*Zkrácená verze – pouze popis doporučovacích algoritmů*

- *Upraveno L. Peška*
- *Publikováno se souhlasem autorky*

Maria Kukhar

# Content-based doporučovací systémy

Katedra softwarového inženýrství

Vedoucí diplomové práce: Mgr. Peška Ladislav

Studijní program: Informatika
Studijní obor: softwarové systémy

## *1.1 Přehled existujících doporučovacích technik a systémů*

Existují dvě základní techniky doporučování [1], podle kterých lze klasifikovat doporučovací systémy:

- **Kolaborativní filtrování** (Collaborative filtering)
- **Doporučení založené na obsahu** (Content-based filtering)

Systémy které spojují v sebe tyhle dvě techniky nazývají hybridní.

### 1.1.1 Kolaborativní filtrování

Systémy založené na **kolaborativním filtrování** doporučují položky na základě podobnosti uživatelů. Uživatelovi bude doporučena ta položka, která se líbila podobnému uživatelovi.

Pokud chceme dozvědět, na kolik uživatel preferuje nějakou položku, jeden ze způsobu to dozvědět je podívat se na kolik ostatní podobné uživatele preferují tuhle položku. Jeden z nejznámějších algoritmu kolaborativního filtrování založeného na *podobnosti uživatelů* je **Neighborhood-based algoritm,** který skládá z následujících kroku [16]:

1. Výpočet podobnosti mezi uživateli, pro který se nejčastěji používá Pearsonová korelace (6) nebo kosinová míra (7).

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}}, \quad (6)$$

kde $i \in I$ položky které ohodnotil jak uživatel *u* tak i uživatel *v*, $\bar{r}_u, \bar{r}_v$ - průměrné ratingy uživatelů *u* a *v*.

$$w_{u,v} = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| * \|\vec{v}\|}, \quad (7)$$

kde $\vec{u}, \vec{v}$ jsou řádky z matici uživatel - položka (viz. Chyba! Záložka není definována.) predstavuící uživatelů *u* a *v.*

2. Získání předpovědi nebo doporučení pomocí váženého součtu sousedských hodnocení. Předpověď pro uživatele *a* a položku, *i* se vypočítává podle vzorce (8):

[1] http://en.wikipedia.org/wiki/Recommender_system

$$P_{a,i} = \bar{r_a} + \frac{\sum_{u \in U}(r_{u,i} - \bar{r_u}) \cdot w_{a,u}}{\sum_{u \in U}|w_{a,u}|}, \quad (8)$$

kde, $\bar{r_a}, \bar{r_u}$- pruměrné ratingy pro uživateli *a* a *u*, $w_{a,u}$- vaha jejích podobnosti.

3. Uživateli vrací Top-N doporučení - N položek s největším podvedeným ratingem.

*Podobnost položek* je jiná možnost zjištění uživatelských preferencí pro nějakou položku. V tomto případě třeba najít podobné jí položky a najít uživatelské preference ohledně těch podobných položek. Podobnost položek se také vypočítává na základě hodnocení - pokud hodně uživatelů ohodnotili dvě položky stejně, pak ty dvě položky jsou podobné. Výhoda toho přístupu je v tom že podobnost položek se nemění s časem, ale podobnost uživatelů může. Ale v případě velkého množství položek a malého množství uživatelů přístup je méně užitečný než podobnost uživatelů [17].

Podobnost položek lze vypočítat stejným způsobem, jako podobnost uživatelů podle Pearsonové korelaci (6) nebo kosinové míry (7), s tím, že místo uživatelů *u*,*v*, budou zpracovávaní položky a jejích vektory. Pro předpověď ratingu, v případě doporučovaní založeného na *podobnosti položek*, lze použit jednoduchý vážený průměr a předpověď pro uživatele *u* a položku *i*, získat podle (9)

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} \cdot w_{i,u}}{\sum_{n \in N}|w_{i,u}|}, \quad (9)$$

kde součet je podle všech hodnocených položek, pro uživatele *u*, $w_{i,u}$ je váha podobnosti mezi položkami *i* a *n*, $r_{u,n}$je rating uživatele *u* pro položku *n* [16].

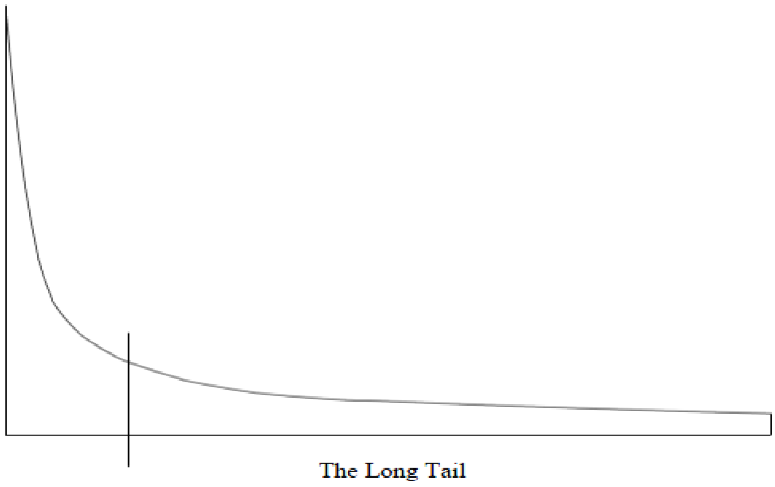
*Faktorizace matic* [18] je další populární technika kolaborativního filtrování. Na rozdíl od předchozích přístupů, kde se doporučení zakládalo na podobnosti uživatelů, nebo položek, tento přístup předpokládá, že podobnosti mezi uživateli a položkami jsou vyvolávány nějakou skrytou nízko-dimenzionální strukturou v datech [19].

Hlavní prioritou kolaborativního filtrování je to, že dokáže poskytovat doporučení a i když neposkynuta žádná kontextuální informace o položce. Při použiti kolaborativní techniky však může dojít k následujícím **problémům**:

- *Studený start (cold start)*. Problém se vyskytuje při přidaní do databáze nových

uživatelů nebo nových položek. Pro tvorbu spolehlivých doporučení systém potřebuje aby uživatel ohodnotil dostatečný počet položek. Analogicky, aby položka se dostala do processu doporučení třeba, aby ji ohodnotilo potřebné množství uživatelů.

- *Problém řídkosti (sparsity problem).* Nejaktivnější uživatelé hodnotí pouze malou podmnožinu všech položek v databáze. Proto, i nejpopulárnější položky mají velmi málo hodnocení. Kvůli tomu data jsou řídká a nedostatečná pro identifikaci sousedy co omezuje kvalitu doporučení.
- *Problém škálovatelnosti (scalability problem).* Vznika s růstem počtu uživatelů a položek v systému. Pokud třeba spočítat doporučení pro miliony uživatelů a objektů, systém potřebuje velký výpočetní výkon. V případě že systém musí okamžitě reagovat na on-line dotazy od všech uživatelů, pak vyžadovaná ještě větší škálovatelnost.
- *Long tail*. Jednou z cíli doporučovacího systému je pomoct uživatelovi objevit nové produkty. Fyzicky doporučovací systém může ukázat zákazníkovi pouze malé množství všech položek, které existují a obvykle to jsou nejpopulárnější položky, které ohodnotilo mnoho uživatelů. Tím pádem položky které ohodnotilo jen málo uživatelů nebudou doporučeny. Množstvo těchto položek představuje je tzv. «long tail». Takže uživatel může přijít o doporučení zajímavých položek které nejsou populární. Na Obrázek 12 zobrazen «long tail». Svislá osa představuje popularitu (kolikrát položka je ohodnocena). Položky jsou seřazeny na vodorovné ose v závislosti na jejich popularitě. System doporučí pouze nejpopulárnější položky vlevo od svislé čáry [9].




*Obrázek 4: Long tail*

Rozsáhlý přehled problemů kolaborativního filtrování mají X. Su, T.M. Khoshgoftaar v své práce [24].

### 1.1.2 Doporučení založené na obsahu

**Doporučení založené na obsahu** funguje na principu, kde uživatel bude preferovat podobné věci jako preferoval v minulosti a proto content-based systém doporučuje uživatelovi položky podobné těm, které v minulosti dostali od uživatele vysoké hodnocení. Každá položka představená sadou vlastností. Systém analyzuje položky dříve hodnocené uživatelem a na základě vlastností těch položek vytváří profil zájmů uživatele (dale jen **profil uživatele**). Cílem doporučovacího systému je najít mezi položkami ty, které uživatel neohodnotil a vlastnosti kterých se nejvíc odpovídají vlastnostem profilu uživatele. Proces doporučení spočívá v tom, že atributy v profilu uživatele se porovnávají s atributy položky. Výsledek představuje úroveň zájmů uživatele o tuto položku. [3] ,[20]. Popis obecné architektury a určitých algoritmů doporučovacích systémů založených na obsahu obsahuje kapitola 7.

V [20], [5, 7] uvěděný následující výhody a nedostatky doporučení založeného na obsahu.

