- `commonMealsCount1` a `commonMealsCount2`
  - hodnoty prahů určující minimální počet objednání společného jídla prvního, resp. druhého uživatele ve dvojici
  - využívají se pouze pokud `commonMealsType`=5 nebo `commonMealsType`=6
  - číslo z oboru $\mathbb{Z}_0^+$

3.7 Implementace metody založené na znalosti uživatelů s využitím formální konceptuální analýzy

V této kapitole si vysvětlíme základy formální konceptuální analýzy, metody analýzy dat. Na základě těchto znalostí popíšeme způsob, kterým je možné pomocí formální konceptuální analýzy doporučovat položky. Následně sestavíme doporučovací systém a porovnáme ho s již sestavenými metodami.

3.7.1 Základní pojmy FKA

Formální konceptuální analýza (dále jen FKA) je metoda analýzy dat, která umožňuje získat jiný pohled na data. Právě skryté informace v datech o objednávkách získáme pomocí FKA. V této kapitole si popíšeme základní pojmy nutné

k pochopení popisované metody. Všechny pojmy z FKA, které jsou níže popsány pochází z velké části z knihy Bělohlávka [10]. Více informací o FKA je k dispozici právě v této knize nebo v článku, kterým Wille položil základy FKA [11].

Definice 1. *(Formální kontext). Formální kontext je trojice $\langle X, Y, I\rangle$, kde I je binární relace mezi množinami X a Y.*

Definice 2. *(Šipkové operátory). Pro formální kontext $\langle X, Y, I\rangle$ definujeme operátory $^{\uparrow} : 2^X \to 2^Y$ a $^{\downarrow} : 2^Y \to 2^X$ tak, že pro množiny $A \subseteq X$ a $B \subseteq Y$ platí:*

$$A^{\uparrow} = \{y \in Y |\ \textit{pro každý } x \in A : \langle x, y\rangle \in I\},$$
$$B^{\downarrow} = \{x \in X |\ \textit{pro každý } y \in B : \langle x, y\rangle \in I\}.$$

Dle definice 2 řekneme, že $A^{\uparrow}$ je množina všech atributů $y \in Y$, které jsou společné všem objektům z A a že $B^{\downarrow}$ je množina všech objektů, které sdílejí všechny atributy $x \in X$.

Definice 3. *(Formální koncept). Formální koncept v kontextu $\langle X, Y, I\rangle$ je dvojice $\langle A, B\rangle$ taková, že pro množiny $A \subseteq X$ a $B \subseteq Y$ platí:*

$$A^{\uparrow} = B \qquad a \qquad B^{\downarrow} = A.$$

Pro formální koncept (pojem) $\langle A, B\rangle$ v kontextu $\langle X, Y, I\rangle$, množinu A nazýváme *extent* (rozsah) a množinu B nazýváme *intent* (obsah).

Definice 4. *(Částečné uspořádání subconcept-superconcept (podpojem-nadpojem)). Pro formální koncepty $\langle A_1, B_1\rangle$ a $\langle A_2, B_2\rangle$ kontextu $\langle X, Y, I\rangle$ platí, že*

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \qquad \textit{právě když} \qquad A_1 \subseteq A_2 \textit{ (nebo, ekvivalentně, } B_2 \supseteq B_1\textit{).}$$

Pomocí relace $\leq$ můžeme o konceptu říct, že je obecnější, resp. konkrétnější než jiný koncept. Řekneme, že koncept $\langle A_1, B_1\rangle$ je menší nebo roven konceptu $\langle A_2, B_2\rangle$ (tedy $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$), pokud je $\langle A_1, B_1\rangle$ konkrétnější než $\langle A_2, B_2\rangle$, duálně, $\langle A_2, B_2\rangle$ obecnější než $\langle A_1, B_1\rangle$. Takovému konceptu $\langle A_1, B_1\rangle$ se říká *subconcept* konceptu $\langle A_2, B_2\rangle$ (budeme dále označovat českým termínem *podpojem*). Naopak, konceptu $\langle A_2, B_2\rangle$ se říká *superconcept* konceptu $\langle A_1, B_1\rangle$ (budeme dále označovat českým termínem *nadpojem*).

Definice 5. *(Konceptuální svaz). Nechť $\mathcal{B}(X, Y, I)$ je množina všech formálních konceptů pro kontext $\langle X, Y, I\rangle$, platí tedy:*

$$\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \in 2^X \times 2^Y | A^{\uparrow} = B, B^{\downarrow} = A\}.$$

Potom dvojici $\langle \mathcal{B}(X, Y, I), \leq\rangle$ nazýváme konceptuální svaz kontextu $\langle X, Y, I\rangle$.

Konceptuální svaz je tedy množina všech formálních konceptů uspořádaných dle jejich obecnosti.

Tabulka 16: Příklad tabulkových dat

	m_1	m_2	m_3	m_4	m_5
u_1	×	×	×		×
u_2			×		×
u_3	×		×		×
u_4	×	×		×	×
u_5		×	×	×	×

3.7.2 Formální kontext objednávek

Metoda FKA pracuje s tabulkovými daty, tedy daty, které lze reprezentovat tabulkou, kde v řádcích jsou objekty a ve sloupcích atributy. V našem případě jako objekty použijeme množinu uživatelů $U = \{u_1, \ldots, u_m\}$ a jako atributy množinu jídel $M = \{m_1, \ldots, m_n\}$. Část takových dat si můžeme představit pomocí tabulky 16.

V našem případě za formální kontext můžeme považovat trojici $\langle U, M, P\rangle$, kde P je binární relace mezi množinami U a M a pro kterou platí, že $\langle u, m\rangle \in P$ pokud uživatel $u \in U$ provedl objednávku jídla $m \in M$. Buňka tabulky 16 obsahuje křížek, pokud platí vztah mezi uživatelem a jídlem. Například křížek v buňce na řádku označeném u_4 a ve sloupci označeném m_2 značí „uživatel u_4 provedl objednávku jídla m_2“. Pokud dále v textu nebude uvedeno jinak, budeme předpokládat použití formálního kontextu $\langle U, M, P\rangle$.

Následně můžeme definovat operátory $^{\uparrow}$ a $^{\downarrow}$ pro formální kontext $\langle U, M, P\rangle$. Pro množinu $A \subseteq U$ řekneme, že $A^{\uparrow}$ jsou všechna jídla $m \in M$, která si objednali všichni uživatelé $u \in A$. Pro množinu $B \subseteq M$ řekneme, že $B^{\downarrow}$ jsou všichni uživatelé $u \in U$, kteří si objednali všechna jídla $m \in B$.

Formální koncept kontextu $\langle U, M, P\rangle$ je dvojice $\langle A, B\rangle$, kde $A \subseteq U$ a $B \subseteq M$ a pro kterou platí, že A obsahuje všechny uživatele, jenž si objednali všechna jídla z B a B obsahuje všechna jídla, která byla objednána všemi uživateli z A.

Dále jako $\mathcal{B}(U, M, P)$ si označíme všechny formální koncepty kontextu $\langle U, M, P\rangle$. Potom množina všech konceptů v $\langle U, M, P\rangle$ společně s částečným uspořádáním $\leq$ na $\mathcal{B}(U, M, P)$ je konceptuální svaz.

