VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NEURONOVÉ SÍTĚ PRO DOPORUČOVÁNÍ KNIH

DEEP BOOK RECOMMENDATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MARTIN GRÁCA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2018

Abstrakt

Tato práce se zabývá oblastí Doporučovacích systémů využívající Hluboké neuronové sítě a jejich využití při doporučování knih. Jsou zde rozebrány tradiční doporučovací systémy a jejich reprezentace i systémy s pokročilejšími technikami na základě strojového učení. Jádrem práce je uplatnění konvolučních neuronových sítí pro zpracování přirozeného jazyka a vytvoření hybridního knižního doporučovacího systému. Navržený systém doporučuje na základě uživatelských dat, včetně uživatelských recenzí a knižních textových dat. Na vytvořené datové sadě systém dosahuje chyby RMSE 1,086.

Abstract

This thesis deals with the field of Recommendation systems using Deep Neural Networks and their use in book recommendation. There are the main traditional recommender systems analysed and their representations are summarized, as well as systems with more advancec techniques based on machine learning.. The core of the thesis is the use of convolutional neural networks for natural language processing and the creation of a book recommendation system. Suggested system make recommendation based on user data, including user reviews and book data, including full texts.

Klíčová slova

doporučovací systém, neuronové sítě, konvoluční neuronové sítě, knihy, knižní doporučovací systém, zpracování přirozeného jazyka, kolaborativní filtrování, faktorizace matice

Keywords

recommender system, neural networks, convolution neural networks, books, books recommender system, book recommendation, natural language processing, collaborative filtering, matrix factorization

Citace

GRÁCA, Martin. *Neuronové sítě pro doporučování knih.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Neuronové sítě pro doporučování knih

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Gráca
23. května 2018

Poděkování

Děkuji mnohokrát panu Hradišovi, který mi poskytl cenné informace, jež mi pomohly s vypracováním této diplomové práce. Velké díky patří také mé přítelkyni, která mě motivovala a byla mi oporou.

Obsah

Kapitola 1

Úvod

Internet poskytuje obrovské množství heterogenních informací a každým rokem se jejich počet rapidně zvyšuje. S tím souvisí pojem informační přetížení, kdy je člověk vystaven nadměrnému objemu informací a není tak schopen efektivně zpracovávat a využívat informace. Doporučovací systémy se tento problém snaží odstranit. Avšak obtížnost vyhodnocování a vybírání relevantních informací neustále narůstá. Proto vznikají stále nové doporučovací systémy využívající různé strategie vyhodnocování. V posledních letech dochází k velkým pokrokům v oblasti hlubokých neuronových sítích a zpracování řeči, analýze obrazu a hlavně zpracování přirozeného jazyka. Objevují se také studie [47], které demonstrují efektivnost jejich použití při řešení problémů s vyhledáváním informací a doporučeními. Na rozdíl od tradičních modelů (například kolaborativního filtrování) hluboké učení poskytuje lepší porozumění uživatelských požadavků, charakteristik položek a historické interakce mezi nimi. Dřívější metody používají k doporučení modely, jako je faktorizace matice [40], SVD [17]. Tyto systémy pracují pouze s číselným hodnocením uživatelů (ratings). Postupně vznikly i doporučovací systémy využívající i dostupné informace o položkách. Například CTR [30] a CDL [43], které zahrnují i zpracování textových dat. Nejnovější doporučovací systémy [7, 15] používají i hluboké neuronové sítě pro zlepšení doporučení.

Situace je podobná i v knižním světě. Každý rok vzniká mnoho nových knih od různých autorů. Od těch tradičních i méně známých. S rozmachem e-knih se také zjednodušuje proces vydání nové knihy, kdy není potřeba řešit tisk a vznikají tak i méně kvalitní knihy. Proto jsou doporučovací systémy důležité i v této oblasti.

Tato práce se zabývá metodami faktorizace matice [40], hlubokými neuronovými sítěmi a konvolučními sítěmi [38]. Cílem této práce je pomocí těchto metod a dostupných informací vytvořit knižní datovou sadu a knižní doporučovací systém, který eliminuje nekvalitní knižní tituly a doporučí uživatelům jen ty relevantní a kvalitní. Úspěšnost navrženého systému je měřena pomocí chyby RMSE [44] a porovnání s baseline metodami [17, 40].

V kapitole 2 jsou vysvětleny základní přístupy Doporučovacích systémů a to z několika pohledů. Z pohledu vstupních dat, výstupních dat a různých přístupů, které se v těchto systémech využívají. Tato kapitola také popisuje problémy těchto systémů a jejich možné řešení. Kapitola 3 vysvětluje problematiku konvolučních neuronových sítí, jaké vrstvy obsahuje, jak dochází ke zpracování přirozeného jazyka a jaké jsou výhody těchto sítí. Kapitola 4 se zaobírá pokročilejšími modely doporučovacích systémů a modely, ze kterých vytvořený knižní doporučovací systém čerpá. Kapitola 5 popisuje navržený knižní doporučovací systém, od vytvoření datové sady, přes návrh až po implementaci. V poslední kapitole 6 jsou uvedeny výsledky experimentů.

Kapitola 2

Doporučovací systémy

Vzhledem k tomu, jak jsou uživatelé informačně přetěžováni, doporučovací systémy představují užitečný a efektivní informační nástroj sloužící jako průvodce při objevování nových produktů a služeb, kterých je nepřeberné množství. Zastávají důležitou úlohu a zásadní roli v různých informačních systémech ke zlepšení rozhodovacích procesu.

Doporučovací systém můžeme definovat jako program, který se snaží doporučit nejvhodnější položky (produkty nebo služby) jednotlivým uživatelům (jedincům či byznysům) predikcí uživatelských zájmů o položku na základě relevantních informací o položkách, uživatelích a vztahů mezi nimi.

K pochopení a analýze aplikačního vývoje doporučovacích systémů, si v této sekci shrneme základní doporučovací techniky a také různé typy těchto systémů. Různé doporučovací systémy a techniky vznikají od poloviny 90. let a mnoho doporučovacích systémů bylo vyvinuto nedávno pro různé typy aplikací. Mnoho výzkumných pracovníků a manažerů si uvědomuje, že doporučovací systémy nabízejí velké příležitosti pro podniky, vládu, vzdělání a jiné oblasti. Důkazem je fakt, že tyto systémy využívají velké mezinárodních podniky a organizace. Zmiňme například LinkedIn, Youtube, Spotify, Amazon aj. Existuje také mnoho open-source doporučovacích, dostupných zdarma. Mnoho z těchto dostupných systémů [21] však využívá starší metody, typu kolaborativního filtrování.

Obecně, výstupem doporučovacího systém je seznam doporučených položek, který je generovaný na základě uživatelských preferencí, vlastností položek, minulé interakce mezi uživatelem a položkou a dalšími informacemi, jako jsou například dočasná a v neposlední řadě prostorová data, která jsou využívána k analýze a predikci geografických dat [2].

Doporučovací modely můžeme kategorizovat do několika typů, modely s kolaborativním filtrováním, Doporučovací systémy založené na obsahu a Hybridní doporučovací systémy.

2.1 Kolaborativní filtrování

Kolaborativní filtrování (KF) pomáhá lidem s rozhodnutím na základě názorů ostatních lidí, kteří sdílí podobné zájmy. KF doporučuje na základě historické interakce vztahu uživatel - položka, buď explicitně - např. z předchozích hodnocení uživatelů (uživatelsky založená technika), nebo pomocí implicitní zpětné vazby (předmětově založená technika) - např. z historie prohlížení. [34] Při uživatelsky založeném přístupu jsou uživateli předložena doporučení položek, které jsou oblíbené u podobných uživatelů. U předmětově založeném přístupu uživatel dostává doporučení produktů, které jsou podobné těm, které měl oblíbené v minulosti.

Podobnost mezi uživateli nebo produkty lze vypočítat pomocí Pearsonovi podobnosti založené na korelaci (Pearson Correlation-based Similarity [31]), omezené Pearsonovi podobnosti založené na korelaci, podobnosti založené na kosínu nebo měření na základě kosínu. Výše zmíněné metody při výpočtu podobnosti mezi položkami jsou brány v úvahu pouze uživatelé, kteří hodnotili obě položky. To však může mít vliv na přesnost podobnosti, když položky, které obdrželi malý počet hodnocení, vyjadřují vysokou úroveň podobnosti s ostatními položkami. Proto byla představena rozšířená verze předmětově založeného KF [36], který kombinuje měření na základě kosínu s Jaccardovou metrikou jako vážené schéma. Rovnice (2.1) demonstruje Pearsonův Koleračí koeficient:

$$sim(a,b) = \frac{\sum_{p\in P}(r_{a,p} - \tilde{r_a})(r_{b,p} - \tilde{r_b})}{\sqrt{\sum_{p\in P}(r_{a,p} - \tilde{r_a})^2}\sqrt{\sum_{p\in P}(r_{b,p} - \tilde{r_b})^2}}, \tag{2.1}$$

kde množina uživatelů je definována jako P = p_1, ...,p_m a R jako matici n x m hodnocení $r_{i,j}$ pro $i \in 1...m$ a $j \in 1...n$. Symbol (r_x) značí průměrné hodnocení uživatele x. Rovnice udává podobnost *sim(a,b)* pro uživatele a a b a vrací reálná čísla v intervalu <-1,1>. Pokud je mezi uživateli silná závislost, výstup se blíží 1. Naopak, pokud mají rozličné názory, hodnota se blíží -1.

Kolaborativní filtrování může být implementované ve formě založené na paměti (obvykle založené na uživatelích) nebo modelu. V přístupu založeném na paměti se celá databáze uživatelů uchovává v paměti a v každé operaci se prochází celá databáze. V tomto přístupu jsou doporučení přesnější, ale pokud je systémová databáze příliš velká, je tento přístup prakticky nemožný, protože existují omezení pro držení databáze v primární paměti. Navíc může být časově náročné procházet celou databázi, pokud je příliš velká.

Přístup založený na modelu [22] na druhé straně nemá výše uvedené omezení paměti. V této metodě, namísto udržování celé databáze v paměti, se uchovává v paměti pouze specifická sbírka dat, které jsou již trénovány metodami strojového učení. Přestože v některých případech existuje přístup založený na modelu, který má podobná paměťová omezení, je tento přístup účinnější a realizovatelnější v reálném světě.

Dvě primární oblasti kolaborativního filtrování jsou tedy metody založené sousedství (*neighborhood methods* a modely latentního faktoru (*latent factor models*). Metody sousedství se zaměřují na výpočet vztahů mezi položkami, nebo podobným způsobem, mezi uživateli. Přístup orientovaný na položky vyhodnocuje uživatelské preference pro položky na základě hodnocení sousedních položek stejným uživatelem. Sousední produkty jsou ty, které mají tendenci získávat stejné hodnocení od stejného uživatele. Uvažujme například knihu *Hvězdné války*, do jejich sousedních položek mohou být zahrnuty další knihy se sci-fi tématikou a tématikou vesmíru. Pro získání uživatelského hodnocení pro tuto knihu se díváme po okolních knihách, které uživatel ohodnotil.

Modely latentního faktoru jsou alternativním přístupem, který se snaží analyzovat hodnocení charakterizováním jak uživatelů, tak položek, nejčastěji mezi dvaceti až sto faktory, které jsou odvozeny od vzorců hodnocení. U knih, vyvozené faktory mohou měřit přímé dimenze, například román versus sci-fi, populárně naučné, či beletrii, mezi hůře definovatelné rozměry může patřit hloubka charakteru nebo zcela nepravděpodobné rozměry. Pro každého uživatele, každý faktor měří, jak moc má uživatel knihu v oblibě, které dosahují vysoké hodnoty na příslušném knižním faktoru. Například můžeme uvažovat dvě hypotetické dimenze, knihy orientované na muže versus knihy orientované na ženy a vážné versus odpočinkové knihy.