**Výhody doporučení založeného na obsahu:**

- *Nezávislost uživatele na ostatních uživatelích.* Doporučení založeno jenom na vlastním profilu uživatele a jeho hodnoceních, proto nepotřebuje hodnocení od jiných uživatelů aby zjistit podobnost, jak to funguje v kolaborativním

filtrování

- *Průhlednost celé metody.* Může byt poskytnuto průhledné vysvětlení toho na základě čeho byla doporučena konkretní položka. Pomocí toho uživatel může rozhodnut jestli má věřit doporučovaní nebo ne. Kolaborativní filtrování funguje jako černa skříňka a může poskytnut jenom vysvětleni tipu: uživateli s podobným vkusem se to líbilo proto položka byla doporučena
- *Doporučení nových a nepopulárních položek.* Možnost doporučovat položky které zatím nebyli ohodnoceny žádným uživatelem
- *Doporučení pro uživatele s jedinečným vkusem.*
- *Neexistuje problémy řídkosti (sparsity problem).*

**Nedostatky doporučení založeného na obsahu:**

- *Informace popisující položku je omezena.* Systém nemůže poskytnout vhodné doporučení, je-li popis položky neobsahuje dostatek informací pro odlišení položek, které uživatel preferuje od těch, které nepreferuje.
- *Přílišná specializace doporučení.* Systémy produkují doporučení s omezeným stupněm novosti. Doporučené položky velmi podobné těm, které uživatel hodnotil v minulosti a proto uživatel nikdy nedostane nějaké nové neočekávané doporučení.
- *Problém nového uživatele.* Uživatel musí předem ohodnotit dostatečné množství položek, než systém bude schopen poskytnout spolehlivé doporučení.

### 1.1.3 Hybridní přístup

Cílem **hybridního systémy** je vylepšit doporučení pomoci odstranění nedostatků kolaborativního a contentent-based přístupu a hlavně těch největších: problemů řídkosti a studeného startu.

Jeden z příkladů hybridního systému popsan v [15] - Fab System, kde profily zájmů uživatelů založené na obsahové analýze primo se porovnávají pro stanovení podobnosti uživatelu a získání kolaborativních doporučení. Podobný princip využití uživatelského profilu (založeného na obsahu) k hledaní podobnosti uživatele má v své práci A. Eckhardt [17].

Jíny Příklad hybridního systémy popsán v [17]. Doporučovací systém pro Fastweb (jeden z nejnejvětších poskytovatelů IP televize v Evropě) implementuje obě technicky: kolaborativní a založenou na obsahu. Doporučovací systém volí techniku doporučení a algoritmus v zavilosti na kontextu. Například, v případě, že uživatel čte stručné shrnutí k filmům, hledá filmy s jeho oblíbenými herci apod., bude zvolené doporučení založené na obsahu.

Dalším druhem hybridních systémů jsou tak zvané Content-Boosted Collaborative Filtering systémy, které zapojují informací o obsahu do standardních algoritmu kolaborativního filtrování. Napřiklad, Forbes, Zhu v [18] předvádí zapojení informací o obsahu jako lineárního omezení do kolaborativní metody faktorizaci matice. Další příklady Content-Boosted kolaborativních systému uvedené v [19], [20].

Existují i další typy hybridních systémů. R. Burke v své prací [21] rozlišuje sedm různých druhu hybridních technik.

## 1.2 Principy budovaní doporučovacího systémů založeného na obsahu

Podle P. Lops a další [20] proces doporučení lze rozlišit na tři základní kroky :

1. **Analýza obsahu a reprezentace položky**. Vytvoření strukturovaného popisu položky z její vlastností. Většinou se jedná o nestrukturované data, jako například dokumenty, články, zprávy, popisy produktů a pod. Položka má obsahovat popis ve vhodné podobě pro další kroky zpracování. Přístupy pro uvedení strukturovaných a nestrukturovaných dat do takovéto podoby se liší. Strukturovaná data lze například uložit do databázové tabulky, kde jména sloupců jsou atributy popisující položku. Každá položka v tomto případě má obsahovat stejnou sadu atributů. V nestrukturovaných textech lze identifikovat slova, která charakterizují téma dokumentu. Proto je třeba na začátku vynechat z textu stop-slova - často vyskytující slova, které nenesou žádnou významovou informaci a mají zpravidla pouze syntaktický význam (spojky, předložky atp.). Pak spočítat TF-IDF značení (váhu slova v rámci dokumentu) pro každé slovo v dokumentu. Slova s největšími hodnotami TF-IDF jsou slova nejvíc charakterizují dokument a sada těchto z slov může reprezentovat položku [9].

2. **Budování profilu uživatele.** Profil uživatele je strukturovaná reprezentace zájmů uživatele, určená k doporučení nových položek. Pokud profil přesně odráží uživatelské preference, pak ho lze využít k filtrování položek, určení, zda uživatel má, nebo nemá zájem o konkrétní položku. Profil uživatele lze vytvořit na základě předchozích hodnocení položek uživatelem a jejich strukturovaném popisu. K problému vytvoření profilu uživatele lze přistupovat jako k problému strojového učení. O položkách oceněných uživatelem lze uvažovat jako o trénovací množině a pro každého uživatele vytvořit klasifikátor, který bude předpovídat hodnocení ostatních položek. [9]. Jednou z populárních metod pro vytvoření profilu uživatele jsou Rozhodovací stromy.

3. **Filtrace dat**. Doporučení relevantních položek na základě profilu uživatele. Na tomto kroku systém pomoci porovnání vlastností v reprezentaci položky s vlastnostmi uloženými v profilu uživatele předpovídá, zda je pravděpodobné, že položka bude zajímavá pro daného uživatele. Obvykle seznam doporučení obsahuje top-*n* potenciálně zajímavých položek, zařáděných podle jejích vhodnosti.

### 1.2.1 Reprezentace položky

Reprezentace položky může byt vytvořená několika způsoby, většinou způsob reprezentace závisí na typu dat které obsahují webové stránky a které bude zpracovávat doporučovací systém. Lze rozlišit tři typu dat:

- Strukturovaná
- Nestrukturovaná
- Polu-strukturovaná

#### *1.2.1.1 Strukturovaná data*

Obsah webové stránky (pokud reprezentuje zboží, film, knihu a pod.) je často uložen v databázové tabulce v strukturované podobě. Jako příklad uvažujme tabulku z knihami

| id | název | autor | cena | žánr | rok vydaní |
|---|---|---|---|---|---|
| 12579 | Zpověď | Gorkij M. | 100 | Beletrie | 1911 |

| | | | | | |
|---|---|---|---|---|---|
| 2576 | Dívka a smrt | Gorkij M. | 150 | Poesie | 1962 |
| 7918 | Ministerstvo strachu | Greene G. | 100 | Beletrie | 1964 |
| 12080 | Bludný kámen | Hanzlík J. | 80 | Poesie | 1962 |
| 11949 | Unesen | Stevenson R. L. | 80 | Dobrodružné | 1926 |

*Tabulka 1: Databázová tabulka s knihy*

Tabulka popisuje 5 knih, každá reprezentovaná jedním řádkem. Názvy sloupců jsou **atributy** těch knih. Takže, v daném příkladě položky (knihy) jsou reprezentovány sadou **atributů:** název, autor, cena, žánr, rok vydaní. Každý záznam tabulky obsahuje hodnotu pro každý atribut. Jedinečný identifikátor, **id** umožňuje ukládaní položek se stejným názvem.

Data popsáné v tomto příkladě jsou **strukturovaná**. V [20] pojem strukturovaných dat definován následovně: pokud je každá položka popsána stejnou sadu atributů a množina hodnot těch atributů je známá, pak položka reprezentovaná prostřednictvím strukturovaných dat.

Pro strukturovaná data lze využit hodně algoritmů strojového učení aby naučit profil uživatele. Například, profil lze naučit pomocí *rozhodovacích stromů* (viz. 13) nebo pomocí *metodů nejbližších sousedů* s použitím *Euklidovské metriky vzdálenosti*. Více metodů uvedeno v 13.

#### *1.2.1.2 Nestrukturovaná data*

Pokud webové stránky obsahují dokumenty, články, zprávy a jiný volny text, to jsou data nestrukturované. Na rozdíl od strukturovaných dat oni neobsahují atributy z dobré definovanými hodnotami. Navíc v přirozeném jazyce jsou takové komplikace jako například mnohoznačný slova a synonymy.

**Model vektorového prostoru založený na klíčových slovech (Keyword-based Vector Space Model)** dále **VSM** je základním přístupem pro reprezentaci nestrukturovaných dokumentů.

Podle tohoto modelů, každý dokument představuje vektor s vahami slov, kde každá váha označuje míru asociace mezi dokumentem a slovem.

Nechť $D = \{d_1, d_2, .., d_N\}$ označuje sadu dokumentů a $S = \{s_1, s_2, .., s_n\}$ je slovnik,

nebo sada slov z dokumentů. S je získané pomoci použití technik z Information Retrieval [9] pro zpracování přirozeného jazyka (tokenizace, odstranění stop-slov,...). Každý dokument $d_j$ reprezentován jako vektor v *n*-dimenzionálním vektorovem prostoru, kde$w_{kj}$je váha slova$s_k$v dokumentu $d_j$. Dokument reprezentovaný pomoci VSM vyvolává dvě otázky:

- vážení slov
- měření podobnosti vektoru vlastnostej

Nejpopularnejší schema na vážení slov je TF-IDF (Term Frequency-Inverse Document Frequency).