3.7.3 Formální koncepty vhodné k doporučování

Formálních konceptů v $\mathcal{B}(U, M, P)$ je velké množství. Neformálně jsou to dvojice všech skupin uživatelů společně s jídly, která si všichni uživatelé ve skupině objednali. Vhodných pro doporučování je však pouze zlomek všech konceptů. Hlavní myšlenku využití těchto konceptů a jejich definici popsali Boucher-Ryan a Bridge v [12]:

> *„Koncept označovaný entry-level objektu e je takový koncept, který*

obsahuje e v extentu a zároveň e není obsažen v extentu žádného podpojmu tohoto konceptu.[6] “

Pro formalizaci *entry-level* konceptu si definujme zobrazení $\text{Sub} : \mathcal{B}(U, M, P) \to 2^{\mathcal{B}(U,M,P)}$, pro které platí:

$$\text{Sub}(\langle A, B\rangle) = \{\langle X, Y\rangle | \langle X, Y\rangle \leq \langle A, B\rangle\}.$$

Pro formální koncept získáme pomocí zobrazení Sub množinu všech podpojmů. Potom můžeme definovat zobrazení $\text{Entry} : U \to \mathcal{B}(U, M, P)$ tak, že:

$$\text{Entry}(u) = \{\langle A, B\rangle \in \mathcal{B}(U, M, P) | u \in A, \forall c \in \text{Sub}(\langle A, B\rangle) : u \notin c\}.$$

Jinými slovy pro naše využití je *Entry-level* uživatele u vzhledem k částečnému uspořádání $\leq$ takový koncept, jenž má počet jídel v intentu největší ze všech konceptů, které uživatele u obsahují v extentu.

Pro každého uživatele u tak použitím $\text{Entry}(u)$ získáme jednoznačný koncept, který je vhodný pro doporučování, protože obsahuje:

- v intentu množinu všech jídel, která si uživatel objednal
- v extentu uživatele, kteří si objednali všechna jídla v intentu

Uživatele v extentu můžeme dělit do následujících kategorií podle toho, jaký je jejich *entry-level*.

***Entry-level* koncept uživatele je tento koncept**
Uživatelé si objednali stejnou množinu jídel, která je v intentu tohoto konceptu.

***Entry-level* koncept uživatele je podpojmem tohoto konceptu**
Uživatelé si objednali vlastní nadmnožinu jídel v intentu tohoto konceptu. S uživateli, pro které je tento koncept *entry-level* však mají společná všechna jídla v intentu tohoto konceptu.

3.7.4 Výpočet vzdáleností s využitím FKA

V kapitole 3.6.1 jsme pro sestavení doporučení počítali vzdálenosti mezi Alicí a všemi ostatními uživateli v databázi. Tento počet však můžeme zredukovat použitím metody FKA. Nejprve předpokládejme, že vzdálenost mezi uživateli je počet jejich společných jídel.

Množinu *entry-level* všech uživatelů budeme značit $\mathcal{E}(U, M, P)$, pro kterou platí:

$$\mathcal{E}(U, M, P) = \{\text{Entry}(u) | u \in U\}$$

[6] Entry-level concept of entity e is the unique concept for which e is a member of the extent and e is not a member of the extent of any subconcept.

Pro sestavení uživatele u získáme příslušný *entry-level* koncept a procházíme uživatele v extentu tohoto konceptu. Mezi uživatelem u a uživateli v extentu můžeme ihned získat vzdálenost. Hodnotu není nutné počítat, je to počet jejich společných jídel, který odpovídá počtu atributů v intentu. Získání vzdáleností těchto uživatelů znázorňuje pseudokód 5. Zlepšení metodou FKA spočívá v rychlém nalezením uživatelů, se kterými uživatel u sdílí všechna jídla.

```
1 FIND_ENTRY_LEVEL_PAIRS(USER)
2   PAIRS ← ∅
3   CONCEPT ← ENTRY-LEVELS[USER]
4   for each (ANOTHER_USER ∈ EXTENT(CONCEPT))
5     do PAIR ← NEW PAIR(USER, ANOTHER_USER)
6       PAIR.SCORE ← COUNT(INTENT(CONCEPT))
7       PAIRS.ADD(PAIR)
```

Zdrojový kód 5: Výpočet vzdálenosti k uživateli v entry-level konceptu

Pokud potřebujeme vypočítat vzdálenost i k uživatelům, kteří si neobjednali všechna stejná jídla jako uživatel u, můžeme vypočítat vzdálenosti i pro uživatele, kteří v extentu tohoto konceptu nejsou. K tomu nám nestačí pouze nalezený *entry-level* koncept. Protože tito uživatelé nejsou v extentu, jejich *entry-level* koncept není podpojmem, tedy množina jejich atributů v intentu není vlastní nadmnožinou atributů v tomto intentu. Naopak, množina společných jídel těchto uživatelů je vlastní podmnožinou atributů v tomto intentu, jejich *entry-level* je tedy nadpojmem. K získání společných jídel těchto uživatelů vypočítáme počet prvků průniku intentu *entry-level* konceptů těchto uživatelů a intentu *entry-level* konceptu uživatele u. Takový počet společných jídel označíme jako vzdálenost. Výpočet vzdáleností k těmito uživatelům znázorňuje pseudokód 6. I přesto, že tyto vzdálenosti musíme vypočítat, na rozdíl od metody bez použití FKA jde o menší počet výpočtů.

```
1 FIND_OTHER_PAIRS(USER)
2   PAIRS ← ∅
3   CONCEPT ← ENTRY-LEVELS[USER]
4   for each (ENTRY_LEVEL ∈ ENTRY_LEVEL_CONCEPTS)
5     do if USER(ENTRY_LEVEL) ∈ OBJECTS(CONCEPT)
6         CONTINUE
7       PAIR ← NEW PAIR(USER, USER(ENTRY_LEVEL))
8       PAIR.SCORE ← COUNT(ATTRIBUTES(CONCEPT(ENTRY_LEVEL)) ∩
            ATTRIBUTES(CONCEPT))
9       PAIRS.ADD(PAIR)
```

Zdrojový kód 6: Výpočet vzdálenosti k ostatním uživatelům

Pro výpočet formálních konceptů kontextu $\langle U, M, P \rangle$ použijeme algoritmus, který popsal Krajča a kol. v [13]. Tento je založený na algoritmu Close-by-One,

jenž navrhl Kuznetsov v [14]. Vždy, když pomocí procedury `GenerateFrom` získáme nový formální koncept, projdeme všechny uživatele v jeho extentu a pro každého provedeme proceduru `PROCESS_CONCEPT` zapsanou pseudokódem 7. Po získání všech formálních konceptů tak máme pro každého uživatele jeho *entry-level* koncept, který můžeme využít pro výpočet vzdáleností.

```
1 PROCESS_CONCEPT(NEW)
2   for each (E ∈ EXTENT(NEW))
3     do CURRENT ← ENTRY-LEVELS[E]
4       if CURRENT = ∅
5         ENTRY-LEVELS[E] = CURRENT
6       else
7         if (COUNT(INTENT(NEW)) > COUNT(INTENT(CURRENT)))
8           ENTRY-LEVELS[E] = CURRENT
```

Zdrojový kód 7: Zpracování formálního konceptu

Výpočet konceptů je výpočetně náročný a je neefektivní vytvářet při každém doporučení celý konceptuální svaz. Proto je vhodné si potřebná data vypočítat jednou a následně aktualizovat pouze v případě změny v datech. Protože vzdálenosti vypočítáváme pouze z množin *entry-level* konceptů, můžeme si množinu těchto konceptů uložit a následně ji opakovaně využívat pro doporučování. Množina *entry-level* konceptů bude mít tedy funkci *modelu*.