Faktorizace Matic

Jak již název napovídá, myšlenou faktorizace matice [40] je nalézt dvě (či více) matic takových, ze kterých dostaneme původní matici, pokud tyto matice vynásobíme mezi sebou. Této techniky se mimo jiné využívá i v doporučovacích systémech.

Faktorizace matic se využívá k nalezení latentních rysů, které podtrhují interakci mezi dvěma entitami, tedy mezi uživatelem a položkou (knihou) a jednou z aplikací je predikce hodnocení v kolaborativním filtrování.

Při zjišťování rysů předpokládáme, že počet rysů bude poměrně menší než počet uživatelů a počet položek. Pokud by tomu tak nebylo, uživatel by byl spojen pouze s jedinečnou vlastností a doporučování by nedávalo smysl, neboť každý z těchto uživatelů by neměl zájem o položky hodnocené jinými uživateli. Podobný předpoklad platí i pro položky.

Matematika faktorizace matic

Mějme množinu uživatelů U a množinu položek I. Nechť R o velikosti $|U| \times |I|$ obsahuje všechny hodnocení, které uživatelé přiřadili položkám. Předpokládáme také, že hledáme K latentních faktorů. Cílem je tedy nalézt dvě matice P ($|U| \times K$ matice) a Q ($|I| \times K$ matice) takové, že jejich skalární součin aproximuje R:

$$R \approx P \times Q^T = \hat{R}. \tag{2.2}$$

Každý řádek P reprezentuje sílu asociace mezi uživatelem a rysy. Obdobně, každý řádek Q reprezentuje sílu asociace mezi položkou a rysy. K získání predikce hodnocení položky d_j udělené uživatelem u_i, počítáme skalární součin dvou vektorů odpovídající u_i a d_j:

$$\hat{r}_{ui} = p_i^T \cdot q_j. = \sum_{k=1}^{k} p_{ik} \cdot q_{kj}. \tag{2.3}$$

Výpočet P a Q začíná inicializací těchto matic náhodnými hodnotami na matice P_0 a Q_0, výpočtem jak moc je rozdílný jejich skalární součin od matice I a poté minimalizovat rozdíl iterativně. Této metodě říkáme gradient descent, mající za cíl nalézt lokální minimum rozdílu.

Rozdíl je zde brán jako chyba mezi odhadovaným hodnocením a reálným hodnocením a může být vypočítán pro každou dvojici uživatel-položka následující rovnicí:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} \cdot q_{kj})^2. \tag{2.4}$$

Uvažujeme kvadratickou chybu, neboť odhadované hodnocení může být menší, či větší než skutečné hodnocení.

Pro minimalizaci chyby musíme znát, kterým směrem upravit hodnoty p_{ik} a q_{kj}. Jinými slovy, musíme znát gradient aktuálních hodnot a a proto rozlišujeme dvě rovnice zpracovávající dvě proměnné zvlášť:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj} \tag{2.5}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik} \tag{2.6}$$

Po získání gradientu můžeme nyní formulovat pravidla pro aktualizaci obou proměnných p_{ik} a q_{kj}:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj} \tag{2.7}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}, \tag{2.8}$$

kde α je konstanta, jejíž hodnota určuje rychlost konvergence k minimu. Obvykle se volí malá hodnota, například 0,0002. Při větším kroku směrem k minimu hrozí riziko nenalezení minima a oscilace kolem něj.

Matice P a Q, kde $P \times Q$ aproximuje matici hodnocení R neprodukuje nulové hodnocení u těch hodnocení, které chybí jak by se mohlo zdát. Při této faktorizaci se totiž nesnaží nalézt takové P a Q, které by reprodukovalo matici R naprosto přesně. Jde pouze o snahu minimalizovat chybu dostupných páru uživatel-položka. Respektive nechť T je množina n-tic, kde každá z nich je ve tvaru (u_i, d_j, r_{ij}), pak T obsahuje všechny páry uživatel-položka s hodnocením patřící této dvojici a úkolem je minimalizovat všechny chyby eij pro $(u_i, d_j, r_{ij}) \in T$. Jinými slovy T je trénovací sada. Hodnoty ostatní neznámých hodnocení je možné určit poté co je naučeno sdružování mezi uživateli, položkami a rysy.

Pomocí výše uvedených rovnic pro aktualizaci hodnot, můžeme cyklicky provádět výpočet, dokud chyba nedosáhne minima. Celková chyba je vypočítávána z následující rovnice a určuje, kdy má dojít k optimálnímu ukončení procesu:

$$E = \sum_{(u_i, d_j, r_{ij}) \in T} e_{ij} = \sum_{(u_i, d_j, r_{ij}) \in T} (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2. \tag{2.9}$$

SVD

V kontextu doporučovacích systémů se Singular Value decomposition (SVD) [17] používá jako algoritmus kolaborativního filtrování (KF). SVD je technika faktorizace matice, která se většinou používá k redukci počtu vlastností datové sady pomocí redukce dimenzionalit z D na K, kde $K < D$. Faktorizace matice [40] je prováděna na matici hodnocení $R = U \times I$, kde U je počet uživatelů a I je počet položek.

Každý vektor může být reprezentován vektorem q_i. Podobně každý uživatel může být reprezentován vektorem p_u a jejich skalární součin je tedy očekávané hodnocení:

$$hodnoceni = \hat{r}_{ui} = q_i^T \cdot p_u. \tag{2.10}$$

Vektory q_i a p_u hledáme takové, aby kvadratická chyba mezi jejich skalárním součinem a mezi známou hodnotou hodnocení v matici R byla co nejmenší:

$$minimum(p, q) = \sum_{(u,i) \in K} (r_{ui} - q_i^t \cdot p_u)^2. \tag{2.11}$$

Pro lepší generalizaci a předejití přetrénování trénovací sady, je při této metodě počítán regulační faktor λ, který se vypočítá jako násobek čtvercového součtu velikosti vektorů uživatelů a položek a reprezentuje penalizační funkci. Rovnice pro minimum poté vypadá následovně:

$$minimum(p, q) = \sum_{(u,i) \in K} (r_{ui} - q_i^T \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2). \tag{2.12}$$

Pro ilustraci tohoto faktoru si můžeme představit extrémní případ, kdy máme pouze nízké hodnocení uživatele udělené položce a žádné jiné hodnocení od uživatelé nemáme. Algoritmus poté minimalizuje chybu tak, že přiřadí vektoru q_i velkou hodnotu. Což má za následek, že všechna hodnocení od tohoto uživatele ostatním filmům budou nízká. Tento fakt však není žádoucí. Přidání velikosti vektorů do rovnice minimalizujeme nastavovaní velkých hodnot vektorům a tím je zajištěno vyhnutí se takovým situacím.

Algoritmus používá pro snížení chyby mezi predikovanou a skutečnou hodnotou hodnocení některé charakteristiky datové sady. Zejména u každé dvojice uživatelských položek (u,i) můžeme odvodit 3 parametry. Parametr μ jako průměrné hodnocení všech položek, b_i jako průměrné hodnocení položky i, od které je odečtena hodnota μ a b_u, což značí průměrné hodnocení dané uživatelem u, od které je taktéž odečtena hodnota μ. Rovnice pro očekávané hodnocení poté vypadá následovně:

$$\hat{r}_{ui} = q_i^T \cdot p_u + \mu + b_i + b_u. \tag{2.13}$$

Výsledná rovnice pro minimalizaci je poté ve tvaru:

$$minimum(p, q, b_i, b_u) = \sum_{(u,i)\in K} (r_{ui} - q_i^T \cdot p_u - \mu - b_i - b_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2). \tag{2.14}$$

Minimalizace pomocí SGD

Rovnice (2.13) je minimalizována pomocí algoritmu SGD - stochastického gradient descent [3]. Tento algoritmus začíná výpočet inicializací parametrů rovnice, kterou se snažíme minimalizovat, následně iteruje a aby snížil chybu mezi predikovanou a aktuální hodnotou využívá pro korekci předchozí hodnoty o malý faktor. Algoritmus využívá faktor nazývaný rychlost učení (learning rate), který určuje po každé iteraci poměr mezi starou a nově vypočítanou hodnotou. Prakticky při použití vysokého můžeme přeskočit optimální řešení, zatímco při použití nízkých hodnot je nezbytně nutné provést mnoho iterací, aby se dosáhlo optimální hodnoty.

2.2 Doporučovací systémy založené na obsahu

Doporučovací systém založený na obsahu je většinou založený na základě srovnání mezi jednotlivými položkami a pomocnými informacemi uživatelů. Mohou být brány v potaz různé druhy informací jako jsou obrázky, videa, či texty (recenze uživatelů, texty knih či hudby, aj.).

Systémy, které implementují doporučení založené na obsahu [26], analyzují soubor dokumentů a popis položek, které uživatel předtím ohodnotil a vytvoří model nebo profil zájmů uživatelů založených na vlastnostech objektů, které uživatel ohodnotil. Profil představuje strukturovanou podobou zájmů uživatelů, který slouží k doporučení nových zajímavých položek pro uživatele. Postup doporučení v podstatě spočívá v přizpůsobení atributů profilu uživatele a atributů obsahového objektu. Výsledkem je posudek relevantnosti, který představuje úroveň zájmu uživatele o daný objekt.

Pro efektivitu procesu přístupu k informacím je velmi důležité, aby profil přesně odrážel uživatelské preference. Například můžou být použity pro filtrování výsledků vyhledávání, a to tak, že systém rozhodne, zda má uživatel zájem o určitou webovou stránku nebo ne. V negativním případě nebude tato stránka zobrazena.

Proces doporučení se provádí ve třech krocích, z nichž každá je zpracovávaná samostatným podsystémem:

- **analyzátor obsahu** - Pokud pracujeme s nestrukturalizovanými informacemi (například texty), je nutné provést určitý typ předzpracování z důvodu získání strukturovaných relevantních informací. Hlavním úkolem této součásti je reprezentovat obsah položek (dokumentů, webových stránek, zpráv, popisů produktů, recenzí) z informačních zdrojů ve vhodné formě pro další kroky zpracování. Datové položky jsou analyzovány technikami extrakce vlastností (feature extraction) k přesunu informací z původního prostoru do cílového (například Webové stránky jsou pak reprezentované jako klíčové vektory). Tato reprezentace pak slouží jako vstup pro *profile learner* a filtrovací část.

- **profile learner** - Tento modul shromažďuje data reprezentující uživatelské preference a snaží se generalizovat data ve snaze vytvořit uživatelský profil. Za generalizační část jsou většinou zodpovědné techniky strojového učení, které dokáží vyvodit model zájmů uživatelů, který vychází z položek, které se uživateli líbili či nelíbili v minulosti. Například profile learner, který doporučuje webové stránky může implementovat metodu relevantní zpětné vazby , která kombinuje vektor pozitivních a negativních příkladů do prototypového vektoru představující uživatelský profil. Příkladem trénovacích dat jsou webové stránky, na které uživatel poskytl pozitivní nebo negativní zpětnou vazbu.