TF-IDF funkce:

$$TF - IDF(s_k, d_j) = TF(s_k, d_j) \cdot \log \frac{N}{n_k}$$

Kde *N* je to celkový počet dokumentů a $n_k$je to počet dokumentů v kolekce ve kterých slovo $s_k$vyskytlo alespoň jednou.

$$TF(s_k, d_j) = \frac{f_{k,j}}{max_z f_{z,j}}$$

kde je maximum vypočítán nad frekvencí $f_{z,j}$ vsech slov $s_z$ které vyskytují v dokumentu $d_j$. Aby váhy patřili do intervalu [0,1] a dokumenty byli reprezentovany pomoci vektorů stejné delky, váhy spočítané pomocí *TF-IDF* treba normalizovat:

$$w_{k,j} = \frac{TF - IDF(s_k, d_j)}{\sqrt{\sum_{t=1}^{|S|} \quad TF - IDF(s_t, d_j)^2}}$$

K zjištění podobnosti mezi dokumenty (vektory) nejčactejí plouživá kosinová podobnost:

$$simil(d_i, d_j) = \frac{\sum_k \quad w_{ki} \cdot w_{kj}}{\sqrt{\sum_k \quad w_{ki}^2} \cdot \sqrt{\sum_k \quad w_{kj}^2}}$$

V doporučení založeném na obsahu a využivajícím VSM, položka a profil uživatele reprezentované váženými vektory. Předpovědi zájmu uživatele o určitou položku lze odvodit výpočtem kosinové podobnosti [20].

#### 1.2.1.3 Polustrukturovaná data

Hodně domén nejlépe reprezentované pomoci polustrukturovaných dat, kde některé atributy mají množinu omezených hodnot a některé obsahují volný text. V tomto pripadě třeba převést volný text do strukturované podoby jako to popsáno v 2.2.1.2.

Další způsob převést volný text do strukturované podoby je vnímat každé slovo jako atribut a přiřadit jemu boolevská hodnotu, označující, zda slovo je v článku nebo integer hodnotu udávající, kolikrát se slovo zobrazí v článku.

### 1.2.2 Budování profilu uživatele

Budování *profilu uživatele* probíhá na základě předchozích hodnocení *položek* uživatelem. Hodnocení může byt *implicitním* nebo *explicitním*. Jedním z častých způsobu budovaní *profilu uživatel*e je použití technik strojového učení. Budování *profilu uživatele* lze považovat za *problém klasifikace*[2].

#### 1.2.2.1 Problém klasifikace a trénovací množina

*Problém klasifikace* spočívá v tom, že existuje konečná množina objektů, u nichž je známo, k jaké třídě patří - *trénovací množina.* Dále existuje množina objektů u nichž není známo, k jaké třídě patří. Pak pomocí *klasifikátoru*, naučeného na *trénovací množině*, je třeba zjistit, do jaké třídy patří libovolný objekt z neklasifikované množiny.

V případe budování *uživatelského profilu*, se *trénovací množina* skládá z *položek* oceněných uživatelem, kde každá položka je tak zvaná *instance trénovací množiny. Klasifikátor*, natrénovaný na této množině, je *profilem uživatele*.

Pro strukturovaná data *trénovací množina* může vypadat jako například v Tabulce 2.

| autor | cena | žánr | rok vydaní | hodnocení |
|---|---|---|---|---|
| Gorkij M. | 100 | Beletrie | 1911 | **0** |
| Gorkij M. | 150 | Poesie | 1962 | **1** |
| Greene G. | 100 | Beletrie | 1964 | **0** |
| Hanzlík J. | 80 | Poesie | 1962 | **1** |

[2]http://en.wikipedia.org/wiki/Statistical_classification

| | | | | |
|---|---|---|---|---|
| Stevenson R. L. | 80 | Dobrodružné | 1926 | **1** |

*Tabulka 2: Testovací množina z databáze knih*

Tabulka obsahuje *položky*, reprezentované sadou značení atributů (které podle předpokladů mají vliv na hodnocení) a také hodnocení těchto *položek*. V daném případě hodnocení nabývá dvě hodnoty *{0, 1}* co odpovídá značení „líbí se“/“nelíbí se”. V rámci problému klasifikace je *C = {0, 1}* množina tříd. Každá *položka* musí odpovídat právě jedné třídě ze sady *C*. Jeden řádek tabulky odpovídá jedné *instance trénovací množiny*

V případě nestrukturovaných dat je třeba aby data nejprv byli převedeny do strukturované podoby výběrem malé podmnožiny klíčových slov (reprezentujících *položky)* jako atributů. Značení těch atributů buď *TF-IDF* váha nebo boolevská hodnota (zda položka obsahuje toto slovo nebo ne). Příklad toho jak vypadá *testovací množina* v případě nestrukturovaných dat uveden v [25] (viz. Obrázek 5).

| Doc-ID | recommender | intelligent | learning | school | **Label** |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | **1** |
| 2 | 0 | 0 | 1 | 1 | **0** |
| 3 | 1 | 1 | 0 | 0 | **1** |
| 4 | 1 | 0 | 1 | 1 | **1** |
| 5 | 0 | 0 | 0 | 1 | **0** |
| 6 | 1 | 1 | 0 | 0 | **?** |

*Obrázek 5: Testovací množina a položka pro kterou třeba zjistit, do jaké třídy patři [25].*

Klíčova slova (recommender, intellegent, learning, school) v tomto příkladě považovaná za atributy, které nabývají hodnoty {0, 1} v závislosti na tom, zda se vyskytují v položkách s Doc-ID 1 až 6. Položky s Doc-ID 1 az 5 je *trénovací množina,* každá s těchto položek patří k jedné z tříd *C= {0, 1}.* Na základě těchto dat je třeba natrénovat *klasifikátor* (což je *profil uživatele*) a pak zjistit k jaké třídě patři položka s Doc-ID 6.

V uvedených přikladach zvolené dvě třídy pro jednoduchost. Ve skutečnosti množina může obsahovat více tříd. Například, pokud hodnocení *položky* probíhá ve tvaru hvězdiček, množina tříd budě obsahovat 5 tříd *C = {1,2,3,4,5}.*

### 1.2.2.2 Algoritmy strojového učení pro budování profilu uživatele.

Hodně různých algoritmů strojového učení jsou schopni naučit funkci (*klasifikátor*), která modeluje zájmy každého uživatele. Tyto algoritmy jsou klíčovou součástí *doporučovacích systémů založených na obsahu*. Naučená funkce, například, může odhadnout pravděpodobnost, toho jestli uživateli se líbí nějaká položka, kterou on zatím neohodnotil. Tuto pravděpodobnost lze využít k seřazení seznamu doporučení. Alternativně, algoritmus může vytvořit funkci, která se přímo předpovídá číselnou hodnotu, odpovídající stupni zájmu uživatele. Nejpoužívanější z algoritmy jsou:

- Rozhodovací stromy (decision trees) a rozhodovací pravídla (rule induction)
- Metody nejbližších sousedů (nearest neighbor methods)
- Zpětná vazba relevance (relevance feedback) a algoritmus Rocchio
- Lineární klasifikátory (linear classifiers)
- Pravděpodobnostní metody (probabilistic methods) a Naivní Bayes (Naïve Bayes)
- Neuronové sítě (neural networks)
- Bayesovské sítě (Bayesian network)

To jsou tradiční algoritmy strojového učení určené k práci na strukturovaných datech. V kapitolách 1.5.2.3 až 24popsáno budování profilu uživatele pomoci *rozhodovacích stromů.* S přehledem ostatních algoritmů se lze seznámit v [3, 4, 26].

*Rozhodovací stromy* jsou výkonným nástrojem pro řešení problému klasifikaci a můžou snadné představovat a učit profil uživatele. *Rozhodovací stromy* byly značně studovány na strukturovaných datech a jsou nejvíc účinné v případě malého počtu strukturovaných atributů, což se projevuje v jejích výkonu, jednoduchosti a srozumitelnosti [3]. Existuje velké množství algoritmů *rozhodovacích stromů* (Hunts, CART, ID3, C4.5, SLIQ, SPRINT a další) popsáných především v literatuře strojového učení a aplikované statistiky.