3.8 Implementace hybridního doporučovacího systému

V této kapitole sestavíme doporučovací systém, který se skládá z více doporučovacích systémů. Konkrétně jde o doporučovací systémy, které jsme sestavili v kapitolách 3.5 a 3.6. Následně provedeme vylepšení doporučovacího systému z předchozí kapitoly za pomoci dat o uživatelích.

3.8.1 Kombinace systémů

Hybridní doporučovací systém je možné z více systémů sestavit několika způsoby. V této kapitole si ukážeme některé z možných postupů.

Doporučovací systémy v předchozích kapitolách jsou sestaveny tak, aby v případě nedostatečného množství dat doporučení nesestavily. V případě 3.5 je to způsobeno nedostatečným počtem objednávek v historii, zatímco v 3.6 jde o situaci, kdy uživatel nemá dostatečný počet blízkých uživatelů. I přesto, že spolu tyto situace souvisí, v některých případech se může stát, že jeden doporučovací systém jídla doporučí, zatímco druhý nikoliv.

Sestavíme tedy takový hybridní doporučovací systém, který bude využívat více nezávislých doporučovacích systémů. V případě, že jeden doporučovací systém doporučení nevrátí, použije se doporučovací systém, který je další v pořadí. To může například řešit situaci, kdy uživatel nemá žádné blízké uživatele, ale má

obsáhlou historii objednávek. V takovém případě se nevyužije doporučení systému založeném na uživatelích, ale doporučení systému založeného na obsahu.

Další možnost, jak sestavit hybridní doporučovací systém je vždy využít výstupy všech předaných doporučovacích systémů. Pro každý předaný doporučovací systém sestaví seznam doporučených jídel a u všech porovná jídlo s nejvyšším ohodnocením. Pokud se seznamy doporučení shodují v první položce, je seznam s touto položkou použitý jako seznam doporučení hybridního doporučovacího systému. V případě, že princip srovnávání první položky je příliš omezující, můžeme podmínku zmírnit a akceptovat i výsledky, které mají stejné jídlo na nižších pozicích.

3.8.2 Přidání znalosti uživatelů doporučovacímu systému se znalostí obsahu

Doporučovací systém sestavený v kapitole 3.5 můžeme vylepšit daty o objednávkách. Všechna jídla, pro která počítáme ohodnocení, má ve výpočtech stejnou váhu. Pokud však máme informace o objednávkách, můžeme prohlásit, jaké jídlo si objednalo nejvíce a jaké jídlo nejméně uživatelů. To, které si objednalo nejvíce uživatelů, můžeme nazvat populárním či obecným. Takové jídlo není pro doporučování tak důležité než jídlo, které je specifické a ne tak populární. Proto zavedeme IDF koeficient pro každé jídlo.

Pro IDF koeficient platí, že čím více uživatelů si dané jídlo objedná, tím bude jeho hodnota menší. Jinak řečeno, IDF koeficient bude nabývat menších hodnot pro populární jídla a naopak. I přesto, že se stále jedná o doporučovací systém na základě obsahu, je zahrnut do této kapitoly, protože navíc využívá data o uživatelích, která doporučovací systémy na základě obsahu běžně nemají k dispozici.

Funkci $\mathrm{IDF}_M : M \to \mathbb{R}$ pro výpočet IDF koeficientu jídla $m \in M$ si definujeme:

$$\mathrm{IDF_M}(m) = \log \frac{|U|}{\sum_{u \in U} U(u, m)},$$

kde

$$U(u, m) = \begin{cases} 1 & \text{uživatel } u \text{ si objednal jídlo } m, \\ 0 & \text{jinak.} \end{cases}$$

Následně můžeme upravit hodnotící funkci tak, aby snižovala váhu populárním jídlům a zvyšovala jídlům, které si objednalo málo uživatelů. Funkci $R_{\mathrm{IDF}_M}(m)$, která opět přiřazuje hodnocení jídlu m definujme:

$$R_{\mathrm{IDF}_M}(m) = \mathrm{IDF}_M(m) \cdot \sum_{w \in W(m)} (P(w) \cdot \mathrm{IDF}_W(w)).$$

3.8.3 Přidání znalosti obsahu doporučovacímu systému se znalostí uživatelů

Naopak, hybridní doporučovací systém můžeme vytvořit z doporučovacího systému se znalostí uživatelů tak, že systému přidáme znalost obsahu. Protože máme z kapitoly 2.4 informaci o tom, která jídla jsou si podobná, můžeme do doporučení přidat i ta jídla, která jsou jídlům v doporučení podobná.

Takovou úpravou dojde k zvětšení seznamu doporučení a tedy zvýšení šance, že doporučené jídlo je v dnešní nabídce.

3.8.4 Konfigurace

Pro metodu kombinace systémů podobně jako již popsané doporučovací systémy mají i hybridní doporučovací systémy vlastní konfiguraci. K obecným konfiguračním proměnným z předchůdce `Config` je zde pouze seznam konfigurací doporučovacích systémů, které hybridnímu systému předáváme. Hybridní doporučovací systém při inicializaci z těchto konfigurací vytvoří instance doporučovacích systémů a s těmi následně pracuje.

V případě úpravy doporučovacího systému se znalostí obsahu se jedná stále o `ContentBasedRecommender`, který ale používá jako `distanceMethod` konstantu `DistanceMethod.WORD_AND_MEAL_IDF` a je možné ho konfigurovat stejným způsobem jako ostatní doporučovací systémy se znalostí obsahu. Upravený doporučovací systém se znalostí uživatelů je možné konfigurovat stejným způsobem, přidání podobných jídel je provedeno nastavením hodnoty `addSimilar` na `true`.

Tabulka 17: Srovnání minimálního počtu různých jídel pro získání doporučení

ID	ID_P	`diff`	Správně (%)	Špatně (%)	Nedoporučeno (%)
a	–	**0**	**28,5**	**54,2**	**17,3**
–	–	1	28,5	54,2	17,3
–	–	2	28,5	54,2	17,3
–	–	3	28,4	53,3	18,3
–	–	4	27,3	51,9	20,7
–	–	5	26,5	49,9	23,6

4 Zhodnocení

V závěrečné kapitole je zhodnocena implementace doporučovacích systémů. Jsou zde popsány rozdíly v úspěšnosti jednotlivých metod, ale i některé zajímavé konfigurace.

Zhodnocení každé metody je podpořeno tabulkami znázorňující srovnání úspěšnosti. Každá tabulka obsahuje sloupce *Správně*, *Špatně* a *Nedoporučeno*. Tyto hodnoty korespondují s hodnotami, které jsme si popsali v kapitole 3.2. Řádky tabulek reprezentují konfigurace doporučovacích systémů. Některé konfigurace jsou označeny identifikátorem ve sloupci ID. Tyto identifikátory jsou následně použity pro nové konfigurace, které jsou na nich založené. Na jaké konfiguraci je jiná konfigurace založená určuje identifikátor ve sloupci ID_P. Kompletní definice pojmenovaných konfigurací jsou k dispozici na přiloženém CD (příloha D) ve třídě `cz.novaklukas.dp.config.Configs`. Ostatní sloupce v tabulkách se liší dle typu srovnání a jsou popsány v kapitole 4.