- **filtrovací část** - Tento modul využívá profil uživatele k tomu, aby navrhl relevantní položky, které získá porovnáním profilové reprezentace s položkami, které mají být doporučeny. Výsledek je binární nebo souvislý odhad vypočítaný pomocí některých z metrik podobnosti, ze kterého je vytvořen seznam potenciálně zajímavých položek. Ve výše uvedeném příkladu s webovými stránkami je porovnání provedeno výpočtem kosinové podobnosti mezi prototypovým vektorem a vektorem položek.

Za účelem vytvoření či aktualizace uživatelského profilu aktivního uživatele u_a (uživatele, pro kterého musí být poskytnuto doporučení) jsou jeho reakce na položky určitým způsobem shromažďovány a ukládány do repozitáře zpětné vazby (*feedback*). Tyto reakce, nazývající se *anotace* či *zpětná vazba*, jsou spolu s popisy souvisejících položek využívány během procesu učení modelu, který je užitečný k předpovědi významu nově prezentovaných položek. Uživatelé také mohou explicitně definovat své oblasti zájmu při prvotním vytváření profilu či účtu bez poskytnutí zpětné vazby.

K zaznamenávání zpětné vazby uživatele lze použít dvě různé techniky. Když systém vyžaduje, aby uživatel explicitně vyhodnotil položky, označujeme tuto techniku obvykle jako explicitní zpětná vazby. Druhá technika nazývaná implicitní zpětná vazba nevyžaduje žádné aktivní zapojení uživatelů v tomto smyslu. Zpětná vazby je odvozována z monitorování a analýzy aktivit uživatelů. Explicitní hodnocení [32] udává, jak důležitá nebo zajímavá položka pro daného uživatele je.

Zde jsou uvedeny tři hlavní přístupy k získávání explicitní zpětné vazby:

- *like/dislike* - položky jsou klasifikovány jako "relevantní"či "nerelevantní"použitím jednoduché binární ohodnovací stupnice

- *ohodnocení (ratings)* - posouzení položky je zde reprezentováno podle diskrétní numerické stupnice (např. 1 - 10, čím větší, tím je položka pro uživatele relevantnější)

[37] nebo je možné použití symbolického ohodnocení, které je následně mapováno na numerickou stupnici [27].

- *textové hodnocení* - komentáře k jednotlivým položkám jsou shromažďovány a předkládány uživatelům jako prostředek k usnadnění rozhodovacího procesu. Například zpětná vazba zákazníků na stránkách *Amazon.com* či *eBay.com*, může pomoci uživatelům v rozhodování v případě, že byla položka oceněna komunitou. Textové komentáře jsou užitečné, ale mohou přetížit aktivního uživatele, neboť musí číst a interpretovat každou poznámku, aby rozhodl, zda je pozitivní nebo negativní, a do jaké míry.

Explicitní zpětná vazba má výhodu jednoduchosti, přestože používáním číselných/symbolických stupnic zvyšuje kognitivní zatížení uživatele a nemusí být dostatečná pro zachycení uživatelského názoru na předměty.

Implicitní metody zpětné vazby jsou založeny na přiřazení skóre relevantnosti konkrétním uživatelským akcím spojených s položkou, například ukládání, vyřazování, tisk, bookmarking (ukládání záložek) atd. Hlavní výhodou je, že nevyžadují přímou účast uživatele, i když se může stát, že dojde k odchýlení.

Tvorba uživatelského profilu

K vytvoření uživatelského profilu aktivního uživatele u_a je potřeba definovat trénovací sadu TR_a pro uživatele u_a. TR_a je množina dvojic typu $<I_k,r_k>$, kde r_k je udělené ohodnocení poskytnuté uživatelem u_a reprezentaci položce I_k. Po předložení množiny obsahující reprezentaci položek s ohodnoceními, *profile learner* použije učící algoritmus s učitelem a vygeneruje prediktivní model - *uživatelský profil*, který je obvykle uschován v úložišti profilů k pozdějšímu využití *filtrovací části*. Po předložení nové položky *filtrovací část* predikuje, zda položka spadá do oblasti zájmu aktivního uživatele a to tak, že porovná vlastnosti v reprezentaci nové položky s těmi vlastnostmi, které jsou uloženy v uživatelském profilu. *Filtrovací část* tedy zahrnuje některé strategie pro řazení potenciálně zajímavých položek dle relevance s ohledem na uživatelský profil. Nejlépe hodnocené položky jsou obsaženy v seznamu doporučení L_a, který je předložen aktivnímu uživateli u_a. Vzhledem k tomu, že se uživatelské preference během času mění, musejí být aktualizované informace překládány části *profile learner* za účelem automatické aktualizace uživatelského profilu.

Další zpětná vazba se shromažďuje po poskytnutí uživatelům jisté doporučení, kdy uživatelé mají možnost vyjádřit jejich spokojenost či nespokojenost s položkami v seznamu L_a. Po shromáždění těchto informací se provádí učící proces znovu na nové trénovací sadě a poté se uživatelský profil adaptuje na změny v zájmech uživatelů. Metoda zpětnovazebného učení umožňuje reagovat na dynamickou povahu uživatelských preferencí.

Reprezentace položek

Položky, které mají být doporučovány uživatelům můžeme reprezentovat pomocí množiny vlastností, neboli atributů či rysů. Například u doporučování knih, atributy používané pro popis knih jsou: autor, žánr, název, originální název, rok publikace...). Každá položka je definována stejnou sadou atributů a množinou hodnot, které mohou atributy nabývat. Dostáváme pak položku, která je reprezentována strukturovanými daty. V takovém případě můžeme pro učení uživatelského profilu použít mnoho technik strojového učení.

Ve většině filtračních systémů založených na obsahu, jsou popisy položek textové vlastnosti extrahované z webových stránek, emailů, článků zpráv nebo popisů produktů. Na

rozdíl od strukturovaných dat zde nejsou žádné atributy s úplně definovanými hodnotami. Textové rysy při učení uživatelského profilu vytvářejí řadu komplikací z důvodu nejednoznačnosti přirozeného jazyka. Problém je v tom, že tradiční profily založené na klíčových slovech nejsou schopny zachytit sémantiku zájmů uživatelů, protože jsou primárně založeny na porovnání řetězců. Pokud je v profilu i v dokumentu nalezen řetězec nebo nějaká morfologická varianta, provede se shoda a dokument je považován za relevantní. Porovnávání řetězců však trpí těmito problémy. Polysemii (přítomnost více významů pro jedno slovo) a synonymy (více slov se stejným významem).

Důsledkem je, že kvůli synonymům, mohou být vynechány důležité informace, pokud profil neobsahuje přesná klíčová slova obsažená v dokumentu. Zatímco kvůli polysemii, můžeme špatné dokumenty považovat za relevantní.

Sémantická analýza a její integrace do personalizačních modelů je jedním z nejinovativnějších a nejzajímavějších přístupů navržených v literatuře k vyřešení těchto problémů. Klíčová myšlenka je přijetí znalostních základen, jako jsou lexikony, ontologie, pro anotaci položek a reprezentaci profilů za účelem získání "sémantické"interpretace potřeb uživatelů informací.

2.3 Problémy tradičních doporučovacích modelů

Výše zmíněné modely však mají své limity, které se týkají řídkosti dat (data sparsity [35]), problémů studeného startu (cold-start problems) a vyvažování doporučení kvality z hlediska různých vyhodnocovacích metrik (viz 4.5).

2.4 Hybridní doporučovací systémy

K dosažení vyššího výkonu a překonání nevýhod tradičních doporučovacích postupů byla navržena hybridní doporučení, která kombinují nejlepší vlastnosti dvou nebo více doporučovacích technik do jedné hybridní techniky. Nejběžnějším postupem ve stávajících technikách hybridních systémů je zkombinovat techniku doporučení kolaborativního filtrování s dalšími doporučovacími technikami, aby se zabránilo problémům se studeným startem, řídkostí nebo škálovatelností dat. Tyto systémy využívají jak informaci o hodnocení položek od uživatel, tak využívají pro lepší doporučení dostupné informace a data o uživatelích a položkách. V potaz jsou brány jak implicitní tak explicitní informace (získané při vytváření uživatelského profilu). [21]

Problém studeného startu

Velkým problémem pro doporučovací systémy je takzvaný problém studeného startu [19] (anglicky cold-start problem). Abychom se systém mohl adaptovat uživateli, musí vědět, co uživatel chtěl a co pro něho bylo relevantní. To je zapotřebí při filtrování založené na obsahu, aby bylo možné učinit rozhodnutí o položkách podobným těm, které se uživatelům líbili v minulosti.

Co když ještě nevíme nic o uživatelích, který právě začali používat systém? Návrháři těchto systémů mají tendenci problém vyřešit buď tím, že uživatelé ohodnotí položky na začátku, nebo tím, že se jim předloží dotazník na demografické otázky, ze kterých lze vyvodit určité stereotypy (například starší lidé více poslouchají klasickou hudbu). Vyžadujeme tedy od uživatelů explicitní vyplnění uživatelského profilu.

Obě metody vyžadují uživatelské úsilí. Rovněž není snadné se rozhodnout, které položky uživatelé mají hodnotit. Navíc mohou být stereotypy poměrně špatné a urážlivé (například někteří lidé dávají přednost populární hudbě a neradi bývají považování za starší).

O vkusu nového uživatele se dozvíme postupně například ohodnocením našich doporučených položek, nebo pomocí více implicitní metody tak, že zkoumáme zda tráví čas na těchto položkách či nikoli. Poskytujeme novému uživateli doporučení, díky nimž by skupina stávajících uživatelů mohla být spokojena, včetně nového uživatele (nebo přesněji, člověka, kterého nyní považujeme za nového uživatele). Váha, která je k novému uživateli přidělena, bude zpočátku nízká, protože ještě o ještě moc nevíme a postupně se bude zvyšovat.

Řídkost dat

Problém řídkosti dat je způsoben nedostatkem počtu transakcí a dat zpětné vazby, které omezují použitelnost a úspěšnost kolaborativního filtrování a dalších metod. Tento problém jde například minimalizovat pomocí přímo a nepřímo podobnosti mezi uživateli a výpočtem podobnostní matice přes relativní vzdálenost mezi hodnocením uživatelů [45]. V poslední době však vznikají nové doporučovací systémy, které se snaží tento problém co nejvíce minimalizovat. Využívají k tomu strojové učení a doplňují nedostatky kolaborativního filtrování při malé hustotě dat uživatelských hodnocení položek.

2.5 Rozdělení doporučovacích systémů dle výstupu

Doporučovací systémy se tedy používají k odhadu uživatelských preferencí položek, které ještě uživatelé neviděli. Na základě výstupu pak dělíme doporučovací systémy s predikci hodnocení, s predikci top-n položek a s klasifikací.

Systém s predikcí hodnocení má za cíl, co nejpřesněji vyplnit chybějící položky v matici obsahující hodnocení, které uživatel přidělil jednotlivým položkám v minulosti. Výstupem top-n systému je ohodnocený seznam položek délky n. Klasifikační systém se zaměřuje na roztřídění kandidátních položek do správných kategorií pro doporučení.