### 1.2.2.3 Rozhodovací stromy

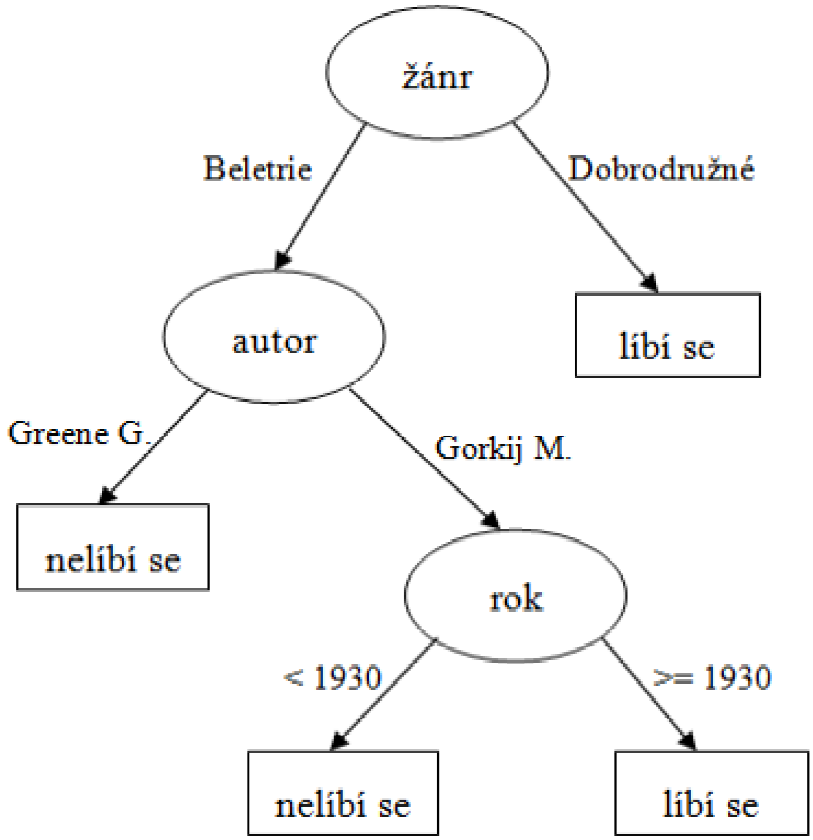
***Struktura rozhodovacího stromu***

*Rozhodovací strom* se skládá z vnitřních uzlů, hran a koncových uzlů.

- *Vnitřní uzel (uzel rozhodnutí)* představuje test na atributu nebo na podmnožině atributů. Každý uzel má právě jednu příchozí hranu. Uzel, který

nemá žádné příchozí hrany se nazývá *kořen*.

- *Hrana*, spojující uzly, označena určitou hodnotu nebo rozsahem hodnot atributů a představuje výsledek testu. Vnitřní uzly spolu s jejich hranami děli datovou množinu na dvě nebo více části.
- *Koncový uzel (list)*, terminální uzel stromu, který představuje třídu (rozhodnutí přijaté po výpočetní testu na všech atributech).




*Obrázek 6: Jednoduchý příklad rozhodovacího stromu.*

Cesta z *kořene* do *listu* představuje *rozhodovací pravidlo*.

Obrázek 6 zobrazuje jednoduchý příklad rozhodovacího stromu, kde pomoci kroužků označeny *uzly rozhodnuti a* pomoci čtverců *koncové uzly.* V tomto příkladu, máme tři atributy na základě kterých probíhá štípání {žánr, autor, rok} spolu se dvěma třídami {líbí se, nelíbí se}.

***Zobecněný algoritmus pro budování stromu***

Pro všechny atributy z trénovací množiny se používá funkce měření, cílem které je najít nejlepší atribut pro rozštěpení množiny. Po zjištění takového atributu, množina

se dělí na několik částí. V rámci každé části, pokud všechny trénovací instance patří k pravě jedné třídě, algoritmus končí. Jinak, proces štěpení bude pokračovat rekurzivně, dokud každá část nebude obsahovat jenom trénovací instancí ze stejné třídy. Po vybudovaní rozhodovacího stromu lze snadno vytvářet klasifikační pravidla, které mohou být použity pro klasifikaci nových instancí [27].

### *Algoritmy ID3, C4.5*

Dobře známými algoritmy pro generování rozhodovacích stromu založenymi na jednorozměrných rozdělích jsou *algoritmus ID3*[3] a jeho vylepšená verze *C4.5*[4] [29]. Tyhle algoritmy byli použity ve mnohých doporučovacích systémech, některé z nich popsány v [29, 30, 31, 32].

Aby pomocí *ID3* a *C4.5* vybudovat rozhodovací strom, vstupná data musí splňovat několik podmínek.

- Informace o objektech, které mají být klasifikované, by měla být prezentována ve formě konečné množiny hodnot atributů, kdě každý atribut má nominální nebo numerickou hodnotu. Množina hodnot atributu popisující jeden objekt se nazývá *instance*. Atributy použité k popisu *instance* se nesmí měnit případ od případu a jejích počet pro všechy *instance* musí být stejný.
- Množina tříd, podle kterých budou rozděleny instance by měla mít konečný počet prvků a každá instance by měla jasně odkazovat k určité třídě.
- Trénovací množina by měla obsahovat mnohem víc příkladů, než je počet tříd, navíc každá instance by měla být předem spojená s třídou.

### *Implementace*

Open source Java implementace algoritmu C4.5, zahrnuta v softwaru Weka[5] (balík programů strojového učení). Implementace v Weka se nazývá J48[6]

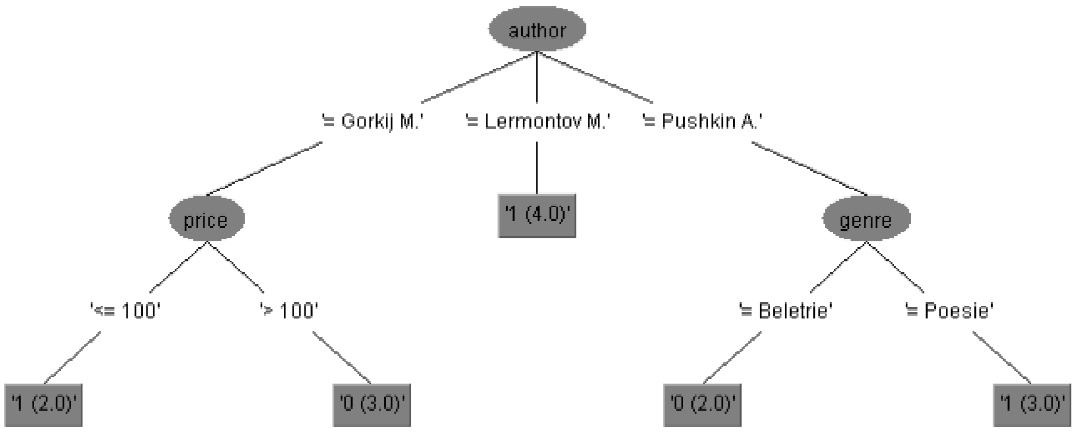
Obrázek 7 zobrazuje rozhodovací strom, vybudovaný pomocí nástroje Weka z trénovací množiny uvedené v tabulce 3. Podobná testovací množina může charakterizovat profil nějakého uživatele. Knihy, které ohodnotil uživatel popsané pomocí atributů {author, cena, žánr}. Předpokládá se že na těchto atributech záleží hodnocení.

[3] http://en.wikipedia.org/wiki/ID3_algorithm
[4] http://en.wikipedia.org/wiki/C4.5_algorithm
[5] http://www.cs.waikato.ac.nz/~ml/weka/
[6] http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

| autor | cena | žánr | hodnocení |
|---|---|---|---|
| Gorkij M. | 100 | Beletrie | 1 |
| Gorkij M. | 200 | Beletrie | 0 |
| Gorkij M. | 150 | Poesie | 0 |
| Gorkij M. | 200 | Poesie | 0 |
| Gorkij M. | 100 | Poesie | 1 |
| Lermontov M. | 100 | Beletrie | 1 |
| Lermontov M. | 150 | Poesie | 1 |
| Lermontov M. | 70 | Beletrie | 1 |
| Lermontov M. | 75 | Poesie | 1 |
| Pushkin A. | 150 | Beletrie | 0 |
| Pushkin A. | 70 | Beletrie | 0 |
| Pushkin A. | 150 | Poesie | 1 |
| Pushkin A. | 150 | Poesie | 1 |
| Pushkin A. | 200 | Poesie | 1 |

*Tabulka 3: Treovací množina Knihy*

Rozhodovací strom na Obrázek 7 tedy zobrazuje uživatelsky profil, podle kterého lze předpovědět hodnocení jíních knih.




*Obrázek 7: Rozhodovací strom vytvoreny z trenovací množiny uvedene v tabulce 3*

***Generování rozhodovacích pravidel***

Aby model rozhodovacího stromy byla čitelnější. Cesta ke každému listů může být proměněna na IF – THEN *rozhodovací pravidlo*.

Pro rozhodovací strom na obrázku 7 odpovídající *rozhodovací pravidla* budou vpadat nasledovaně:

*If* author = Gorkij M. *and* rice < = 100
*Then* Classification = 1 (2.0/0);

*If* author = Gorkij M. *and* price > 100
*Then* Classification = 0 (3.0/0);

*If* author = Lermontov M.
*Then* Classification = 1 (4.0/0);

*If* author = Puchkin A. *and* genre = Beletrie
*Then* Classification = 0 (2.0/0);

*If* author = Puchkin A. *and* genre = Poesie
*Then* Classification = 1 (3.0/0).

Každé rozhodnutí obsahuje dva parametry ve formě $(|T_i|/E)$, kde $|T_i|$ je počet instancí z *testovací množiny*, které dosáhnou list a E je počet *instancí*, které patří do jiných tříd než označena v listu.

## 1.3 Hodnocení doporučovacího systémů

Navrhnuty doporučovací systém lze považovat za úspěšný, pokud splňuje kladené na něho požadavky a to především schopnost předpovědět výběr uživatele. Také ve mnohých případech důležité objevování nových položek, rychlá odezva, zachování soukromí uživatelů a další vlastností [33].