4.1 Zhodnocení metody založené na znalosti obsahu

Tabulka 17 znázorňuje změnu úspěšnosti při zvyšování minimálního počtu různých jídel pro získání doporučení. Všimněme si, že při zvyšování minima úspěšnost klesá. Můžeme tak prohlásit, že systém doporučuje nejúspěšněji v případě, že sestavuje doporučení i těm uživatelům, kteří měli pouze jedno jídlo (nezávisle na počtu objednávek). Můžeme se domnívat, že při zvýšení minima systém doporučuje pravděpodobně lépe uživatelům, kteří si objednali více jídel, avšak kvůli nedoporučení ostatním je jeho úspěšnost nižší.

Podobná je situace i u změně minima počtu objednávek nutných pro zařazení klíčových slov do uživatelského profilu, jenž znázorňuje tabulka 18. I zde se nám potvrzuje domněnka, že výhodnější je neomezovat.

Mírné zlepšení můžeme pozorovat při změně jednoduché metody na metodu využívající IDF. Zvýšení úspěšnosti si všimněme v tabulce 19. To můžeme přisuzovat principu výpočtu IDF, který je popsaný v kapitole 3.5.2, tedy že klíčová slova, která se vyskytují v méně jídlech, mají větší váhu.

Tabulka 18: Srovnání počtu objednávek nutných pro zařazení klíčových slov do uživatelského profilu

ID	ID_P	`count`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	a	0	28,5	54,2	17,3
–	a	1	28,5	54,2	17,3
–	a	2	28,5	54,2	17,3
–	a	3	24,2	46,0	29,8
–	a	4	19,5	37,7	42,8
–	a	5	15,8	31,2	52,9

Tabulka 19: Srovnání úspěšnosti po využití IDF koeficientu slov

ID	ID_P	`idf`	Správně (%)	Špatně (%)	Nedoporučeno (%)
b	**a**	✓	**31,2**	**51,5**	**17,3**
–	a	–	28,8	53,9	17,3

4.2 Zhodnocení metody založené na znalosti uživatelů

Tabulka 20 znázorňuje srovnání typu metody pro výpočet ohodnocení dvojice uživatelů. Sloupec `type` určuje typ metody nalezení blízkých uživatelů. Tyto metody byly popsány v 3.6.1. Všimněme si, že změnou typu metody se úspěšnost příliš nemění. To si můžeme vysvětlit faktem, že většina uživatelů si stejné jídlo objednává zřídka a tedy rozdíly mezi počty jídel uživatelů ve dvojici vliv na úspěšnost nemají. Situace se však mění při ořezávání uživatelů na základě počtů objednávek jídla. Tyto počty reprezentují hodnoty ve sloupci `value` a korespondují s prahy určující minimální počet objednávek společného jídla uživatelem ve dvojici. Zjišťujeme, že čím více ořezáváme uživatele s malým počtem objednávek, tím klesá úspěšnost doporučování, a to zejména z důvodu nedostatku dat o objednávkách. To nám potvrzuje počet nedoporučení, který poměrně s klesající úspěšností narůstá. Zajímavá skutečnost je i ta, že ořezáváním uživatelů dochází k zvětšování poměru mezi správným a špatným doporučením. To při konfiguraci na třetím řádku nabývalo hodnoty 56 %, zatímco na posledním hodnoty 78 %. Zde si můžeme klást otázku: „Je výhodnější doporučit špatně nebo nedoporučit vůbec?“ Toto rozhodnutí však zřejmě záleží na oblasti použití a provozovateli doporučovacího systému.

V tabulce 21 je znázorněno, jak se s přidáním počtu blízkých uživatelů a jídel těchto uživatelů zvyšuje počet správných doporučení. Sloupce `users` a `meals` určují počet blízkých uživatelů a jídel a korespondují s `nearestUsersCount` a `nearestMealsCount`, hodnotami, které byly popsány v kapitole 3.6.2. Znak „•“ zde značí výběr všech uživatelů, resp. jídel. Všimněme si, že při využití pouze

Tabulka 20: Srovnání typu metody pro výpočet ohodnocení dvojice uživatelů

ID	ID_P	`type`	`value`	Správně (%)	Špatně (%)	Nedoporučeno (%)
c	–	**1**	–	**25,0**	**44,3**	**30,7**
–	–	2	–	25,0	44,2	30,8
–	–	6	0 0	25,0	44,2	30,8
–	–	6	0 1	23,2	41,8	35,0
–	–	6	0 2	19,5	29,2	51,4
–	–	6	1 0	10,8	15,9	73,3
–	–	6	1 1	10,7	15,3	73,9
–	–	6	1 2	9,4	12,0	78,7

Tabulka 21: Srovnání parametrů jednoduchého sestavení dvojic uživatelů

ID	ID_P	`users`	`meals`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	c	1	1	1,4	1,4	97,2
–	c	1	30	10,3	11,9	77,8
–	c	1	•	18,3	26,9	54,8
–	c	30	1	2,8	4,7	92,6
–	c	30	30	23,6	44,3	32,1
–	c	30	•	23,3	45,5	31,1
–	c	•	1	7,4	9,2	83,4
d	**c**	•	**30**	**25,1**	**44,2**	**30,7**
–	c	•	•	25,0	44,2	30,8

nejbližšího uživatele máme počet správných doporučení velmi malý, přidáním více uživatelů se však tento počet zvyšuje. Ještě více je znát rozdíl přidání více jídel těchto uživatelů do výpočtu doporučení. K podobným závěrům jsme došli i v příkladu. Můžeme tedy říci, že čím více blízkých uživatelů a čím více jejich jídel využijeme, tím máme přesnější doporučení.

Tabulka 22 znázorňuje srovnání podobné tomu z předchozího odstavce s tím rozdílem, že nyní jsou sestavovány dvojice uživatelů s využitím eukleidovského prostoru. Tato tabulka a tabulka 21 se liší jen minimálně, což můžeme přisoudit tomu, že obě metody využívají data velmi podobně.

V tabulce 23 je znázorněno, jak se měnila úspěšnost při zvýšení prahu minimálního počtu společných jídel. V případě, že se počet společných jídel rovná nebo je vyšší než tato hodnota, pak je dvojici přiřazeno nenulové ohodnocení. Zde si všimněme podobné situace, jako při srovnání typu metody pro výpočet ohodnocení dvojice uživatelů. Zvyšováním prahu klesá úspěšnost, ale zvyšuje se poměr mezi správným a špatným doporučením. Z toho můžeme vyvodit, že uživatele můžeme shlukovat podle preferencí s pouze jedním společným jídlem.

Tabulka 22: Srovnání parametru sestavení dvojic uživatelů s využitím eukleidovského prostoru

ID	ID_P	`users`	`meals`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	c	1	1	1,4	1,4	97,2
–	c	1	30	10,3	11,9	77,8
–	c	1	•	18,3	26,9	54,8
–	c	30	1	2,8	4,7	92,6
–	c	30	30	23,4	44,4	32,2
–	c	30	•	23,4	45,3	31,2
–	c	•	1	9,7	10,4	80,0
e	**c**	•	**30**	**25,1**	**44,2**	**30,7**
–	c	•	•	24,6	41,9	33,5

Tabulka 23: Srovnání minimálního počtu společných jídel pro nalezení blízkých uživatelů

ID	ID_P	`meals`	Správně (%)	Špatně (%)	Nedoporučeno (%)
f	**d**	**1**	**25,1**	**44,2**	**30,7**
–	d	2	24,1	41,7	34,3
–	d	3	22,5	38,7	38,8
–	d	4	20,8	34,7	44,5
–	d	5	19,1	30,3	50,6

Tabulka 24: Srovnání výpočtu vzdáleností mezi uživateli pomocí FKA

Čas	FKA	Správně (%)	Špatně (%)	Nedoporučeno (%)
00:27:602	–	21,2	38,8	39,9
01:33:730	✓	21,2	38,8	39,9

Tabulka 25: Srovnání výpočtu vzdáleností mezi uživateli, kteří se nacházejí v *entry-level* konceptu, pomocí FKA

Čas	FKA	Správně (%)	Špatně (%)	Nedoporučeno (%)
00:01:691	✓	11,9	9,8	78,4

4.3 Zhodnocení metody založené na znalosti uživatelů s využitím FKA

Pro srovnání metody s využitím FKA implementujeme v metodě bez využití FKA výpočet vzdálenosti takovou, aby vzdálenost mezi uživateli byl počet jejich společných jídel. Jde tedy o stejnou vzdálenost, kterou jsme využili v metodě s využitím FKA.