Kapitola 3

Konvoluční neuronové sítě (CNN)

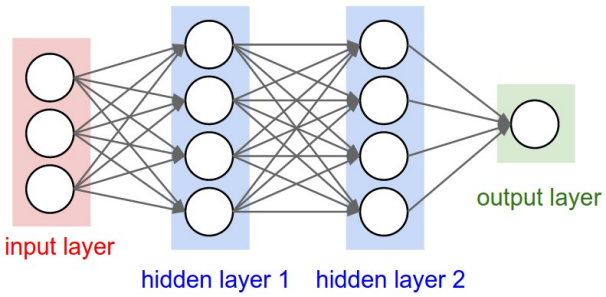
Neuronové sítě jsou biologicky inspirované programovací paradigma, které umožňuje počítačům učit se ze zkoumaných dat. Jsou složeny z velkého počtu propojených prvků, neuronů, pracujících společně při řešení problému. Umělé neuronové sítě jsou používány při řešení různých problémů, například rozpoznávání vzorů či klasifikace dat [18]. Konvoluční neuronové sítě jsou velmi podobné běžným neuronovým sítím: jsou tvořeny neurony, které obsahují naučené váhy a biasy. Každý neuron obdrží určité vstupy, provede *dot product* (sumu součinů), případně následovaný nelinearitou (nelineární aktivační funkcí, například ReLu (3.1) či tanh (3.2)). Celá síť stále představuje jednu funkci diferenciovatelného skóre. Od prvních pixelů či znaků na jednom konci po skóre třídy na konci druhém (při klasifikaci). Poslední, plně propojená síť, stále obsahuje Loss funkci (například SVM/Softmax) a síť může stále obsahovat všechny optimalizace učení, které se používají při učení obvyklých neuronových sítí.

$$f(x) = max(0, x). \tag{3.1}$$

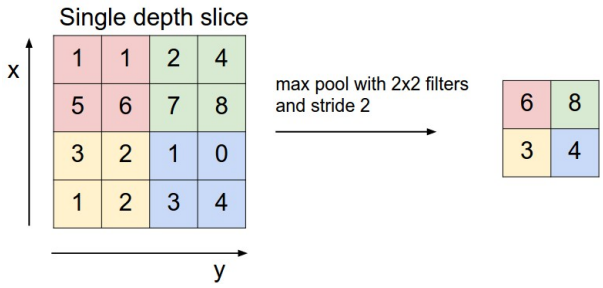
$$tanh(\alpha) = \frac{e^{2\alpha} - 1}{e^{2\alpha} + 1}. \tag{3.2}$$

Tradiční neuronové sítě mají na vstupu jeden vektor, který transformují skrze řadu skrytých vrstev. Každá skrytá vrstva je tvořena sadou neuronů, kde je každý neuron plně propojený se všemi neurony předchozí vrstvy a neurony v jedné vrstvě fungují zcela nezávislé a nesdílí žádné spojení. Poslední plně propojenou vrstvu nazýváme "výstupní vrstva"a při klasifikaci představuje skóre třídy. Ukázka takovéto sítě je uvedena na obrázku 3.1.

[1] http://cs231n.github.io/convolutional-networks/



Obrázek 3.1: tradiční 3-vrstvá neuronová síť. Převzato z [1].



Obrázek 3.2: CNN maxpool. Převzato z [2].

3.1 vrstvy CNN

Pro sestavování architektury CNN [38] využívá 3 hlavní typy vrstev: Konvoluční vrstva, Pooling vrstva a plně propojená vrstva (shodná s tou, která se využívá při tradičních neuronových sítích popsaných výše).

Konvoluční vrstva

Konvoluční vrstva je základní stavební blok konvolučních sítí, která provádí většinu výpočetně náročného procesu.

Parametry konvoluční vrstvy se skládají ze sady filtrů s možností učení. Filtr neboli pohybující se okno také nazýváme jádro (kernel), či detektor vlastností. Každý filtr je prostorově malý (v rovině délky a šířky), ale pokrývá celou hloubku vstupního objemu. Například typický filtr první vrstvy CNN při zpracování obrazu může mít velikost 5x5x3 (tj. 5 pixelů pro výšku a šířku a 3, protože obrázky mají 3 barevné kanály). Během průchodu posouváme každý filtr přes výšku a šířku vstupu a počítáme *dot product* mezi vstupy filtru a vstupy v libovolné poloze. Budeme mít tedy celou řadu filtrů (například 12 filtrů), kde každý bude produkovat oddělenou 2-dimenzionální aktivační mapu a tyto výsledky poté kombinujeme.

Pooling vrstva

V architektuře konvolučních sítí se Pooling vrstva [38] běžně vkládá za vrstvu konvoluční. Její funkcí je postupné snižování velikosti reprezentace, aby se snížili počty parametrů a výpočtů v síti, a tudíž i kontrola přetrénování. Vrstva pracuje nezávisle, na každé hloubce řezu zvlášť, který prostorově upravuje pomocí operace *max*. Například můžeme použít pooling pro okno 2x2 3.2 (u NLP typicky aplikujeme pooling přes celý výstup a tím dostaneme pouze jedno číslo pro každý filtr). Pooling vrstva také poskytuje invariantnost vůči translaci, rotaci a škálování.

Dalšími možnostmi pro pooling vrstvu je průměrné sdružování nebo sdružování pomocí L2-norm. Tato vrstva má své opodstatněné zastoupení, neboť pokud budeme vědět, že je určitá vlastnost obsažena v originálním vstupním svazku, její přesné umístění není tak důležité jako její relativní umístění k ostatním vlastnostem. Počet parametrů v této vrstvě se sníží až o 75%, čímž se sníží výpočetní náklady a také možnost přetrénování. Příznakem přetrénování je fakt, kdy model dává 100% nebo 99% na trénovací sadě, ale na testovací sadě dosahuje výkonnosti například jen 50%.

[2] http://cs231n.github.io/convolutional-networks/

Dropout vrstva

Dropout [38] vrstvy v neuronových sítích zastupují velmi specifickou roli. V předchozím odstavci byl zmíněn problém přetrénování, kdy jsou váhy sítě po natrénování nastaveny specificky pouze na testovací data a na nových vzorcích síť nedosahuje požadované přesnosti. Tato vrstva náhodně nastavuje sadu aktivací na hodnotu nula, čímž inaktivuje určitou část neuronů v dané vrstvě. Dropout vrstva se využívá pouze při fázi trénování, nikoliv však během testovacího času.

Plně propojená vrstva

Neurony v plně propojené vrstvě mají plné spojení se všemi aktivacemi v předchozí vrstvě, jak je patrné z tradičních neuronových sítí. Jejich aktivace lze tedy vypočítat pomocí násobení matice a biasu.

Konvoluční neuronové sítě jsou typicky spojeny s počítačovým viděním. CNN mají na svědomí hlavní průlomové objevy v klasifikaci obrázků a jsou jádrem většiny dnešních systémů počítačového vidění. Nedávno byly tyto sítě aplikovány na problémy zpracování přirozeného jazyka, kde dosáhli některých zajímavých výsledků. Všechny tyto vrstvy stačí na vytvoření úplné architektury konvoluční sítě. Poté, co posouváním filtru projdeme přes celou šířku a výšku, vytvoříme dvourozměrnou aktivační mapu. Síť se tedy učí filtry, která se aktivují pokud rozpoznají nějaký typ vizuálního prvku (okraj nějaké orientace, skvrny barev v první vrstvě, případně celé vzory na vyšších vrstvách sítě.

3.2 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka [6] (Natural Language Processing - NLP)se zaměřuje na interakci mezi lidským jazykem a počítači. Kříží se zde oblast informatiky, umělé inteligence a výpočetní lingvistiky. NLP je způsob, jak počítače analyzují, chápou a odvozují smysl z lidského jazyka inteligentním a užitečným způsobem. S využitím NLP mohou vývojáři organizovat a strukturovat znalosti pro provádění úkolů jako jsou automatická sumarizace, překlad, rozpoznání pojmenovaných entit, extrakce závislostí, *sentimentální analýza*, rozpoznávání řeči a rozpoznávání témat. Vývoj aplikací NLP je náročný, neboť počítače obvykle vyžadují, aby lidé s nimi komunikovali prostřednictvím programovacího jazyka. Programovací jazyky jsou přesné, jednoznačně a vysoce strukturované. Lidská řeč však není vždy přesná, je často nejednoznačná a jazyková struktura může záviset na mnoha složitých proměnných, včetně slangu, regionálních dialektů a sociálním kontextu.

Oproti pixelům při zpracování obrazu, jsou u NLP úkolů vstupem věty či dokumenty reprezentovány jako matice [16]. Každý řádek matice odpovídá jednomu *tokenu*, typicky slovu, ale může se jednat i o znak. Každý řádek je tedy vektor reprezentující slovo. Tyto vektory jsou většinou nízko-dimenzionální reprezentace (word2vec [9] či GloVe [28], viz 3.2), ale mohou to být také jednorázové vektory, které indexují slovo do slovníku. Pro větu o deseti slovech s použitím 100 dimenzí, bychom měli vstupní matici 10 x 100.

V počítačovém vidění se filtry posouvají po malých částech matice, ale u NLP obvykle používáme filtry, které se přesouvají přes celé řádky (slova). Šířka filtrů je tedy většinou stejná jako šířka vstupní matice. Výška nebo velikost oblasti se mohou lišit, ale typické jsou klouzavé okna zpracovávající 2 až 5 slov najednou. Ukázka použití konvoluční neuronové sítě pro zpracování jazyka je uvedena na obrázku 3.3.

Jako nejvhodnější problémy pro CNN se jeví klasifikační úlohy, sémantická analýza, detektor spamů nebo kategorizace témat. Konvoluční a pooling operace ztrácejí informaci o lokálním pořadí slov, takže sekvenční označování, jako například v tzv. "PoS tagging"nebo "Entity extraction", je trochu těžší vzhledem k čisté CNN architektuře.

reprezentace slov v NLP

Při zpracování přirozeného jazyka jsou slova většinou převáděny na vektory s určitými vlastnostmi [24]. Většina takzvaných *embedding* algoritmů je schopna převádět lexikální jednotky, většinou se jedná o slova, do vektorového prostoru, ve kterém morfologické, syntaktické i některé sémantické vlastnosti těchto slov zachovávají lineární závislosti.

Embedding slov je třída přístupů pro reprezentaci slov a dokumentů, které využívají hustou vektorovou reprezentaci. Jedná se o zdokonalení tradičního kódového schématu bag-of-words [42], kde se používají velké řídké vektory, které reprezentovaly každé slovo ve vektoru tak, aby reprezentovalo celou slovní zásobu. Tato vyjádření byla řídká, neboť slovní zásoby bývají většinou obrovské a dané slovo nebo dokument představuje velký vektor, který se skládá většinou z nulových hodnot.

Namísto toho, u *embedding* algoritmů [9, 28], jsou slova reprezentované hustými vektory, kde vektor představuje projekci slova do kontinuálního vektorového prostoru. Pozici slova ve vektorovém prostoru se algoritmus učí z textu a je založeno na slovech, které obklopují slovo při jeho použití. Pozice slova v naučeném vektorovém prostoru je označována jako jeho *embedding* (česky ukotvení nebo usazení).

Dva populární příklady metod učení slovních ukotvení z textu představují jsou Word2Vec [9] a GloVe [28].