Jedním ze způsobu zjištění úspěšnosti systémy je ověření jeho funkčnosti pomocí praktických experimentů. Uživatelské odezvy získané v rámci experimentu lze pak použit k vyhodnocení systému a stanovení jeho silných a slabých stránek. Shani G., Gunawardana V. v [33] uvádí tři druhy experimentů, které popsané v kapitole 18.

Také pro zjištění efektivity doporučovacího systému a jeho algoritmů lze použit některé metriky hodnocení. To je dobrý způsob pro porovnání efektivity několika algoritmů (více v 20).

### 1.3.1 Druhy experimentů

Existují tři druhy experimentů: offline, uživatelská studie a online experimenty.

#### 1.3.1.1 Offline experimenty

Offline experiment se provádí pomocí předem shromážděných údajů o uživatelských preferencích, to můžou být, například, ratingy poskytnuté uživatelem pro položky. Pomocí této sady dat se lze pokusit simulovat chování uživatelů, které interagují s doporučovacím systémem. Předpokládáme, že chování uživatelů, v době shromáždění údajů bude dost podobné uživatelskému chování po nasazení doporučovacího systému, proto na základě těchto dat můžeme učinit spolehlivé rozhodnutí.

Předpokládejme že máme sadu doporučovacích algoritmů z nichž je třeba zjistit jaké jsou nejvhodnější pro danou doménu. Cílem offline experimentů je odfiltrovat nevhodné doporučovací algoritmy a nechat jenom male množství nejefektivnějších, které lze pak testovat pomocí uživatelské studie nebo online experimentu.

Aby ohodnotit algoritmus offline je třeba simulovat online proces, kde systém vrací předpovědi či doporučení a uživatel opravuje tyhle předpovědi nebo používá doporučení. To se obvykle provádí nahráváním historii uživatelských dat a pak se skrývá některé z těchto interakcí, aby simulovat znalosti o tom, jak uživatel bude hodnotit položku, nebo s jakým doporučením se uživatel bude jednat.

Existuje řada způsobů, jak si vybrat položky, které mají být skryty. Dále uvedeny některé z nich:

- Vybrat experimentální množinu uživatelů, vybrat časové razítko a skrýt všechny položky oceněné po tom časovém razítku pro všech uživatelů.
- Vybrat časové razítko zvláštní pro každého uživatele a skrýt položky oceněné po tom časovém razítku u každého uživatele.
- Vybrat experimentální množinu uživatelů, vybrat počet položek pro skrýti pro každého uživatele, vybrat množinu položek pro skrýti odpovídající předem vybranému počtu.

Poslední způsob může byt použit v případě, že časová razítka z uživatelských akcí nejsou známy. Všechny tři způsoby rozdělí data do trénovací a testovací množiny. Pak systém se pokusí doporučit položky pro uživatele. Cílem je předpovědět kryté položky. Je důležité zvolit alternativu, která je nejvhodnější pro danou doménu.

Offline experimenty jsou atraktivní, protože nevyžadují interakce s reálnými uživateli a tím umožňují, například, porovnání efektivity široké sady doporučovacích algoritmů při nízkých nákladech.

Nevýhodou offline experimentů je, že můžou odpovědět na velmi úzký kruh otázek, to jsou většinou otázky se tykající přesnosti predikce. Takže nelze přímo měřit vliv doporučujícího systému na chování uživatelů, protože chování uživatelů bylo modelované před nasazením doporučovacího systému.

#### *1.3.1.2 Uživatelská studie*

Uživatelská studie se provádí pomocí zapojení testovacích osob, které jsou požadovaná splnit několik úkolů vyžadujících interakce s doporučovacím systémem. Zatím co testovací osoby plní úkoly, jejich chování se zaznamenává. Sbírá se libovolný počet kvantitativních měření, jako například: jaka část úkolu byla dokončena, přesnost výsledků, čas potřebný pro splnění úkolu apod. V mnoha případech testovacím osobám mohou byt položeny kvalitativní otázky před provedením úkolu, v jeho průběhu a po dokončení. Takové otázky mohou shromažďovat data, která nejsou přímo pozorovatelná jako například zda uživatelské rozhraní je pohodlné, nebo zda úkol je snadný pro splnění.

Obvykle uživatelská studie porovnává několik doporučovacích metod, kde každý

přístup musí být testovány za stejných úkolů. Metody mohou byt porovnané dvěma způsoby:

- mezi subjekty (*between subjects* nebo *A-B testing*)- každé testovací osobě přidělen jeden doporučovací metod, pro provádění experimentu nad nim.
- uvnitř subjektu (*within subjects*) - každá testovací osoba testuje sadu metod na různých úkolech.

Výhodami tohoto druhu experimentu na rozdíl od offline experimentu je to, že uživatelská studie umožňuje testovat chování uživatelů při interakci s doporučovacím systémem a vidět vliv doporučení na chování uživatele. Také uživatelská studie umožňuje shromažďovat kvalitativní údaje, které jsou často rozhodující pro interpretaci kvantitativních výsledků.

Hlavní nevýhodou uživatelských studií je to že jsou velmi nákladné na provádění (třeba najít velký počet testovacích osob pro plnění velkého počtu úkolu, navíc ty osoby by měli byt co nejblíž k realitním uživatelům systému).

#### *1.3.1.3 Online experimenty*

Online experiment se provádí pomocí reálných uživatelů, které plní skutečné úkoly. Online experiment je nejvíc důvěryhodný, z tohoto důvodu, mnoho skutečných světových systémy využívají online testování systému, kde mohou být porovnány více algoritmu. Typicky, tyto systémy provádí přesměrování male části uživatelů na jiný alternativní doporučovací systém. Tímhle způsobem lze získat záznamy uživatelských interakcí s různými systémy. Důležité vybírat uživatelů pro přesměrování náhodně, aby srovnání mezi alternativami bylo spravedlivé. V některých případech tyto experimenty jsou riskantní. Například, testovací systém, který poskytuje irelevantní doporučení, může odradit uživatele od použití skutečného systému. Z těchto důvodů má smysl na začátku provádět offline experiment nebo uživatelskou studii aby odfiltrovat nevhodné algoritmy.

### 1.3.2 Měření přesnosti předpovědi

V případě že doporučovací systém *předpovídá hodnocení položky* uživatelem je vhodné měření přesnosti tohoto hodnocení. Jedna z populárních metrik pro tento účel je *Root Mean Squared Error (RMSE).*

Systém generuje předpokládané hodnocení $\hat{r}_{ui}$ pro testovací množinu *T* uživatel-položka *(u, i),* pro které skutečné hodnocení $r_{ui}$jsou zname. Typicky, $r_{ui}$jsou známé,

protože jsou skryty v offline experimentu, nebo protože byly získány prostřednictvím uživatelských studií, nebo on-line experimentu. *RMSE* mezi předpokládanými a skutečnými hodnocení je dána vztahem:

$$RMSE = \sqrt{\frac{1}{T}\sum_{(u,i)\in T} (\hat{r}_{ui} - r_{ui})^2} \qquad (10)$$

Populární alternativou *RMSE* je *Mean Absolute Error (MAE)*:

$$MAE = \sqrt{\frac{1}{T}\sum_{(u,i)\in T} |\hat{r}_{ui} - r_{ui}|} \qquad (11)$$

Mnoho doporučovacích systémů přímo nepředpovídá hodnocení položky, ale snaží se *doporučit uživatelům položky, které se jím mohou líbit.*

Předpokládejme, že datová množina pro offline experiment se skládá z položek, které mají uživatelské hodnoceni. Část těch položek pro testových uživatelů byla skryta a systém požadován o doporučení. V tomto případě existuje čtyři možné výsledky pro doporučené a skryté položky (viz tabulka 4).

| | **Doporučena** | **Nedoporučena** |
|---|---|---|
| **Má hodnocení** | Pravdivě-pozitivní (*tp*) | Falešně-negativní (*fn*) |
| **Nemá hodnocení** | Falešně-pozitivní (*fp*) | Pravdivě-negativní (*tn*) |

*Tabulka 4: Klasifikace možných výsledků doporučení položky pro uživatele.*

Můžeme počítat počet položek, které spadají do každé buňky v tabulce a získáme následující hodnoty:

$$Precision = \frac{\#tp}{\#tp + \#fp} \qquad (12)$$

$$Recall(TruePositiveRate) = \frac{\#tp}{\#tp + \#fn} \qquad (13)$$

$$FalsePositiveRate(1 - Specificity) = \frac{\#fp}{\#fp + \#tn} \qquad (14)$$

V aplikacích, kde počet doporučení, které mohou být prezentovány uživateli je předem určen, nejužitečnější míra zájmu je *Precision * N*.

Pokud počet doporučení neurčen předem, je lepši, vyhodnocovat algoritmy v rozsahu délek seznamu doporučení, nežli používat pevnou délku. Třeba tedy počítat křivky porovnávající *Precision* a *Recall (precision-recall curves)* nebo *True Positive Rate* a *False Positive Rate (Receiver Operating Characteristic (ROC) curves).* Oba druhy křivek měří podíl preferovaných položek, které jsou ve skutečnosti doporučené, ale *precision-recall* křivky zdůrazňují podíl doporučených položek, které jsou preferované, zatímco *ROC* křivky zdůrazňují podíl položek, které nejsou preferované ale byli doporučeny.