Protože obě dvě metody využívají stejnou funkci pro výpočet vzdálenosti, úspěšnost obou metod je totožná. Je zde však řádový rozdíl v čase. Doporučení pomocí metody s využitím FKA trvá výrazně delší dobu, zejména je to však kvůli výpočtu nového konceptuálního svazu pro každé doporučení. Využijeme proto předvýpočtu modelu, jenž obsahuje všechny *entry-level* koncepty. Takové srovnání metod je znázorněno v tabulce 24. Všimněme si, že i s předvýpočtem modelu trvalo doporučení s metodou výpočtu vzdáleností mezi všemi uživateli pomocí FKA téměř o minutu více. To je způsobeno především tím, že samotné prohledávání *entry-level* konceptů je výpočetně náročná operace.

Počítejme tedy pouze vzdálenosti k uživatelům, kteří se nacházejí v *entry-level* konceptu. Tato operace je triviální, jde pouze o nalezení *entry-level* konceptu a získání počtu jídel v intentu. Potom úspěšnost této metody je znázorněna v tabulce 25. Všimněme si, že úspěšnost proti předchozí metodě klesla téměř o polovinu. To je způsobeno tím, že jsme počítali vzdálenosti pouze pro uživatele, jejichž množina objednávek je vlastní nadmnožina jídel v intentu. Protože je tato operace velmi jednoduchá, doporučení je sestaveno za necelé dvě sekundy.

4.4 Zhodnocení hybridního doporučování

Metody hybridního doporučování popsané v kapitole 3.8 jsou srovnány v tabulce 26. První dva řádky popisují úspěšnost v případě kombinace systémů tak, že se v případě nevytvoření doporučení jednoho doporučovacího systému použije

Tabulka 26: Srovnání metod hybridního doporučovacího systému

ID	ID_P	Systémy	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	–	f→b	25,2	70,1	4,7
g	–	**b→f**	**36,7**	**58,5**	**4,7**
h	–	**f+b**	**9,0**	**6,5**	**84,6**

Tabulka 27: Srovnání úspěšnosti po přidání znalosti uživatelů do doporučovacího systému se znalostí obsahu

ID	ID_P	`add`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	b		31,2	51,5	17,3
i	**b**	✓	**35,4**	**47,4**	**17,3**

doporučení dalšího systému v pořadí. Tato metoda byla vyzkoušena na nejúspěšnější metodě doporučovacího systému se znalostí uživatelů a nejúspěšnější metodě doporučovacího systému se znalostí obsahu. V případě prioritního doporučení prvně jmenovaným systémem zůstala úspěšnost stejná. To je zřejmě dáno tím, že pokud nemá takový systém pro uživatele k dispozici dostatek dat pro nalezení jeho blízkých uživatelů, nemá pravděpodobně ani dostatek dat k vytvoření doporučení pouze se znalostí obsahu.

Při použití doporučovacího systému se znalostí obsahu jako prioritního však ke zlepšení úspěšnosti došlo. I přesto, že doporučovací systém měl k dispozici málo dat o objednávkách uživatele a nevytvořil doporučení, tomu stejnému uživateli dokázal doporučovací systém se znalostí uživatelů vyhledat nejbližší uživatele a na základě jejich objednávek doporučení sestavit.

Druhá možnost kombinace, a to využití výsledků obou doporučovacích systémů má úspěšnost velmi malou. Všimněme si však skutečnosti, že u této metody je více správných než špatných doporučení. Vysvětlení je zjevné, takový doporučovací systém totiž doporučí jen v případě shody blízkých uživatelů a klíčových slov aktuálního uživatele. Takový doporučovací systém je tedy vhodné použít v případě, kdy je důležité doporučit správně.

Hybridizační metoda, kterou jsme si popsali v kapitole 3.8.2, spočívala ve využití dat o uživatelích v doporučovacím systému se znalostí obsahu. Konkrétně jsme ke každému jídlu vypočítali IDF dle popularity jídel. Tato hybridizace zvýšila doporučovací systém se znalostí obsahu o 4 procentní body. To bylo způsobeno použitím vah jídel a principu, že populární jídlo má menší váhu než jídlo, které si objednalo menší množství uživatelů. Změna úspěšnosti je znázorněna v tabulce 27.

Nakonec tabulka 28 srovnává, jak se změnila úspěšnost doporučovacího systému se znalostí uživatelů tím, že jsme do doporučení přidali podobná jídla.

Tabulka 28: Srovnání úspěšnosti po přidání znalosti obsahu do doporučovacího systému se znalostí uživatelů

ID	ID_P	`sim`	Správně (%)	Špatně (%)	Nedoporučeno (%)
j	**c**	✓	**41,5**	**53,8**	**4,7**
–	f	✓	40,7	54,6	4,7

Přidáním znalosti obsahu do doporučovacího systému se znalostí uživatelů jsme získali další hybridizační metodu. Všimněme si, že tento krok zvýšil úspěšnost o 15 procentních bodů. To nám potvrzuje, že jsme minimálně část podobných jídel detekovali úspěšně. V určitém počtu případů tedy doporučovací systém doporučil správně podobné jídlo tomu, které si uživatel objednal dříve.

4.5 Srovnání obecných parametrů

Mimo specifických parametrů pro jednotlivé metody je možné konfigurovat doporučovací systémy pomocí obecných parametrů, které byly popsány v kapitole 3.3. V této kapitole si popíšeme jaký měla změna jednotlivých parametrů vliv na úspěšnost doporučovacích systémů.

Jak bylo popsáno v kapitole 3.2, můžeme určit poměr trénovací k testovací množině. Závislost úspěšnosti doporučovacích systémů na poměru těchto množin znázorňuje tabulka 29. Všimněme si, že v případě doporučovacího systému se znalostí uživatelů, je nejúspěšnější poměr trénovací k testovací množině 7:3. Menší i větší poměr trénovací množiny znamená menší úspěšnost. V případě doporučovacího systému se znalostí obsahu je sice vyšší poměr trénovací množiny úspěšnější, nicméně v takovém případě nemáme příliš vypovídající hodnotu o úspěšnosti, neboť byla otestována pouze na 10 % objednávek. Potvrdila se nám tak, že běžně používaný poměr 7:3 je možné využít i v této oblasti.