Word2Vec

Nástroj *Word2Vec*[5] poskytuje efektivní implementaci architektury bag-of-words a skip-gram pro výpočet vektorové reprezentace slov. Skip-gram model obsahuje korpus slov w a jejich kontextů c. Uvažujeme podmíněnou pravděpodobnost p($c|w$) a vzhledem ke korpusu *Text* se snažíme nastavit parametry ϕ pravděpodobnosti p($c|w;\phi$) k dosažení maximální pravděpodobnosti korpusu (3.3):

$$\underset{\phi}{\operatorname{argmax}} \prod_{w \in Text} \left| \prod_{c \in C(w)} p(c|w;\phi) \right| \tag{3.3}$$

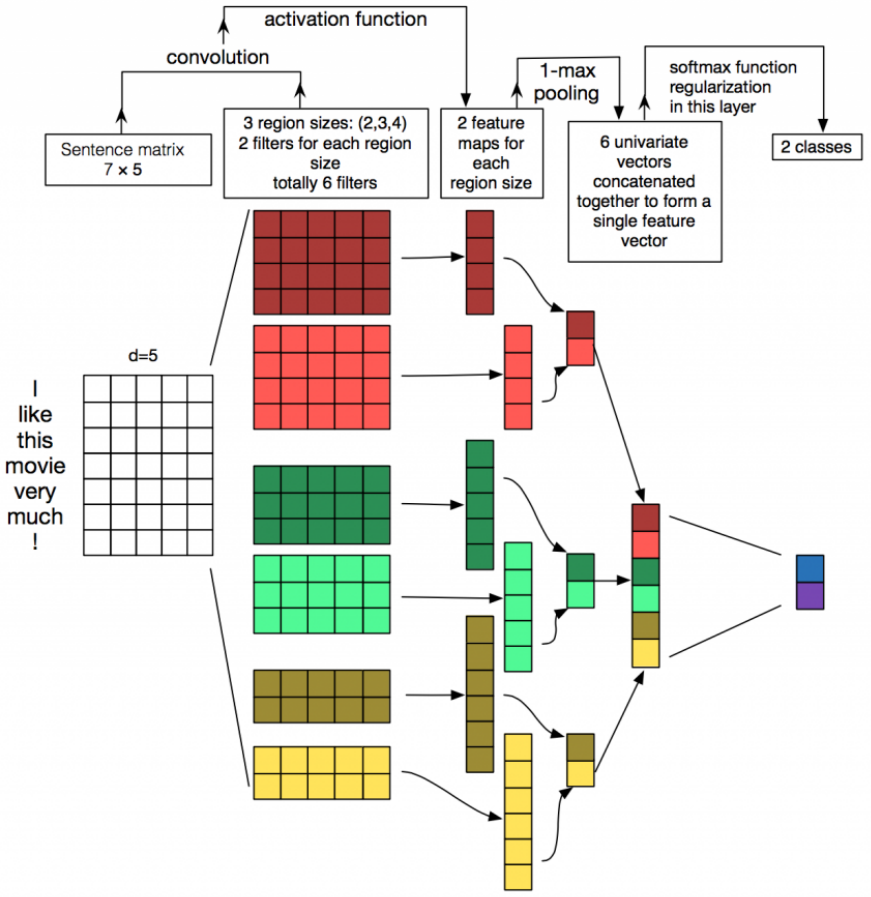
kde C(w) je množina kontextů slova w.

Jeden přístup pro parametrizaci modelu skip-gram a modelaci podmíněné pravděpodobnost pomocí softmaxu je uveden v rovnici (3.6).

$$p(c|w;\phi) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}, \tag{3.4}$$

kde v_c a $v_w \in R^d$ je vektor reprezentující c, respektive w a C je množina všech dostupných kontextů. Parametry ϕ jsou v_{c_i}, v_{w_i} pro $w \in$ V, $c \in C, i \in 1,\ldots,d$ (celkem |C| $\times$ |V| $\times$ d parametrů). Parametry volíme tak, abychom dosáhli maximální hodnoty pro (3.3) [9].

[4] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
[5] https://code.google.com/archive/p/word2vec/



Obrázek 3.3: ilustrace architektury konvoluční neuronové sítě pro klasifikaci věty. Uvádíme zde 3 velikosti filtru (2, 3, 4), z nichž každý má 2 filtry. Každý filtr provádí konvoluci na matici věty a generuje aktivační mapu různé délky. Poté aplikujeme 1-max pooling na každou z map. Tím vybereme největší číslo z dané mapy. Výsledkem je vektor ze všech šesti map, kde jsou všechny rysy spojeny a tvoří vektor pro předposlední vrstvu. Poslední softmax vrstva přijímá tento vektor jako svůj vstup a použije jej ke klasifikaci věty. Předpokládáme zde binární klasifikaci, proto zobrazujeme dva možné výstupní stavy. Převzato z [4].

GloVe

GloVe [28] je druhým algoritmem pro získávání vektorové reprezentace slov s učením bez učitele. Trénování se provádí pomocí souhrnných globálních statistik o společném výskytu mezi slovy z korpusu a výsledné reprezentace ukazují zajímavé lineární substruktury slovního vektorového prostoru.

lineární substruktury

Metriky podobnosti [28] používané pro vyhodnocení nejbližších sousedů (metody pracující s dvěma vektory slov, které efektivně měří lingvistickou, nebo sémantickou podobnost odpovídajících slov) produkují jednu skalární hodnotu, která kvantifikuje souvislost dvou slov. Tato jednoduchost může být problematická, protože dvě dané slova téměř vždy vykazují spletitější vztahy, než mohou být pokryta jedním číslem. Například slovo muž může být považováno za podobné ženě v tom, že obě slova popisují lidské bytosti. Na druhou stranu jsou obě slova často považována za protiklad, protože zdůrazňují primární osu, podél které se lidé navzájem odlišují.

Aby bylo možné kvalitativně odchytit nuance potřebné k odlišení člověka od ženy, je nutné, aby model asocioval více než jedno číslo pro dvojice slov. Jednoduchým kandidátem na rozšíření souboru diskriminačních čísel je vektorový rozdíl mezi dvěma vektory slov. GloVe je navržen tak, aby takové vektorové rozdíly zachytil nejlépe, jak je jen možné přirovnáním dvou slov.

Základní koncept, který odlišuje muže od ženy, tj. pohlaví, může být rovnocenně specifikován různými dalšími dvojicemi slov, jako jsou král a královna nebo bratr a sestra. Pro vyjádření tohoto pozorování matematicky, můžeme očekávat, že vektorové rozdíly muže a ženy, krále a královny, bratra a sestry, by mohly být téměř stejné. Tento fakt je možné vidět na obrázcích ....

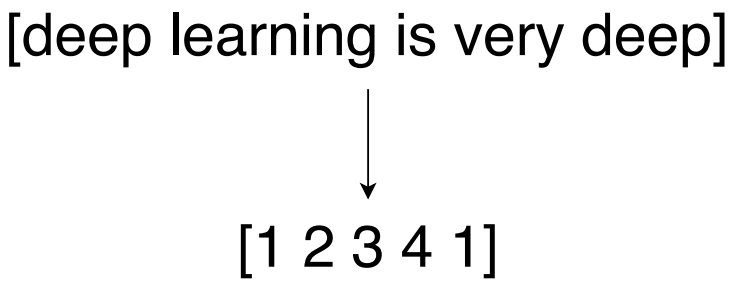
Model GloVe [28] je trénovaný na nenulových vstupech globální matice souvislosti.

Embedding vrstva NN

Embedding vrstva [4] transformuje kladná celá čísla, neboli indexy, do hustých vektorů pevné velikosti. Což je rozdíl oproti *one-hot* kódování, kdy pokud máme například slovník o 1000 slovech, reprezentujeme dané slovo vektorem o délce 1000, který obsahuje velké množství nul. Tento stav není pro velké datové sady výpočetně výhodný. Při trénování neuronové sítě s touto vrstvou, dochází k aktualizaci vektoru, který je přidružen k jednotlivým vstupům vrstvy. Důležitou fázi při použití Embedding vrstvy, je zakódování vstupních slov pomocí indexů. Ukázková reprezentace vstupní věty lze vidět na obrázku 3.4. Následně se vytváří embedding matice. Rozhodujeme se, kolik latentních faktorů přiřadíme každému indexu, což udává, jak dlouhý bude daný vektor. Obvykle se volí délky jako 32 nebo 50 [29]. To znamená, že oproti rozměrnému vektoru při *one-hot* kódování, udržujeme velikost embedding matice v rozumnější míře.

Vzhledem k tomu, že vestavěné vektory se také aktualizují během tréninku hluboké neuronové sítě, můžeme prozkoumat, která slova jsou v multidimenzionálním prostoru podobná. Vizualizace je možná pomocí technik, při kterých dochází ke snižování počtu dimenzí, například techniky *t-SNE*[6]. Zachycení takových vztahů v běžném jazyce je poměrně složité, proto jsou slovní embeddingy při zpracování přirozeného jazyka velmi důležité.

[6] https://lvdmaaten.github.io/tsne/



Obrázek 3.4: Zakódování věty pomocí indexů, které je nutné pro vstup embedding vrstvy.

Techniky analýzy slov textu

Bag of Words (BoW) [42] je algoritmus, který počítá kolikrát se dané slovo vyskytuje v dokumentu. Jednotlivé počty slov slouží k porovnání dokumentů a měření jejich podobností. Této techniky se využívá při klasifikaci, vyhledávání, či při vytváří statistických modelů. Určitý počet nejčastěji se vyskytujících slov, nebo předem stanovený slovník slov, které se mají počítat, slouží k vytvoření vstupu hluboké neuronové sítě.

TF-IDF [1] je metodika hodnocení relevance při vyhledávání textu, které vychází z BoW [42], ale bere v potaz frekvenci slov ve všech dokumentech. Z toho plyne název *Time Frequency* - frekvence slov a *Inverse Document frequency* - inverzní frekvence slov ve všech dokumentech. Tato technika zohledňuje důležitost slova v celém korpusu dokumentů, což z ní dělá jednu z nejpoužívanějších technik v doporučovacích systémech, které jsou založené na zpracování textu. první složka je definována jako:

$$tf_i, j = \frac{n_i, j}{\sum_k n_k, j} \tag{3.5}$$

, kde n_i, j je počet výskytu slova t_i v dokumentu d_j a ve jmenovatel obsahuje součet všech počtu slov v dokumentu d_j.

IDF část zohledňuje důležitost slova. Nejčastěji vyskytující se slova jsou ty nejméně důležité. Jedná se například o anglické členy "a"nebo "the". Výpočet provídáme podle vzorce:

$$idf_i = log\frac{|D|}{|\{j : t_i \in d_j\}}, \tag{3.6}$$

kde i je analyzované slovo, $|D\ rvert$ je počet dokumentů a jmenovatel obsahuje počet dokumentů, ve kterých je obsaženo zpracovávané slovo i.

Výslednou hodnotu TF-IDF [1] poté dostaneme vynásobením obou části mezi sebou, tedy $TF \cdot IDF$.

3.3 výhody CNN

Použití Rekurentních neuronových sítí ve zpracování jazyka může být intuitivnější, neboť připomínají, jak zpracováváme jazyk: čtením textu zleva doprava [38]. To však neznamená, že CNN v této oblasti nefungují. Velkou výhodou CNN je jejich **vysoká rychlost**. Konvoluce je centrální částí počítačové grafiky a je implementována na hardwarové úrovni GPU [46] (grafického procesu). Ve srovnání s n-gramy (zpracování po n slovech) jsou CNN [38] také efektivní z hlediska reprezentace. S velkou slovní zásobou může být výpočet, při použití

více než 3-gramů nákladný. Ani Google neposkytuje více než 5-gramy. Konvoluční filtry se dokáží naučit dobré interpretaci, aniž by musely reprezentovat celou slovní zásobu.