Jakou z těch dvou druhu křivek zvolit pro měření závisí na doméně a cílí aplikace. Pokud dáne dva algoritmy, lze vypočítat dvojici křivek (*precision-recall* nebo *ROC*), jednu pro každý algoritmus. Pokud se jedna křivka zcela dominuje nad ostatními křivky, odpovídající jí algoritmus je lepší. Nicméně, když se křivky protínají, rozhodnutí o lepším algoritmu je méně zřejmé a bude záviset na konkrétní aplikaci.

Více o způsobech měřeni přesnosti doporučovacích algoritmů uvedeno v [33].

### 1.3.3 Algoritmy strojového ucení z WEKA

Weka je populární balík programů strojového učení napsaný v Javě, vyvinutý na University of Waikato, Nový Zéland. Weka je svobodný software dostupný podle GNU General Public License[7]. Obsahuje množství algoritmů pro analýzu dat a prediktivní modelování. Některé z algoritmu strojového učení z Weka byli implementování do navrhované doporučovací komponenty.

Princip prací algoritmů strojového učení spočívá v tom, že na základě zadaného množství ohodnocených dat (trénovací množiny) jsou schopny vybudovat klasifikátor, který pak lze použit k získání hodnocení pro nové data (testovací množinu). Podrobnější v 11.

Pro vytvoření doporučení pro uživatele $u_0$ všechny algoritmy strojového učení, použité v doporučovací komponentě, jako trenovací množinu dostávají položky s ratingem od uživatele $u_0$ buď explicitním, nebo získaným na základě implicitní zpětné vazby. Testovacích množina pro uživatele $u_0$ jsou všechny položky bez

[7]https://cs.wikipedia.org/wiki/Weka

hodnocení. Algoritmy se snaží na základě vybudovaného klasifikátoru předpovědět ratingy pro testovací množinu. Pak pro doporučení jsou vybrané položky s největšími předpověděnými ratingy.

Dále v teto kapitole uvedeny algoritmy použití v doporučovací komponente.

#### *1.3.3.1* J48

C4.5 (ve Weka se nazývá J48) je jedním z nejpopulárnějších algoritmů strojového učení založený na rozhodovacích stromech 13. *Algoritmus C4.5* je vylepšená verze *algoritmu ID3*, proto na začatku se seznámím s *algoritmem ID3.*

**Algoritmus ID3**

1. V prvním kroku máme strom který obsahuje pouze kořen a počáteční trenovací množinu *T* (spojenou s kořenem). Pokud *T* obsahuje instance, které patří k více než jediné třídě, třeba vybrat atribut pro rozdělení množiny.
2. Pro vyber atributu je třeba spočítat *informační zisk* pro každý atribut trenovací množiny *T*. Pak, pro rozdělení vybrat atribut, který má největší *informační zisk.* Pojem *informační zisk* založen na konceptu teorii informaci: *entropie.*

Nechť atribut *X* nabývá hodnoty $a_{1,}a_{2,}\ldots,a_n$Pokud množinu *T* rozdělit podle atributu *X* vzniknou podmnožiny $T_{1,}T_{2,}\ldots,T_n$.

Nechť $freq(C_i,S)$je počet instancí z některeže množiny S, které patří do stejné třídy $C_i$. $C_i$je jedna z možných tříd $C_{1,}C_{2,}\ldots,C_k$. Pak, pravděpodobnost toho, že náhodně vybraná instance z množiny *S* bude patřit do třídy s$C_i$se rovná $P = \frac{freq(C_i,S)}{|S|}$.

*Entropii* množiny *T* lze spočítat následujícím způsobem:

$$Info(T) = -\sum_{i=1}^{k}((freq\,(C_i,T)/|\,T\,|)\cdot \log_2(freq\,(C_i,T)/|\,T\,|)) \qquad (36)$$

*Entropie* po rozělení *T* podle atributu *X:*

$$Info_X(T) = \sum_{i=1}^{n}((|\,T_i\,|/|\,T\,|)\cdot Info(T_i)) \qquad (37)$$

*informační zisk* pro atribut *X*:

$$Gain(X) = Info(T) - Info_x(T) \qquad (38)$$

3. Po nalezení vhodného atributu pro rozdělení *T,* třeba vytvořit *uzel rozhodnutí* obsahující tento atribut. Hrany odchozí z tohoto uzlu budou obsahovat hodnoty toho atributu. Množina *T* pak bude rozdělena podle nich na podmnožiny.

4. Algoritmus pokračuje v recurse na každé podmnožině. Na každé iteraci algoritmus prochází všechny zatím nepoužité atributy množiny *T* a vypočítá *informační zisk* těchto atributů.

Rekurze na podmnožině může se zastavit v jednom z těchto případů:

- Každá instance v podmnožině patří do stejné třídy, pak uzel se promění v list s označením třídy.
- Nejsou k dispozici žádné další atributy, které budou vybrány pro rozdělení, ale všechny instanci v podmnožině stále nepatří do stejné třídy, pak uzel se promění v list s označením té třídy, ke které patří nejvíc instancí.
- Nejsou k dispozici žádné instance v podskupině, toto se stane, když v rodičovské množině nebyl nalezen žádný příklad, odpovídající konkrétní hodnotě vybraného atributu. Pak se vytvoří list s označením té třídy, ke které patří nejvíc instancí v rodičovské množině.

**Algoritmus C4.5**

Vylepšení *C4.5* zahrnují metody pro práci s číselnými atributy, chybějícími hodnotami, nevyčištěnými daty a umožňují generování pravidel ze stromu [21].

Budování rozhodovacího stromu pomocí *algoritmu C4.5* není zásadně odlišné od jeho budování pomocí *ID3*. Na rozdíl od *ID3*, při budování rozhodovacího stromu pomocí *C4.5*, povolené opakované použíti atributů. Každý z atributů, lze použít neomezeně. Proto pro zastavení *C4.5* platí jenom dva kriteria:

- Každá instance v podmnožině patří do stejné třídy.
- Nejsou k dispozici žádné instance v podskupině.

Kritérium *informační zisk* v *ID3,* který popisuje vzorec (38) má jeden závažný nedostatek který spočívá v tom, že algoritmus dává přednost těm atributům, které mají mnoho hodnot. Pokud jeden z atributu má stejný počet hodnot jako počet instancí v *trénovací množině*, množina bude rozdělena podle tohoto atributu a každá podmnožina bude obsahovat pouze jeden příklad, což není užitečně. Problém je vyřešen pomocí některeře normalizaci.

$$Split - info(X) = -\sum_{i=1}^{n}((|\ T_i\ |/|\ T\ |)\log_2(|\ T_i\ |/|\ T\ |)) \qquad (39)$$

$$Gain - ratio(X) = Gain\,(X)/Split - info(X) \qquad (40)$$

V modifikovánem *algoritmu C4.5* kritérium (40) nahrazuje kritérium (38), který byl použit v *ID3.*

***Neznámé hodnoty atributů***

V *trénovací množině*, je často můžou chybět některé hodnoty atributů pro některé *instancí. Algoritmus C4.5* umožňují práce s *instancí* s chybějícími udají. V *C4.5*, je platí, že *instancí* s neznámými hodnotami jsou pravděpodobnostně distribuovány podle relativní frekvence známých hodnot. $Info(T)$a $Info_x(T)$se vypočítává jako předtím, kromě toho, že uvažují se pouze *instance* se známými hodnoty. Pak *informační zisk* může být korigován pomocí faktoru *F*, který je pravděpodobnost toho, že daný atribut je známy. $F = (|\ T\ | - U)/|\ T\ |$, kdě *U* je počet nedefinovaných hodnot pro daný atribut. Pak, vzorek *(38)* bude mít podobu:

$$Gain(X) = F \cdot (Info(T) - Info_x(T)) \qquad (41)$$

Podobně, $Split - info(X)$muže být změněna, *instance* s neznámými hodnotami se berou jako další podmnožinu v rozdělení. Pokud atribut *X* má *n* hodnot, jeho $Split - info(X)$se vypočítá tak, jako by množina *instancí* byla rozdělená do $n + 1$podmnožin. Tato modifikace má přímý vliv na finální hodnotu modifikovaného kriteria $Gain - ratio(X)$.

***Prořezávání stromů***

Po zastavení algoritmu může dojít k přeučení. *Instance* odpovídající jednotlivým listovým uzlům budou patřit do stejné třídy (jak to vyžaduje algoritmus), ale listy budou asociované s malým množstvím instancí. V důsledku toho, strom bude příliš rozvětvený a málo srozumitelný. Aby zmenšit strom se používá prořezávání. Po prořezávání s listem budě asociovaná množina instancí v které převažují *instanci* té třídy, kterou označen list.

*C4.5* používá metodu, která se nazývá pesimistické prořezávání, které se provádí poté, co strom byl vytvořen. Pro každý uzel ve stromu pomocí statistických tabulek pro binomické rozdělení se vypočítavá odhad horní meze spolehlivosti $U_{cf}$. $U_{cf}$je

funkce parametrů $T_i$a $E$ pro daný uzel, kde $E$ je počet *instancí*, které patří do jiných tříd než označená v liste. *C4.5* používá 25% jako výchozí úroveň spolehlivosti a porovnává $U_{25\%}(|\ T_i\ |/E)$pro daný uzel s váženou spolehlivosti jeho listů. Vahami jsou počet instancí asociovaných s každým listem. Pokud odhadnutá chyba kořenu v podstromu je menší než vážený součet $U_{25\%}$listů, pak podstrom bude nahrazen za jeho kořenový uzel, který se stane novým listem prořezaného stromu.