Další obecný parametr nezávislý na doporučovacím systému je typ metody detekce stejných jídel, který jsme popsali v kapitole 2.3. V tabulce 30 je znázorněno, že detekce stejných jídel vylepšila úspěšnost doporučovacího systému. Úspěšnost jednotlivých metod je srovnatelný. V případě doporučovacího systému se znalostí uživatelů je úspěšnější použití metody regulárních výrazů. Tato metoda má však nevýhodu v tom, že není příliš obecná. Regulární výrazy pro tuto metodu bylo nutné definovat manuálně dle popisů jídel. Teoreticky by bylo možné vhodným vylepšením těchto regulárních výrazů úspěšnost ještě zvýšit, nicméně obecně vhodnější volba je použít metodu Levenshteinovy vzdálenosti, kterou není potřeba podobně *učit* data.

K základním transformacím popisů jídel patří odstranění přílohy, které bylo popsáno v kapitole 2.3.3. Tabulka 31 znázorňuje, jak se změnila úspěšnost doporučovacích systémů v závislosti na této metodě. V případě doporučovacího systému se znalostí uživatelů se úspěšnost zvýšila o 22 %. Můžeme předpokládat,

Tabulka 29: Srovnání poměrů rozdělení databáze na trénovací a testovací množiny

ID	ID_P	`train`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	j	40 %	15,0	25,6	59,3
–	j	50 %	32,9	52,7	14,4
–	j	60 %	40,7	52,0	7,3
–	**j**	**70 %**	**41,5**	**53,8**	**4,7**
–	j	80 %	41,7	55,8	2,5
–	j	90 %	32,6	67,1	0,3
–	j	40 %	11,1	21,4	67,5
–	j	50 %	20,8	38,0	41,2
–	j	60 %	27,3	48,9	23,7
–	**j**	**70 %**	**31,1**	**51,6**	**17,3**
–	j	80 %	36,6	52,3	11,1
–	j	90 %	43,8	53,8	2,4

Tabulka 30: Srovnání úspěšnosti metod detekce stejných jídel

ID	ID_P	`method`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	j	`OFF`	38,9	56,4	4,7
–	j	`REGEX`	41,9	53,4	4,7
–	j	`LEVEN`	41,5	53,8	4,7
–	b	`OFF`	29,5	50,4	20,1
–	b	`REGEX`	31,1	50,9	18,0
–	b	`LEVEN`	31,2	51,5	17,3

Tabulka 31: Srovnání úspěšnosti po odstranění přílohy

ID	ID_P	`remove`	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	j	–	33,9	61,4	4,7
–	j	✓	41,4	53,8	4,7
–	**b**	–	**34,7**	**50,7**	**14,6**
–	b	✓	31,2	51,6	17,3

Tabulka 32: Srovnání úspěšnosti metod detekce podobných jídel

ID	ID_P	`method`	Počet s.	Správně (%)	Špatně (%)	Nedoporučeno (%)
–	j	`E`	353	25,0	44,3	30,7
–	j	`I 1`	8	16,2	79,1	4,7
–	j	`I 2`	54	36,6	58,7	4,7
–	j	`I 3`	178	38,5	56,8	4,7
–	j	`I 4`	260	33,9	61,4	4,7
–	j	`I 5`	316	29,6	62,8	7,5
–	j	`D 0,2`	8	22,4	72,9	4,7
–	j	`D 0,5`	174	41,2	54,1	4,7
–	**j**	**`D 0,55`**	**185**	**41,5**	**53,8**	**4,7**
–	j	`D 0,6`	224	36,2	59,1	4,7
–	j	`D 0,8`	331	29,5	55,2	15,3

že detekce stejných jídel s pomocí odstranění přílohy je úspěšnější, protože uživatelé si objednávají hlavní část jídla, nikoliv přílohu. Naopak doporučovacímu systému se znalostí obsahu se s odstraněním přílohy úspěšnost snížila. Je zřejmé, že tento systém, jehož princip spočívá právě v práci s klíčovými slovy, potřebuje k doporučování co nejvíce klíčových slov. Odstraněním přílohy však byl tento počet zredukován.

Další parametr, který patří do obecných parametrů, jenž nezávisí na metodě doporučovacího systému, koresponduje s typem detekce podobných jídel. V kapitole 2.4 jsme si popsali tři metody. Jejich srovnání je znázorněno v tabulce 32. Nejjednodušší z nich je metoda, jenž považuje za podobná jídla ta, která obsahují stejná klíčová slova. Ta má poměrně nízkou úspěšnost. Vyšší úspěšností se vyznačuje metoda zmírňující podmínku a detekující podobná slova na základě počtu prvků průniku klíčových slov. Tři totožná klíčová slova pro detekci podobných slov má o 54 % větší úspěšnost než jednoduchá metoda. Vylepšení je dále možné použitím metody Dice koeficientu, jehož výpočet na rozdíl od metody průniku využívá počet klíčových slov v jídle. Prahem 0,55 byla zvýšena úspěšnost o dalších 12 % proti jednoduché metodě. Hodnoty ve sloupci `Počet s.` uvádí počet vytvořených skupin podobných jídel. Část těchto skupin je znázorněna v tabulce 34.

Tabulka 33: Srovnání úspěšnosti vybraných konfigurací

ID	Správně (%)	Špatně (%)	Nedoporučeno (%)
a	28,5	54,2	17,3
b	31,2	51,5	17,3
c	25,0	44,3	30,7
d	25,1	44,2	30,7
e	25,1	44,2	30,7
f	25,1	44,2	30,7
g	**36,7**	**58,5**	**4,7**
h	9,0	6,5	84,6
i	**35,4**	**47,4**	**17,3**
j	**41,5**	**53,8**	**4,7**

4.6 Zhodnocení metod doporučovacích systémů

V tabulce 33 jsou znázorněny některé konfigurace doporučovacích systémů a jejich úspěšnost. Všimněme si, že nejvyšší úspěšnost mají hybridizační metody. Jde o metodu prioritní kombinace, metoda se znalostí uživatelů s přidanými znalostmi o obsahu a metoda se znalostí obsahu s přidanými znalostmi o uživatelích. Můžeme tak prohlásit, že hybridizační metody mají největší úspěšnost, jelikož kombinacemi více metod jsou snižovány nevýhody metod samotných.

Závěr

Podařilo se naplnit cíle této diplomové práce – nastudovat problematiku doporučovacích systémů a implementovat doporučovací systém v oblasti objednávání jídel. Práce přinesla přehled přístupů, které je vhodné použít v této oblasti doporučování. To bylo možné díky získání testovacích dat, jenž poskytlo vedení Správy kolejí a menz Univerzity Palackého.

Bylo implementováno několik metod doporučovacích systémů, přičemž nejlepší výsledek je 41 %. To znamená, že doporučovací systém sestavil takové množství doporučení, jenž nabídlo uživateli jídlo, které si následně objednal. To bylo možné zjistit pomocí rozdělení dat na trénovací a testovací množiny. Nejlepších výsledků dosahovaly zejména hybridizační metody, tedy kombinace několika přístupů do jednoho doporučovacího systému. Následně byl sestaven doporučovací systém na základě formální konceptuální analýzy, bylo však zjištěno, že efektivně doporučuje pouze v určitých případech.

Otestování v reálné aplikaci neproběhlo z důvodu nutnosti nasadit implementaci na informační systém, jenž využívá denně tisíce uživatelů. K případnému nasazení popsaného řešení je nutná spolupráce třetích stran. Nejsnadnější řešení uvedení do praxe a zároveň to s nejmenší mírou potřeby spolupráce je implementovat doporučovací systém založeného na obsahu do mobilní aplikace *MobilKredit*, jenž byla vytvořena v rámci autorovy bakalářské práce. Touto problematikou se zabývá příloha C. Autor práce bude usilovat o to, aby takové nasazení do praxe proběhlo.