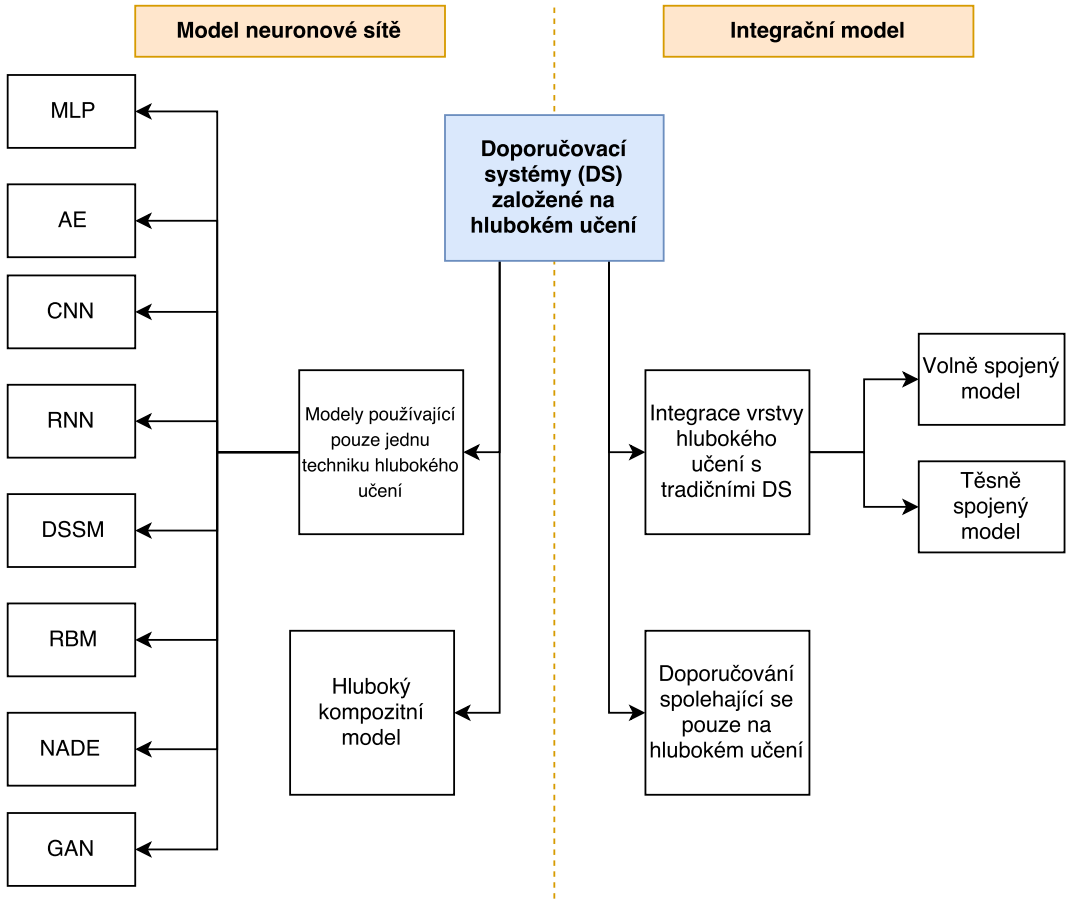
Kapitola 4

Neuronové sítě v doporučovacích systémech

Doporučovací systém je nezbytnou součástí průmyslové oblasti a je kritickým nástrojem pro podporu prodeje zboží a služeb pro mnoho on-line webových stránek a mobilních aplikací. Na základě analýzy bylo zjištěno, že 80 procent zhlédnutých filmů na streamovací službě Netflix bylo zhlédnuto právě na základě doporučení [10] a 60 procent kliknutí na video pochází z doporučení z domovské stránky YouTube [7]. V posledních době se mnoho firem uchýlilo k hlubokému učení pro zlepšení kvality doporučení [5]. Například doporučování YouTube videí, doporučovací systém pro Yahoo zprávy [25] a další. Všechny tyto systémy prokázaly výrazné zlepšení oproti tradičním doporučovacím modelům. Potenciál Hlubokých doporučovacích systémů utvrzuje také exponenciální nárůst výzkumných publikací na toto téma.

Hluboké učení je součást vědecké oblasti strojového učení. Učí se více úrovňových reprezentací a abstrakcí z dat, které dokáží vyřešit, jak s učením s pomocí učitele, tak s učením bez něj. Níže bude uveden přehled různých konceptů hlubokého učení související s doporučovacími systémy. Grafické schéma je možné vidět na obrázku 4.1.

- Vícevrstvý perceptron [33] (MLP) - dopředná neuronová síť s jednou, či více skrytými vrstvami mezi vstupní a výstupní vrstvou. Perceptron může používat různou aktivační funkci a nemusí se striktně jednat o binární klasifikátor.
- Autoenkodér [14] (AE) - dopředná neuronová sít, nejčastěji 3-vrstvá, model bez učitele. Učí se obecné znaky o vstupních datech a vyhledává mezi nimi souvislosti.
- Konvoluční neuronová síť [38] - specifická dopředná síť s konvolučními a pooling vrstvami. Je schopna zachytit globální i lokální vlastnosti a výrazně zvyšuje účinnost i přednost. Viz 3.
- Rekurentní neuronová síť - vhodná pro modelování sekvenčních dat. Na rozdíl od dopředné neuronové sítě, obsahuje smyčky a paměť k pamatování minulých výpočtů. Variantou je síť LSTM (Long Short Term Memory) nebo bránová síť GRU (Gated Recurrent Unit), která řeší problém přetrénování (Vanishing Gradient problem). Jedním z využití těchto sítí je i zpracování přirozeného jazyka.
- Model sémantické podobnosti [8] (DSSM, Deep Structured Semantic Model) - neboli hluboce strukturovaný sémantický model, je hluboká neuronová síť pro učení séman-



Obrázek 4.1: 2D schéma pro klasifikaci doporučovacích systémů založených na hlubokém učení, levá část specifikuje modely s neuronovými sítěmi, pravá část ilustruje modely integrační. Převzato z [47].

tických reprezentací entit ve společném spojitém sémantickém prostoru a pro měření jejich sémantických podobností.

- Omezený Boltzmanův stroj [39] (RBM, Restricted Boltzmann Machine) - je dvouvrstvá neuronová síť skládající se z viditelné a skryté vrstvy. Omezení znamená, že neexistuje žádná vnitřní komunikace na jednotlivých vrstvách (nenacházejí se zde spoje v rámci jedné vrstvy).
- Neural Autoregressive Distribution Estimation [41] (NADE) - volně přeloženo jako Neurální autoregresivní distrubuční odhadce je model založen na RBM 4. Je tvárný a poskytuje efektivní odhad pro modelování distribuce dat a jejich hustoty.
- Generative Adversarial Network [11] (GAN) - je generativní neuronová síť, která se skládá z diskriminátoru a generátoru. Trénují se dvě neuronové sítě současně tím, že se spolu soupeří v minimax hře.

0

4.1 Modely neuronové sítě

Jak lze vidět na obrázku 4.1, modely neuronových sítí jsou dle [47] děleny na dvě kategorie: Modely s neuronovými sítěmi a itegrační modely. Níže budou analyzovány pouze modely s neuronovými sítěmi, které jsou pro tuto práci relevantní.

Modely s neuronovými se dále dělí na dvě široké kategorie: tzv. single modely (používající pouze jednu techniku hlubokého učení) a kompozitní modely (používající dvě či více technik hlubokého učení).

Modely obsahující jednu techniku učení využívají osm hlavních technik, na kterých je postaven doporučovací systém: MLP [33], AE [14], CN [38], RNN [23], DSSM [8], RBM [39], NADE [41] a GAN [11]. Techniky Hlubokého učení určují silné stránky a aplikační možnosti doporučovacích systémů. Například MLP může jednoduše modelovat nelineární interakce mezi uživateli a položkami. CNN je schopna extrahovat lokální a globální reprezentaci z heterogenních dat (textových či vizuálních). Rekurentní neuronové sítě umožňují systémům modelovat časovou dynamiku ohodnocených dat a sekvenční vlivy obsahu. DSSM je schopna provést sémantické porovnání mezi uživateli a porovnání (disjunkci $\perp$, shodu $\equiv$, přesnější $\sqsubseteq$ nebo méně specifická $\sqsupseteq$ shoda).

Kompozitní modely. Některé doporučovací systémy používají modely s více než jednou technikou hlubokého učení. Jejich motivací je fakt, že různé techniky učení se mohou navzájem doplnit a vytvořit výkonnější hybridní model. Existuje mnoho kombinace osmi výše zmíněných technik hlubokého učení.

Z integračních modelů zmiňme jen *integrační modely s tradičním doporučovacím modelem.* Při těchto doporučovacích systémech jsou modely hlubokého učení kombinovány s tradičními doporučovacími technikami (například faktorizační matici, pravděpodobnostní faktorizační matici, faktorizačním strojem nebo metodou k-nejbližších sousedů).

Výše zmíněné modely se používají například při predikci obrázků, hudby, oblasti zájmů (Point of Interest), doporučování zpráv, hashtagů, citátů, citací a konečně i knih [47].

4.2 CTR

Kolaborativní regrese téma [30] (Collaborative Topic Regression) je jedna z dosud nejmodernějších technik, která kombinuje kolaborativní filtrování (PMF) a techniku dolování informací z textu - LDA. Využívá tak uživatelského hodnocení i dokumentů.

Jedná se o pravděpodobnostní grafový model, který integruje tématický model, latentní Dirichlet alokaci [1] (LDA) a metodu kolaborativního filtrování založenou na modelu, PMF. LDA [1] je generativní statistický model, který umožňuje pozorovanou sadu analyzovat nesledovanými skupinami, které vysvětlují, proč jsou si některé části dat podobné. Například, pokud jsou pozorovány slova, která tvoří dokumenty, pak se předpokládá, že každý dokument je směsí malého počtu témat a že každé vytvořené slovo je přiřazeno jednomu z témat dokumentů.

při LDA se na každý dokument můžeme dívat jako na směs různých témat, u nichž je každý dokument považován za soubor s tématy, které mu byly přiděleny prostřednictvím LDA. Tento přístup je totožný s pravděpodobnostní latentní sémantickou analýzou (pLSA), avšak u LDA se předpokládá, že distribuce témat má řídký Dirichlet prior. Řídký Dirichlet priors kódují intuici, že dokumenty pokrývají jen malou množinu témat a témata používají často jen malou množinu slov. V praxi to vede k lepšímu rozdělení slov a přesnějšímu přidělování dokumentů k tématům. LDA je generalizací pLSA modelu, který je ekvivalentní LDA pod jednotnou Dirichlet prior distribucí.

Předpokládejme, že existuje K témat $\beta = B_{1:K}$. Generativní proces CTR modelu je poté následující:

for all uživatele i **do**
  vypočítej latentní vektor $u_i \sim N(0, \lambda_u^{-1} I_K)$;
end for
for all položku j **do**
  vypočítej tématický poměr $\theta \sim Dirichlet(\alpha)$;
  vypočítej offset latentní položky $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$ a nastav latentní vektor položky jako $v_j = \epsilon_j + \theta_j$;
  for all slovo w_{jn} **do**
  Vypočítej přiřazení tématu $z_{jn} \sim Mult(\theta)$;
  Vypočítej slovo $w_{jn} \sim Mult(\beta z_{jn})$;
  end for
end for
for all dvojici uživatel-položka (i, j) **do**
  vypočítej hodnocení $r_{ij} \sim N(u_i^T v_j, c_{ij}^{-1})$;
end for

Model CTR [30] dobře využívá informaci o obsahu pro doporučování položek, avšak tento model není schopen se naučit uživatelský latentní prostor pro nové nebo neaktivní uživatele. Model byl důkladně studován a na základě analýz sociálních sítí bylo zjištěno, které uživatelské sociální vztahy ovlivňují rozhodovací proces uživatelů a jejich zájmů. Například uživatelé obecně důvěřují doporučení svého kamaráda při koupi nového zboží či nového filmu nebo knihy. [12]

CTR [30] je atraktivní metoda, neboť vytváří slibné a interpretovatelné výsledky. Nicméně, latentní reprezentace naučené CTR nemusí být moc efektivní, pokud jsou pomocné informace velmi řídké. Tento problém se snaží řešit metody, které budou zmíněny v další kapitolách.