#### *1.3.3.2* M5P

Algoritmus M5 na základě trénovací množiny generuje založené na stromech, po částech lineární modely. Původní algoritmus M5 byl vynalezen R. Quinlan [22] a jeho vylepšení udělal Yong Wang [23].

Algoritmus na vstup dostává kolekci z $T$ trénovacích instanci (položek ohodnocených uživatelem). Každá z instance se skládá z určené množiny nominačních nebo numerických atributů asociovaných s cílovou hodnotou (ratingem). Pro budovaní modelu:

1. Plouživá se indukční algoritmus pro budování rozhodovacího stromu (viz. 13). M5 buduje stromy, jejichž listy jsou spojeny s vícerozměrnými lineárními modely a uzly stromu jsou vybrání z atributů, které maximalizujou očekávané snížení chyb :

$$\Delta error = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i), \qquad (42)$$

kde, $T_i$je podmnožina instancí, která spojena z $i$-tym značením atributu podle kterého bylo uděláno rozdělení; $sd(T_i)$ je standardní odchylka cílových hodnot instanci z $T_i$.

2. Štěpení probíhá do té doby, pokud uzel neobsahuje velmi málo instancí a jejích cílové hodnoty se výrazně liší.
3. *Prořezávání*. Každy nelistový uzel se zkoumá začíná z dolních. Jako finalní model pro uzel M5 vybere buď zjednodušený lineární model nebo model podstromu, podle toho, co má nižší odhadovanou chybu. Pokud zvolen lineární model, podstrom v tomto uzlu se prořezávají do listu.

4. Aby se zabránilo ostré nespojitosti mezi podstromy je použita procedura hlazení. M5 vyhlazuje odhadovanou hodnotu listu ke kořeni takto:

   - Odhadovaná hodnota v listu je hodnota vypočítaná podle modelu uvedeného v tomto listu.
   - Nechť $S_i$ je větev podstromu *S*, $n_i$ je počet trénovacích instanci v $S_i$, $PV(S_i)$je predpovezena hodnota v $S_i$a *M(S)* je hodnota dána modelem v S. Pak predpovezena hodnota v *S* se rovna:

$$PV(S) = \frac{n_i \times PV(S_i) + k \times M(S)}{n_i + k}, \quad (43)$$

kde *k* je konstanta vyhlazování (výchozí se rovna15).

**Numerické atributy:**

Všechny numerické atributy jsou promění v binární proměnné tak, aby všechny rozdělí v M5P byli binární.

**Chybějící hodnoty**

Pokud jde o chybějících hodnotách, M5P používá techniku zvanou "surogatní rozdělení", která vyhledá další atribut pro rozdělení místo původního a použije ho. [24]

#### *1.3.3.3* RandomForest

Algoritmus buduje les náhodných stromů. Nechť trénovací množina obsahuje *N* instanci, kde každá instance obsahuje *M* atributů a dán parametr *m*, obvykle $m \approx \sqrt{M}$. Všechny stromy jsou vybudované nezávisle na sobě pomocí následujícího postupu:

1. Pomoci bootstrap agregaci 8 algoritmus generuje z trénovací množiny náhodnou podmnožinu s opakovaní velikosti *n*. Tak, některé instance vyskytnou v té podmnožině několikrát a asi *N/3* instanci nevyskytne vůbec.
2. Pro tuto podmnožinu algoritmus vybuduje rozhodovací strom (viz. 13), přičemž při vytvoření uzlů stromů, atribut podle kterého bude provedené štěpení bude zvolen ne ze všech *M* atributu a jen z *m* náhodně zvolených. Výběr nejlepšího pro štěpení z těchto *m* atributů může být prováděna různymi

[8]https://en.wikipedia.org/wiki/Bootstrap_aggregating

způsoby, v původním kódu se používá *Gini impurity*[9].

3. Strom se buduje do úplného vyčerpání podmnožiny a prořezávání se neprovádí.

Klasifikace testovacích instancí pak se provádí prostřednictvím voleb: každy strom přiřadí každou testovací instance určíte třídě. Zvítězí třída kterou zvolilo co nejvíc stromů. Více informace o algoritmu uvedeno v [25]

#### *1.3.3.4* RandomTree

Algoritmus buduje rozhodovací strom (viz. 13), který obsahuje *K* náhodně vybraných atributů v každém uzlu. Neprovádí žádné prořezávání.

#### *1.3.3.5* BayesNet

Bayes Network (Bayesovská síť) *B* nad množinou atributů *U* je síťová struktura $B_S$, která je orientovaným acyklickým grafem nad *U* a je množinou pravděpodobnostních tabulek $B_P = \{p(u \mid pa(u)) \mid u \in U\}$, kde *pa(u)* je množina rodičů $u$ v $B_S$. Bayesovské sítě představujou rozdělení pravděpodobnosti $P(U) = \prod_{u \in U} p(u \mid pa(u))$.

Bayesovská síť pro učení používa různé vyhledávací algoritmy a míry kvality. Vzledem k rozsáhlosti jejích popisu, algoritmy neuvedeny v texte dané diplomové práce. Seznámit se s nimi lze v [26].

#### *1.3.3.6* MultilayerPerceptron

Multilayer Perceptron (MLP)[10] je model umělé neuronové sítí který mapuje množiny vstupních dat na množiny odpovídajících výstupů. MLP se skládá z několika vrstev uzlů v orientovaném grafu, přičemž každá vrstva plně připojena k další. S výjimkou vstupní uzly, každý uzel je neuronem s nelineární aktivační funkce. Algoritmus používá techniku zvanou backpropagation pro klasifikací instance.

Aktivační funkce je modelována několika způsoby. Dvě hlavní aktivační funkce používané v současných aplikacích jsou :

[9] https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity
[10] http://en.wikipedia.org/wiki/Multilayer_perceptron

$$y(v_i) = \tanh(v_i), \quad (44)$$

kde, funkce je hyperbolický tangens, který se pohybuje od -1 do 1.

$$y(v_i) = (1 + e^{-v_i})^{-1}, \quad (45)$$

logistická funkce, je podobného tvaru, ale se pohybuje od 0 do 1. $y(v_i)$je výstup z i-tého uzlu (neuronů) a $v_i$je vážený součet vstupních synapsí (synapse jsou spojení mezi jednotlivými neurony).

MPL obsahuje tří nebo více vrstev (vstupní a výstupní vrstvu s jednou nebo více skrytých vrstev) nelineárně-aktivovaných uzlů. Každý uzel v jedné vrstvě pripojen s určitou váhou $w_{ij}$ke každému uzlu v následující vrstvě (viz. Obrázek 4).

Učení se vyskytuje v perceptronu (perceptron je nejjednodušším modelem dopředné neuronové sítě, sestává pouze z jednoho neuronu) změnou spojujcích vláh po tom, jak každý kus dat je zpracován, na základě velikosti chyby ve výstupu ve srovnání s očekávaným výsledkem.

Chyba ve výstupním uzlu *j* v *n*-té instancí představena jako $e_j(n) = d_j(n) - y_j(n)$, kde *d* je cílová hodnota a *y* je hodnota produkovaná perceptronem. Pak se provádí oprava vah uzlů založená na těch opravách, které minimalizují chyby (viz. vzorec (46) ) v celém výstupu

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \quad (46)$$

Pomocí gradientu nalezneme změny v každé vaze (viz. vzorec (47) ).

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n), \quad (47)$$

kde $y_i$je výstup předchozího neuronu a $\eta$je rychlost učení, které je pečlivě vybírána, aby zajistit, že váhy konvergují k odpovědi dostatečně rychle (používá se značení 0,8 [21]).

Derivace se vypočítá v souvislosti na indukovaných lokálních polích$v_j$

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n)), \quad (48)$$

kde $\phi'$je derivace aktivační funkce, která sama o sobě nemění. Analýza je obtížnější pro změny vah na skrytých uzlech, prislušna derivace je:

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \varepsilon(n)}{\partial v_k(n)} w_{kj}(n). \qquad (49)$$

Vzorek (49) závisí na vazích *k* uzlů které reprezentují výstupní vrstvu. Takže aby změnit váhy skryté vrstvy, musíme nejprve změnit váhy výstupní vrstvy podle derivace aktivační funkce a tak tento algoritmus představuje backpropagation aktivační funkce.

# Seznam použité literatury


1. RICH Elane. User modeling via stereotypes. Cognitive Science, vol. 3, no. 4,. October 1979, , s. pp. 329–354.

2. GOLDBERG David , David NICHOLS, Brian M. OKI and Douglas TERRY. Using collaborative filtering to weave an information tapestry. Communications of the ACM, vol. 35,no. 12. 1992, , s. pp. 61–70.

3. Michael D. Ekstrand, John T. Riedland Joseph A. Konstan. Collaborative Filtering Recommender Systems. Foundations and Trends inHuman–Computer Interaction. , , s. 81–173.