A Podobná jídla

V kapitole 2.4 byly popsány metody detekce podobných jídel, které uspořádávají jídla do skupin. V tabulce 6 je znázorněna část takových skupin, které byly vytvořeny metodou `MealSimilarityMethod.DICE` s prahem `0,55` za použití metody detekce stejných jídel `MealNormalizeMethod.LEVEN`.

Tabulka 34: Ukázka vytvořených skupin podobných jídel

Popis
400g Boloňské špagety s vepř. masem syp. sýrem, nápoj VEG, BEZLEP 350g Mexické zeleninové rizoto (čínská zelenina VOK) syp. parmazánem, nápoj STUDENTSKÝ TIP: BEZLEP 350g Rizoto z kuřecího masa syp. sýrem, nápoj 400g Špagety s tuňákem syp. sýrem, nápoj STUDENTSKÝ TIP: BEZLEP 350g Srbské rizoto s vepř. masem syp. sýrem, steril. okurek, nápoj 400g Milánské špagety s vepřovým masem syp. sýrem, nápoj VEG 350g Toskánské špagety syp. sýrem, nápoj VEG, BEZLEP 350g Zeleninové rizoto syp. sýrem, steril. okurek, nápoj 400g Milánské špagety s vepřovým masem syp. sýrem, nápoj VEG, BEZLEP 350g Mexické zeleninové rizoto sypané sýrem (čínská zelenina, kečup), steril. okurek, nápoj BEZLEP 350g Rizoto z vepř. masa se smetanou syp. parmazánem, steril. okurek, nápoj A 7, 9
100g Karlovarský kotouč (vepřové maso, uzenina, steril. okurky), šťouchané brambory, nápoj 100g Vepřový závitek (uzenina, vejce, steril. okurek), rýže, nápoj
100g Hovězí maso vařené, koprová omáčka, houskový knedlík, nápoj 100g Hovězí maso vařené, rajská omáčka, houskový knedlík, nápoj
100g Smažený kuřecí řízek v sýrovém těstíčku, bramb. kaše, nápoj STUDENTSKÝ TIP: 100g Kuřecí prsa v pikantním těstíčku, bramb. kaše, nápoj 150g Moravský vepřový řízek (plněný smažený v těstíčku), brambory maštěné, obloha, nápoj 100g Smažený kuřecí řízek v pikantním těstíčku, brambory maštěné, nápoj 100g Smažený kuřecí řízek, brambory maštěné, nápoj 125g Smažený sekaný řízek se sýrem, bramb. kaše, nápoj 125g Smažený sekaný vepř. řízek se sýrem, brambory maštěné, nápoj 120g Smažený krůtí řízek v sýrovém těstíčku, brambory maštěné, obloha, nápoj STUDENTSKÝ TIP: 100g Smažená kuřecí prsa v sýrovém těstíčku, brambory maštěné, nápoj STUDENTSKÝ TIP: 100g Smažený kuřecí řízek v pivním těstíčku, brambory maštěné, nápoj
100g Přírodní kuřecí řízek, rýže, nápoj 100g Přírodní kuřecí plátek, rýže, nápoj 100g Přírodní kuřecí plátek, brambory, mrkev na másle, nápoj
100g Smažená vepř. kotleta, brambory maštěné, nápoj 100g Smažený vepř. řízek, bramb. kaše, nápoj A 1, 3, 7
STUDENTSKÝ TIP: 120g Hamburská vepř. kýta, houskový knedlík, nápoj 100g Hamburská vepř. kýta (smetana, uzenina, steril. okurky), houskový knedlík, nápoj
240g Smažené kuřecí stehno, bramb. kaše, nápoj 220g Uzené kuřecí stehno, bramb. kaše, nápoj STUDENTSKÝ TIP: 100g Kuřecí závitek na smetaně, houskový knedlík, nápoj STUDENTSKÝ TIP: 300g Kuřecí špalíčky v medové marinádě, rýže, nápoj BEZLEP 240g Kuřecí stehno na medu, rýže, nápoj 240g Kuřecí stehno na kari (smetana), těstoviny penne, nápoj 240g Pečené kuřecí stehno, rýže, nápoj 240g Kuřecí stehno v medové marinádě, rýže, nápoj
⋮

B Testovací rozhraní

V této příloze popíši testovací rozhraní, které bylo implementováno pro snadnější testování jednotlivých konfigurací. Toto testovací rozhraní je možné spustit příkazem spuštěným z adresáře `/bin` na přiloženém CD (příloha D):

```
$ java -jar MealRecommender.jar VSTUPNI_CESTA VYSTUPNI_CESTA
```

kde argument `VSTUPNI_CESTA` značí cestu k adresáři, ve kterém se nachází databázové soubory (na CD je cesta k adresáři `/data/`) a `VYSTUPNI_CESTA` značí cestu k adresáři, kde budou umístěny dočasné soubory.

Pokud nejsou předány argumenty, potom je jako argument `VSTUPNI_CESTA` použitý adresář `../data/` (`..\data\` v případě OS Windows) a jako argument `VYSTUPNI_CESTA` použita cesta `./` (`.\` v případě OS Windows). K cestě předané v argumentu `VYSTUPNI_CESTA` musí mít uživatel právo zapisovat. Z toho důvodu nelze spustit program bez argumentů přímo z CD.

Po úspěšné inicializaci databáze je zobrazena hlavní nabídka:

```
// MealRecommender test interface //

Active user: 0
Active method: Collaborative Base

1) make order by active user
2) show orders for active user
3) add meal to today menu
4) generate random today menu
5) show today menu
6) show all meals in database
7) show all users in database
8) show orders for user id
9) change user
10) change method

R) recommend meal to active user
E) evaluate against test db
Q) quit

$
```

V tuto chvíli je aktivní uživatel `0`. To je takový uživatel, kterému doporučovací systém sestavuje doporučení. Uživatel s identifikátorem `0` se od ostatních odlišuje skutečností, že zpočátku nemá žádnou historii objednávek.

Následně je možné využít jednu z voleb:

- `1)` pro vytvoření objednávky aktivním uživatelem,
- `2)` pro zobrazení objednávek aktivního uživatele,
- `3)` pro přidání jídla do denní nabídky,
- `4)` pro vygenerování denní nabídky čítající náhodné čtyři jídla,

- `5)` pro zobrazení denní nabídky,
- `6)` pro zobrazení všech jídel v databázi,
- `7)` pro zobrazení všech uživatelů v databázi,
- `8)` pro zobrazení objednávek určitého uživatele,
- `9)` pro změnu aktivního uživatele,
- `10)` pro změnu aktivní metody doporučování,
- `R)` pro sestavení doporučení aktivnímu uživateli pomocí aktivní metody,
- `E)` pro otestování aktivní metody oproti testovací databázi,
- `Q)` pro ukončení programu

Po vybrání jedné z voleb určené identifikátorem `1)`, `3)`, `8)`, `9)` a `10)`, program požádá o vložení dalších informací. V případě volby určené identifikátorem `10)` je to výběr jedné z konfigurací, které byly popsány v kapitole 4.

C Otestování v reálné aplikaci

V zadání této diplomové práce stojí:

> *„Bylo by vhodné, kdyby daný algoritmus byl otestován v reálné aplikaci a do práce byla zahrnuta analýza toho, jak přidání doporučovacího systému ovlivnilo chování uživatelů.“*

V příloze je popsán tento požadavek a vysvětleno proč nebyl naplněn.