4.3 CDL

CDL neboli kolaborativní hluboké učení [43] je další z nejmodernějších metod, která zvyšuje přesnost predikce hodnocení analýzou dokumentů pomocí SDAE.

SDAE [43] je dopředná neuronová síť pro učení reprezentací (kódování) vstupních dat, která se učí predikovat čistý vstup na výstupu. Prostřední skrytá vrstva X_2 má většinou úzký profil, tedy obsahuje menší počet neuronů, než ostatní vrstvy a vstupní vrstva X_0 je poškozenou verzí čistých vstupních dat. Jedná se o síť, která se učí bez učitele vrstvu po vrstvě. Síť se učí jako autoenkodér [14] minimalizující chybu při rekonstrukcí jejího vstupu, což je výstupní kód vrstvy předchozí. Jakmile je natrénováno prvních k vrstev, můžeme trénovat k+1 vrstvu, neboť nyní můžeme vypočítat kód či latentní reprezentaci dokumentu z předchozí vrstvy.

Jakmile jsou všechny vrstvy přetrénovány, síť prochází druhou fázi trénování, čímž je jemné doladění (fine-tuning). Zde probíhá učení s učitelem s cílem optimalizovat chybu pro konkrétní řešenou úlohu. Za tímto účelem přidáme logistickou regresní vrstvu, která je připojena na výstupní kód výstupní vrstvy. Celá síť se poté učí stejně jako by se jednalo o vícevrstvý perceptron [33]. V této části uvažujeme pouze kódovací část každého autoenkodéru. V této fázi již používáme cílovou třídu během trénování.

SDAE řeší následující optimalizační problém:

$$min_{\{W_l\},\{b_l\}} \|X_c - X_L\|_F^2 + \alpha \sum_l \|W_l|_F^2, \tag{4.1}$$

kde α je regularizační parametr a $\|\cdot\|_F$ reprezentuje Froenius normalizaci.

Pokud předpokládáme, že je pozorován jak čistý vstup, tak vstup poškozený, definujeme následující generativní proces:

for all vrstvu l SDAE sítě **do**
  for all sloupec n váhové matice W_l **do**
    $\mathrm{W}_{l,*n} \sim (0, \lambda_w^{-1} I_{K_l})$;
  end for
  Vypočítej bias vektor $b_l \sim N(0, \lambda_w^{-1} I K_l)$;
  for all řádek j patřící X_l **do**
    $\mathrm{X}_{l,j*} \sim N(\rho(X_{l-1,j*} W_l + b_l), \lambda_s^{-1} I K_l)$.;
  end for
  for all položku j **do**
    $\mathrm{X}_{c,j*} \sim N(X_{L,j*,\lambda_n^{-1} I_J})$.; //vypočítá čistý vstup
  end for
end for

Pokud se λ_s blíží k nekonečnu, Gausovská distribuce z rovnice 4 se stává Diracovou delta distribucí s centrem v $\rho(X_{l-1,j*} W_l + b_l)$, kde $\rho(\cdot)$ je sigmoidní funkce. Poznamenejme, že první $L/2$ vrstvy sítě vystupují jako kodéry a posledních $L/2$ vrstev vystupují jako dekodéry.

S využití Bayesovské SDAE jako části CDL, je generativní proces definován následovně:

for all vrstvu l SDAE sítě **do**
  for all sloupec n váhové matice W_l **do**
    $\mathrm{W}_{l,*n} \sim (0, \lambda_w^{-1} I_{K_l})$;
  end for
  Vypočítej bias vektor $b_l \sim N(0, \lambda_w^{-1} I K_l)$;
  for all řádek j patřící X_l **do**
    $\mathrm{X}_{l,j*} \sim N(\rho(X_{l-1,j*} W_l + b_l), \lambda_s^{-1} I K_l)$;
  end for
  for all položku j **do**
    $\mathrm{X}_{c,j*} \sim N(X_{L,j*,\lambda_n^{-1} I_J})$.; //vypočítá čistý vstup
    $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$; //vypočítá latentní offset vektor položky
    $\mathrm{v}_j = \epsilon_j + X^T_{\frac{L}{2},j*}$ //nastaví latentní vektor
  end for
  for all uživatele i **do**
    $\mathrm{u}_i = N(0, \lambda_u^{-1} I_K)$; //vypočítá latentní vektor
  end for
  for all dvojici uživatel-položka (i, j) **do**
    $\mathrm{R}_{rj} \sim N(u_i^T, v_j, C_{ij}^{-1})$; //vypočítá hodnocení
  end for
end for

kde λ_* jsou hyperparametry a Cij je parametr důvěrnosti podobně jako u metody CTR ($Cij = a$ pokud $R_{ij} = 1$, jinak $C_{ij} = b$)

4.4 KonvMF

KonvMF je model doporučovacího systému využívající k doporučení faktorizaci matice a konvoluční neuronovou síť. Model pochází od autorů z článku dostupný z [15]. Autoři se snaží vypořádat s problémem extrémního nárůstu uživatelů a tím zvyšující se řídkost hodnocení v datech obsahující interakci mezi uživatelem a položkou. Jedná se o kontextový doporučovací model, který zachycuje kontextové informace popisu dokumentů pomocí konvoluční neuronové sítě. Konvoluční faktorizace matice, zkráceně KonvMF, integruje CNN do PMF (Probabilistic Matrix Factorization nebo-li pravděpodobnostní faktorizace matice), která se v mnoha případech používá pro řešení problémů doporučování. Model efektivně využívá jak kolaborativní informaci, tak kontextovou informaci. V důsledku toho, síť KonvMF přesně předpovídá neznámé hodnocení i v případě, kdy jsou údaje o hodnocení extrémně řídké. Dle kategorizace doporučovacích systémů se tedy jedná o hybridní doporučovací systém 2.4.

Jako důkaz efektivnosti KonvMF, autoři vyhodnocovali model na třech různých datových sadách. Na, v doporučovacích systémech dobře známé datové sadě, MovieLens *ML-1m*, obsahující 1 milion hodnocení od 6000 uživatelů a 4000 filmů s hustotou $4,641\%$. Dále použili datovou sadu *ML-10*, která je rozšířením sady předchozí, zde se nachází 10 milionů hodnocení od 72 tisíc uživatelů a 10 tisíc filmů s hustotou $1,413\%$. Třetí datovou sadou je AIV (Amazon Instant Video), která obsahuje 135 tisíc hodnocení od 30000 uživatelů, 15000 položek a je extrémně řídká, její hustota je pouze $0,030\%$.

Všechny tři datové sady obsahuje explicitní hodnocení uživatelů, které udělili položkám na škále od 1 do 5. Dataset Amazonu obsahuje i slovní popis položek. U MovieLens datasetu tento popis chybí, proto byl použit popis z filmové databáze IMDB.

Jejich dost rozsáhlé experimenty s různými datovými sady a hustotami dat dokazují, že převyšují nejmodernější modely doporučovacích systémů. To je způsobeno tím, že KonvMF generuje latentní faktory položek, které efektivně zachycují kontextovou informaci popisu položek, i v případě, že jsou data hodně řídká. Dále autoři zkoumali, zda předtrénované slovní embeddingy 3.2 zlepšují kvalitu modelu KonvMF.

Pravděpodobnostní model KonvMF

Předpokládejme, že máme N uživatelů a M položek, pozorované hodnocení jsou reprezentovány maticí $R \in \mathbb{R}^{N \times M}$. Cílem je tedy najít model latentní vektor uživatelů a položek, tedy pro uživatele $U \in R^{k \times N}$ a $V \in R^{k \times M}$, kde součin $U^T \cdot V$ dává zpět matici hodnocení:

$$R = r_i j = u_i^T v_j. \tag{4.2}$$

Latentní model položky v_j je generovány ze tří proměnných 1) vnitřních váh W konvoluční neuronové sítě, 2) X_j reprezentující dokument položky j a 3) epsylon proměnné, která značí Gausův hluk, který dále umožňuje optimalizovat latentní model pro hodnocení. Finální latentní model pro položky v_j poté vypadá následovně:

$$v_j = cnn(W, X_j) + \epsilon_j \tag{4.3}$$

$$\epsilon_j = N(0, \rho_V^2 I) \tag{4.4}$$

CNN architektura KonvMF

Cílem navržené CNN architektury je generovat latentní vektor dokumentu z dokumentů patřících položkám, které se používají pro k sestavení latentní modelu položky. Navržená

CNN architektura se skládá ze 4 vrstev: 1) embedding vrstvy, 2) konvoluční vrstvy, 3) pooling vrstvy a 4) výstupní vrstvy.

Embedding vrstva transformuje vstupní dokument do husté numerické matice, která reprezentuje dokument pro další konvoluční vrstvu. Přesněji, pokud jde o dokument skládající se ze sekvence l slov, pak je reprezentován maticí, vznikající spojením vektorů slov dokumentu. Vektory slov jsou náhodně inicializovány (hodnotami od 0 do 1), nebo jsou inicializovány přetrénovanými slovními embeddingy jako jsou například GloVe 3.2, či Word2Vec 3.2. Vektory slov jsou dále trénovány pomocí optimalizačního procesu. Matice $D \in \mathbb{R}^{p \times l}$ reprezentující dokument poté vypadá následovně:

$$D = \begin{bmatrix} \dots & w_{i-1} & w_i & w_{i+1} & \dots \end{bmatrix}, \tag{4.5}$$

kde l je délka dokumentu a p je velikost embedding dimenze pro každé slov w_i.

Konvoluční vrstva extrahuje kontextové vlastnosti. Jak bylo popsáno v Sekci 3.2, dokumenty jsou poněkud rozdílné od zpracování signálu, či počítačového vidění v povaze kontextových informací. Z toho důvodu byla navržena následující architektura: kontextová vlastnost $c_i^j \in \mathbb{R}$ je extrahována ze sdílené váhy $W_c^j \in \mathbb{R}^{p \times ws}$, jejíž velikost jádra, neboli okna, velikosti ws určuje počet obklopujících slov:

$$c_i^j = f(W_c^j * D_{(:,i:(i+ws-1)} + b_c^j), \tag{4.6}$$

kde $*$ je konvoluční operátor, $b_c^j \in \mathbb{R}$ je bias pro Wc^j a f je nelineární aktivační funkce. Jako aktivační funkce je zde použita ReLU, z důvodu vyhnutí se problému "vanishing gradient", který způsobuje pomalou optimalizaci konvergence a může vést k oscilaci kolem minima.

Kontextový vektor $c_{j \in \mathbb{R}^{l-ws+1}}$ patřící dokumentu s W_c^j je poté zřejmě definován jako:

$$c^j = [c^{1j}, c_2^j, \dots, c_i^j, \dots, c_{l-ws+1}^j]. \tag{4.7}$$

Jedna sdílená váha tedy zachycuje jeden typ kontextové vlastnosti. Z toho důvodu jsou v tomto modelu používáno více sdílených vah, které zachycují více typů kontextových vlastností. Tento fakt umožňuje vytvářet vektory kontextový vlastností, kterých je tolik, kolik udává číslo n_c vektoru W_c, kde pro W_c^j je $j = 1, 2, \dots, n_c$.