4. MELVILLE, Prem and Vikas SINDHWANI. Recommender Systems. In: Encyclopedia of Machine Learning. . : Springer, 2010, . .http://www.vikas.sindhwani.org/recommender.pdf

5. LINDEN, Greg , Brent SMITH and Jeremy YORK. Amazon.com Recommendations. Item-to-Item Collaborative Filtering. IEEE INTERNET COMPUTING. 2003, , s. .

6. JACOBI, Jennifer A., Eric A. BENSON, Gregory D. LINDEN. . In: Personalized recommendation of items represented within a database . . : Google Patents, 2006, . .http://www.google.com/patents/US7113917

7. HOWE, Michael . Pandora’s Music Recommender. . 2007, , s. .

8. GLASER, William T., Timothy B. WESTERGREN, Jeffrey P. STEARNS, Jonathan M. KRAFT. Consumer item matching method and system. . 2006, , s. .

9. RAJARAMAN, Anand, Jure LESKOVEC a Jeffrey D. ULLMAN. Recommendation Systems. In: Mining of Massive Datasets. v2.1. Palo Alto, CA: Cambridge University Press, 2014, 305-341. . Dostupné také z: http://infolab.stanford.edu/~ullman/mmds.html#original

10. PEŠKA, Ladislav. Uživatelské preference v prostředí prodejních webů. Praha, 2010. Diplomová práce. Univerzita Karlova v Praze. Matematicko-fyzikální fakulta. Vedoucí práce prof. RNDr. Peter Vojtáš, DrSC..

11. ZACHARSKI, Ron. . In: A Programmer's Guide to Data Mining: The Ancient Art of the Numerati. . : , 2012, . .Dostupné z: http://guidetodatamining.com

12. GAUCH, Susan, Mirco SPERETTA, Aravind CHANDRAMOULI, Alessandro MICARELLI. . In: User Profiles for Personalized Information Access. . : Heidelberg: Springer-Verlag Berlin, 2007, pp. 54 - 89. ISBN: 978-3-540-72078-2.

13. CLAYPOOL, Mark, Phong LE, Makoto WASEDA, David C. BROWN. . In: Implicit interest indicators. . : New York, USA: ACM, 2001, pp. 33 - 40. ISBN: 1-58113-325-1.

14. FOX, Steve, Kuldeep KARNAWAT, Mark MYDLAND, Susan DUMAIS, a Thomas WHITE. Evaluating implicit measures to improve web search. ACM Transactions on Information Systems (TOIS). 2005, Volume 23 Issue 2, s. 147-168. ISSN 1046-8188

15. ZEMIRLI, Nesrine. WebCap: Inferring the user's interests based on a real-time implicit feedback. In: Proc. ICDIM, 2012. Macau: IEEE, 2012, 62-67. ISBN 978-1-4673-2428-1

16. SU, Xiaoyuan a Taghi M. KHOSHGOFTAAR. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence. 2009, volume 2009, s. Article No. 4. ISSN 1687-7470

17. ECKHARDT, Alan. Similarity of users' (content-based) preference models for Collaborative filtering in few ratings scenario. Expert Systems with Applications: An International Journal. 2012, volume 39 issue 14, s. 11511-11516. ISSN 0957-4174

18. KOREN, Yehuda, Robert BELL a Chris VOLINSKY. Matrix Factorization Techniques for Recommender Systems. Computer. 2009, volume 42 issue 8, s. 30-37. ISSN 0018-9162

19. VALA, Martin. E-learning – doporučovací systémy. Brno, 2012. Bakalářská práce. Masarykova univerzita. Fakulta informatiky. Vedoucí práce Mgr. Jan Géryk.

20. LOPS, Pasquale , Marco de GEMMIS a Giovanni SEMERARO. Content-based Recommender Systems: State of the Art and Trends. In: F. Ricci et al. (eds.), Recommender Systems Handbook. . : Springer Science+Business Media, 2011, . ISBN 978-0-387-85820-3_3.

21. WITTEN, Ian H., Eibe FRANK, Mark A. HALL. . In: Data Mining: Practical Machine Learning Tools and Techniques. . : Morgan Kaufmann, Burlington, MA, 3rd edition, 2011, . .

22. QUINLAN, Ross J. . Learning with Continuous Classes. . 1992, , s. 343-348.

23. WANG, Y. and I. H. WITTEN. nduction of model trees for predicting continuous classes. . 1997, , s. .

24. KRAMER, Stefan . M5P. . , , s. .

25. BREIMAN, Leo. Random Forests. . 2001, , s. 5-32.

26. BOUCKAERT, Remco R. . Bayesian Network Classifiers in Weka. . 2004, , s. .

**https://sites.google.com/site/novaiso690/schema-a-priklady/pspvky-ve-sborncch**

**http://generator.citace.com/dok/ywOlRYrNsApGpbgo?kontrola=1**

[1] Ian H. Witten, Eibe Frank, and Mark A. Hall. Data Mining: Practical Machine

Learning Tools and Techniques. Morgan Kaufmann, Burlington, MA, 3rd edition, 2011.

WITTEN, Ian H., Eibe FRANK a Mark A. HALL. *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington: Morgan Kaufmann, 2011. 3rd edition. ISBN 978-0-12-374856-0. Dostupné z: http://www.cs.waikato.ac.nz/ml/weka/book.html

[3] Pazzani, M.J., Billsus, D.: Content-Based Recommendation Systems. In: P. Brusilovsky, A. Kobsa, W. Nejdl (eds.) The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, pp. 325–341 (2007). ISBN 978-3-540-72078-2

[7] Alexandros Karatzoglou. Recommender Systems Telefonica Research, Barcelona, 2013

[9] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)

[15] Balabanovic, M., Shoham, Y.: Fab: Content-based, Collaborative Recommendation. Communications of the ACM 40(3), 66–72 (1997)

[17] Bambini, R., Cremonesi, P. and Turrin, R. 2011. A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. Recommender Systems Handbook 2011: 299-331

[18] Peter Forbes, Mu Zhu: Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. RecSys 2011: 261-264

[19] Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendation. In Proceedings of the 18th National Conference on Artificial Intelligence, pages 187–192.

[20]Gözde Özbal, Hilal Karaman, Ferda Nur Alpaslan: A Content Boosted Collaborative Filtering Approach for Movie Recommendation Based on Local & Global Similarity and Missing Data Prediction. ISCIS 2010: 109-112

[21] Burke, R. Hybrid Web Recommender Systems. In P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web: Methods and Strategies of Web, Personalization, Lecture Notes in Computer Science,4321, pp. 377-408, 2007.

[24] X. Su, T.M. Khoshgoftaar, A Survey of Collaborative Filtering Techniques, Advances in Artificial Intelligence (AAI) , vol. 2009, Article ID 421425, 19 pages

[25] D. Jannach, G. Friedrich, Tutorial: Recommender Systems. Slides in International Joint Conference on Artificial Intelligence, 2013

[26] M De Gemmis, L Iaquinta, P Lops, C Musto, F Narducci, G Semeraro Preference learning in recommender systems. In: PREFERENCE LEARNING 41, 29,

2009.

[27] Wei Dai, Wei Ji. A MapReduce Implementation of C4.5 Decision Tree Algorithm. International Journal of Database Theory and Application, atabase Theory and Application, Vol.7, No.1 (2014), pp.49-60

[28] Quinlan, J.: Kantardzic, M. Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition. Chapter 6. Decision Trees and Decision Rules © 2011 by Institute of Electrical and Electronics Engineers. Published 2011 by John Wiley & Sons, Inc

[29] 70. Pazzani, M.J., Muramatsu, J., Billsus, D.: Syskill and Webert: Identifying Interesting Web Sites. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, pp. 54–61. AAAI Press / MIT Press, Menlo Park (1996)

[30] Nikovski, D.,and Kulev, V., Induction of compact decision trees for personalized recommendation. In SAC ’06: Proceedings of the 2006 ACM symposium on Applied computing, pages 575–581, New York, NY, USA, 2006. ACM.

[31] Shilpa Dharkar, Anand Rajavat. Performance Analysis of Healthy Diet Recommendation System using Web Data Mining. International Journal of Scientific & Engineering Research Volume 3, Issue 5, May-2012 ISSN 2229-5518

DHARKAR, Shilpa a Anand RAJAVAT. Performance Analysis of Healthy Diet Recommendation System using Web Data Mining. *International Journal of Scientific & Engineering Research*. 2012.

[32] Li, P., Yamada, S.: A movie recommender system based on inductive learning. In: IEEE Conf. on Cybern. and Intell. Syst., vol. 1, pp. 318–323 (2004)

[33] Guy Shani, Asela Gunawardana. Evaluating Recommendation Systems. F. Ricci et al. (eds.), Recommender Systems Handbook, DOI 10.1007/978-0-387-85820-3_8, © Springer Science+Business Media, LLC 2011

[34]Van Rijsbergen, C.J.: Information Retrieval. Butterworth-Heinemann, Newton, MA, USA (1979). URL http://portal.acm.org/citation.cfm?id=539927

[36] **Vášová, Lidmila. 1987.** *Čtenáři a uživatelé informací.* 2. vydání. Praha : Universita Karlova v Praze, 1987. str. 194. SPN 17-105-70.

[37] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):37, 30, 2009.