Výběrem oblasti doporučování jídel v univerzitní menze byl tento bod zkomplikován. Informační systém, který využívá Správa kolejí a menz Univerzity Palackého (dále jen SKM UP), je kritickou částí IT infrastruktury, kterou denně využívají tisíce lidí. Z toho důvodu je takřka nemožné používat tento informační systém k experimentům.

O práci na implementaci doporučovacího systému v oblasti jídel bylo kontaktováno vedení SKM UP, ale i brněnská společnost ANETE spol. s r.o. (dále jen ANETE), která univerzitě informační systém dodává. Obě strany tuto práci přivítaly a právě díky setkání se zástupcem SKM UP mi byla poskytnuta data pro testování. Výsledky práce považuji pro pro použití v praxi za zajímavé a o nasazení alespoň myšlenek vzešlých z výsledků práce budu usilovat.

Se SKM UP i ANETE jsem v kontaktu od zpracování mobilní aplikace pro objednávání jídel v rámci bakalářské práce „*Aplikace pro objednávání jídel pro platformu Android*“. Tato aplikace je funkční a na pěti českých vysokých školách ji využívají tisíce lidí. I díky možnosti vylepšit tuto aplikaci mám zájem o nasazení určité formy doporučovacího systému.

Využití doporučovacího systému považuji za užitečné právě při použití mobilní aplikace pro objednávání jídel. Aplikace vyvinuté pro platformu Android dokáží uživatele upozorňovat i v případě minimalizace aplikace v pozadí. Toho by bylo možné využít v poledních hodinách na upozornění, že *dnes má menza v nabídce jídlo, které by uživateli mohlo chutnat*. Dovolím si odhadnout, že takové upozornění by velké množství uživatelů ocenilo.

Pracnost implementace doporučovacího systému se liší dle použité metody. Pokud by byl použitý doporučovací systém se znalostí uživatelů popsaný v kapitole 3.6, bylo by nutné implementovat podporu doporučování na serverech SKM UP a následně se tohoto serveru dotazovat. Jednodušší řešení by však mohlo spočívat pouze v implementaci doporučovacího systému se znalostí obsahu popsaného v kapitole 3.5 na straně aplikace. Aplikace by mohla shromažďovat klíčová slova jídel, která si uživatel objednal, a pouze s pomocí těchto slov by mohla doporučovat vhodné jídlo z aktuální nabídky.

D Obsah přiloženého CD

Součástí diplomové práce je CD s následující strukturou.

bin/
: Adresář obsahuje Java archiv `MealRecommender.jar`, který je spustitelnou formou testovacího programu `MealRecommender`.

doc/
: Adresář obsahuje text diplomové práce ve formátu PDF a ZIP archiv `kidiplom.zip`, jenž obsahuje zdrojové soubory, které jsou potřeba k přeložení dokumentu.

src/
: Adresář obsahuje zdrojové kódy programu `MealRecommender` společně s knihovnami potřebnými pro jeho kompilaci.

data/
: Adresář obsahuje soubory `input1.csv` a `input2.csv`, které obsahují data o objednávkách. Soubory poskytl zástupce vedení Správy kolejí a menz Univerzity Palackého. SKM UP si však nepřálo data zveřejňovat, proto mají přiložené soubory promíchána jména jídel. Z tohoto důvodu se mohou výsledky mírně lišit.

install/
: Adresář obsahuje instalátory běhového prostředí `Java Runtime Environment 8` pro různé platformy. pro správný běh programu `MealRecommender` je požadováno běhové prostředí verze 8 a výše.

readme.txt
: Textový soubor obsahující informace ke spuštění programu `MealRecommender` a k přeložení tohoto dokumentu.

Literatura

[1] Resnick, Paul; Varian, Hal R. Recommender Systems. *Commun. ACM*. 1997, roč. 40, č. 3, s. 56–58. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=245121⟩. ISSN 0001-0782.

[2] Jannach, Dietmar; Zanker, Markus; Felfernig, Alexander; Friedrich, Gerhard. *Recommender Systems: An Introduction*. 1st. New York, NY, USA: Cambridge University Press, 2010. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=1941904⟩. ISBN 0521493366, 9780521493369.

[3] Chakrabarti, Soumen. *Mining the Web: Discovering Knowledge from HyperText Data*. 2002. ISBN 1558607544.

[4] Goldberg, David; Nichols, David; Oki, Brian M.; Terry, Douglas. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM*. 1992, roč. 35, č. 12, s. 61–70. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=138867⟩. ISSN 0001-0782.

[5] Resnick, Paul; Iacovou, Neophytos; Suchak, Mitesh; Bergstrom, Peter; Riedl, John. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. Chapel Hill, North Carolina, USA: ACM, 1994, s. 175–186. CSCW '94. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=192905⟩. ISBN 0-89791-689-1.

[6] Schein, Andrew I.; Popescul, Alexandrin; Ungar, Lyle H.; Pennock, David M. Methods and Metrics for Cold-start Recommendations. In. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland: ACM, 2002, s. 253–260. SIGIR '02. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=564421⟩. ISBN 1-58113-561-0.

[7] Herlocker, Jonathan L.; Konstan, Joseph A.; Borchers, Al; Riedl, John. An Algorithmic Framework for Performing Collaborative Filtering. In. *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Berkeley, California, USA: ACM, 1999, s. 230–237. SIGIR '99. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=312682⟩. ISBN 1-58113-096-1.

[8] Levenshtein, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*. 1966, roč. 10, s. 707. Dostupný také z: ⟨http://www.scribd.com/doc/105599835/Binary-codes-capable-of-correcting-deletions-insertions-and-reversals#scribd⟩.

[9] Dice, Lee R. Measures of the Amount of Ecologic Association Between Species. *Ecology*. 1945, roč. 26, č. 3, s. pages. Dostupný také z: ⟨http://www.jstor.org/stable/1932409?seq=1⟩. ISSN 00129658.

[10] Bělohlávek, Radim. Introduction to Formal Concept Analysis. 2008. Dostupný také z: ⟨http://belohlavek.inf.upol.cz/vyuka/IntroFCA.pdf⟩.

[11] Wille, Rudolf. Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In Rival, Ivan (ed.). *Ordered Sets*. 1982, s. 445–470. NATO Advanced Study Institutes Series. Dostupný také z: ⟨http://dx.doi.org/10.1007/978-94-009-7798-3_15⟩. ISBN 978-94-009-7800-3.

[12] Boucher-Ryan, Patrick du; Bridge, Derek G. Collaborative Recommending using Formal Concept Analysis. *Knowl.-Based Syst.* 2006, roč. 19, č. 5, s. 309–315. Dostupný také z: ⟨http://dx.doi.org/10.1016/j.knosys.2005.11.017⟩.

[13] Krajča, Petr; Outrata, Jan; Vychodil, Vilém. Parallel recursive algorithm for FCA. In. *Proceedings of the 6th International Conference on Concept Lattices and Their Applications (CLA)*. 2008, s. 71–82.

[14] Kuznetsov, Sergei O. Learning of Simple Conceptual Graphs from Positive and Negative Examples. In. *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK, UK: Springer-Verlag, 1999, s. 384–391. PKDD '99. Dostupný také z: ⟨http://dl.acm.org/citation.cfm?id=645803.669357⟩. ISBN 3-540-66490-4.