Pooling vrstva

Pooling vrstva extrahuje vlastnosti z konvoluční vrstvy, a také vytváří, pomocí pooling operace, vektor vlastností fixní délky, čímž řeší problém různé délky vstupních dokumentů. Po konvoluční vrstvě je dokument reprezentován vektory s kontextovými vlastnostmi, kde každý z těchto vektorů má variabilní délku $(l-ws+1)$. Tato reprezentace však způsobuje dva problémy, prvním z nich je ten, že existuje mnoho kontextových vlastností c_i, avšak většina z těchto vlastností nemusí pomoci zvýšit výkon. Za druhé, délka vektorů kontextových vlastností je rozličná, což ztěžuje konstrukci následujících vrstev. Z výše zmíněných důvodů byla použita *max-pooling* vrstva, která redukuje reprezentaci dokumentu do vektoru fixní délky n_c, která extrahuje pouze maximální kontextovou vlastnost z každého vektoru a to dle rovnice:

$$d_f = [max(c^1), max(c^2), \dots, max(c^j), \dots, max(c^{n_c})], \tag{4.8}$$

kde c^j je vektor kontextové vlastnosti s délkou $l - ws + 1$ extrahovaný z j-té sdílené vrstvy W_c^j

Výstupní vrstva

Ve výstupní vrstvě jsou již vlastnosti vysoké úrovně z předchozích vrstev zpracovávány podle konkrétního problému, který řešíme. Zde se jedná o projekci d_j z pooling vrstvy do k-dimenzionálního prostoru latentního modelu uživatelů a položek, který je používán při doporučování. Latentní vektor dokumentu je vytvořen dle nelineární projekce dle rovnice:

$$s = tanh(W_{f_2} tanh(W_{f_1} d_f + b_{f_1}) + b_{f_2}), \tag{4.9}$$

kde $W_{f_1} \in \mathbb{R}^{f_c}$, $W_{f_2} \in \mathbb{R}^k$ jsou projekční matice a $bf_1 \in \mathbb{R}^f$, $b_{f_2} \in \mathbb{R}^k$ je bias vektor pro W_{f_1}, W_{f_2} s $s \in \mathbb{R}^k$

Optimalizace

Autoři toho článku provedli u KonvMF řadu optimalizací, která vedla ke zlepšení predikce doporučení na zkoumané datové sadě. Byli optimalizovány proměnné latentního modelu uživatelů i položek, váhy a bias proměnné CNN. Pro tuto optimalizaci byl použit MAP (maximum a posteriori) odhadovač. Výsledkem obsáhlé optimalizace je získání rovnic pro optimální řešení u_i a v_i ve tvaru:

$$u_i \longleftarrow (VI_iV^T + \alpha_U I_K)^{-1} VR_i, \tag{4.10}$$

$$v_i \longleftarrow (UI_jU^T + \alpha_V I_K)^{-1} (UR_j + \alpha_V cnn(W, X_j)), \tag{4.11}$$

kde I_i je diagonální matice obsahující I_{ij}, $j =, \ldots, M$ jako její diagonální prvky a R_i je vektor s $(r_{ij} = 1^M)$ pro uživatele i. Matice I_j a R_j je definována pro každou položku obdobně jako pro I_i a R_i. α_V je balanční parametr.

Matice W, která nemůže být optimalizována analytickým řešením z důvodu CNN architektury, zejména kvůli max-pooling vrstvě a nelinárním aktivačním funkcím, byla optimalizována algoritmem se zpětným šířením chyby (back propagation) a kvadratickou chybovou funkcí s L_2 normalizací.

Analýza časové složitosti

V každé epoše učení, jsou všechny latentní modely uživatelů a položek aktualizovány s časovou složitosti $O(k^n n_R + k^3 N + k^3 M)$, kde n_R je počet dostupných hodnocení. Aktualizace W probíhá se složitostí $O(n_c \cdot p \cdot l \cdot M)$. Celková složitost pro každou z epoch je $O(k^2 n_R + k^3 N + k^3 M + n_c \cdot p \cdot l \cdot M)$.

Poznatky

Z provedených experimentů autoři zjistili, že využití dokumentů položek při hodnocení ve velké míře pomáhá pozitivně při tvorbě latentních modelů i v případě, že jsou data hodnocení velmi řídká. Model KonvMF s přetrénovanými embeddingy předčí nejsilnější model CDL o 3-4% u MovieLens datových sad a u datové sady *AIV* o 16.6%.

Při dostatečné hustotě hodnocení (datových sad ML-1m a ML-10m) však není patrný rozdíl, pokud použijeme přetrénované slovní embeddingy nebo pokud váhy inicializujeme W náhodně. Při přetrénovaných váhách dochází ke zlepšení u ML-10m pouze v řádech tisícin procent. Při dostatečném počtu ratingů, použití přetrénovaných vah slovních vektorů může i zhoršit trénování latentních modelů a tím i výkonnost KonvMF. U méně husté datové

sady jako je AIV, však použití přetrénovaných vah pro W dává smysl a zlepšuje výkonnost modelu. Je to proto, že sémantická a syntaktická informace modelu přetrénovaných embeddingu slov doplňuje nedostatek ratingů. Se zvyšující se hustotou roste i výkonnost modelu KonvMF, což potvrzuje, že integrace CNN sítě je provedena správně.

Co se týče rozdělení datové sady na trénovací a testovací část. Jako nejoptimálnější se jeví rozdělení, kde 80% dat tvoří trénovací část a zbylých 20% je rozděleno na polovinu mezi testovací část a validační část, obě tedy obsahují 10% dat.

Důležité zastoupení má také parametr α_V, který určuje míru důležitost hodnocení a popisu dokumentů. Pokud datová sada obsahuje relativně velký počet hodnocení, přetrénované vektory slov nemají velký dopad na výkonnost při různých hodnotách α_V. Nicméně, pro extrémně řídké datové sady, mají tyto přetrénované vektory požadovaný efekt, zvlášť pokud je $\alpha_V = 100$ nebo 1000. Při relativně nízkých, či vysokých hodnotách α_V však dochází ke snížení výkonnosti. Tedy při správné hodnotě α_V, přetrénované vektory slov zvyšují výkonnost KonvMF, za předpokladu, že máme k dispozici dostatečné množství hodnocení. Pozorovatelnou změnu výkonnosti má také parametr p, který udává velikost dimenze latentních vektorů slov, pouze však při použití přetrénovaných embeddingů slov. Při náhodné inicializaci W dochází naopak u vyšších stovek p ke zvětšení chyby hodnocení.

4.5 Metriky hodnocení doporučovacích systémů

V doporučovacích systémech je pro finálního uživatele důležité dostat seřazený seznam doporučení, seřazený od nejlepších položek po ty nejhorším. V některých případech uživatele však pro uživatele není podstatné přesné seřazení, důležitější je, aby seznam obsahoval dobrá doporučení. Vezmeme-li v potaz tuhle skutečnost pro vyhodnocení doporučovacích systémů, můžeme aplikovat klasické metriky získávání informací pro získávání informací jimiž jsou *Precision* (Přesnost) a a *Recall* [13]. Tyto metriky jsou hojně využívány ve scénářích vyhledávání informací a používají je zejména vyhledávací stroje, jejichž výstupem je množina nejlepších výsledků pocházející z velkého množství možných výsledků.

Například ve vyhledávačích by se neměli objevit v nejlepších výsledcích irelevantní výsledky, ačkoli by měl být schopen vrátit co nejvíce relevantních výsledků. Můžeme říci, že Precision poměr nejlepších výsledků, které jsou relevantní s ohledem na definici relevantnosti vyhovující problému, který řešíme. To znamená, pokud říkáme Přesnost na 10, bude tento poměr brán z deseti nejlepších výsledků. Recall pak měří podíl všech relevantních výsledků, které jsou zahrnuty v seznamu nejlepších výsledků.

Formálně můžeme dokumenty považovat za instance a cílem je vrátit seznam relevantních dokumentů, které vyhovují vyhledávacímu dotazu. Nutné je přiřadit každý dokument do jedné ze dvou kategorií: relevantní nebo irelevantní. *Recall* je definován jako počet relevantních dokumentů (těch, které patří do kategorie relevantní) získaných vyhledáním vydělený počtem všech existujících relevantních dokumentů. Zatímco *Precision* je definována jako počet relevantních dokumentů získaných vyhledáním vydělený počtem všech dokumentů získaných při vyhledání.

V kontextu doporučovacích systémů je tedy *Precision* poměr doporučení, které jsou dobrými doporučeními, a *Recall* je poměr dobrých doporučení, které se objevují v seznamu nejlepších doporučení (top-n seznamu).

Při doporučování, nejlepší výsledek 1.0 pro *Precision* znamená, že každá položka doporučená v seznamu byla dobrá, neříká nic o tom, zda byla navrhnuta všechna dobrá doporučení. Skóre 1.0 pro *Recall* značí, že seznam obsahuje všechny dobré položky, které mohli být doporučení, ale neříká, kolik špatných doporučených položek seznam obsahuje.

Co se týče metrik hodnocení, obvykle se pro vyhodnocování predikce hodnocení používá střední kvadratická odchylka RMSE [44] (Root Mean Square Error) (4.12) nebo střední absolutní chyba MAE [44] (Mean Average Error) (4.13), kde n je počet vzorků, y_i je závislá proměnná, jejíž hodnota známe a $\hat{y}_i$ je predikovaná hodnota. Recall a Přesnost (Precission) [13], NDCG , kde rel_i určuje stupeň relevance výsledku (větší hodnota znamená větší relevanci) a ICG_n vrací hodnota DCG pro ideální doporučovací seznam, plocha pod křivkou AUC jsou často používána pro hodnocení skóre predikovaného pořadí (žebříčku). Přesnost, Recall a F1-skóre jsou široce používány pro hodnocení výsledků klasifikace.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.12}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{4.13}$$

$$NDCG = \frac{DCG_n}{IDCG_n} \tag{4.14}$$

$$DCG_n = \sum_{i=1}^{n}\frac{rel_i}{\ln(i+1)} \tag{4.15}$$

$$DCG_n = \sum_{i=1}^{n}\frac{ideal_i}{\ln(i+1)} \tag{4.16}$$

Kapitola 5

Hybridní knižní doporučovací systém

V této kapitole bude popsán způsob získávání a tvorba datové sady použitá pro hodnocení úspěšnosti navrženého doporučovacího systému. Dále zde bude popsán návrh doporučovacího systému a jeho implementace, od základního modelu až po finální podobu obsahující, jak text uživatelského hodnocení, tak meta informace týkající se jednotlivých knih. V poslední části této kapitoly jsou popsány implementační detaily systému, jaké knihovny byly použity, jak a kde probíhalo trénování neuronových sítí.

5.1 Tvorba datové sady

Pro tvorbu doporučovacího systému byla využita data uživatelů a knih z knižní databáze *goodreads.com*[1], kde se nachází potřebná data.

Knižní databáze *goodreads.com* poskytuje aplikační rozhraní (API), s pomocí kterého je možné extrahovat, jak data uživatelů, tak data o knihách (kromě textů knih). K tomu účelu byl napsán skript v jazyce Python, *GoodreadsDataExtractor.py*, který stahuje data uživatelů a knih ve formátu XML.

O uživatelích jsou dostupné tyto informace: uživatelské jméno, věk, pohlaví, lokace, webová stránka, datum registrace, datum poslední aktivity, seznam zájmů, seznam oblíbených knih, seznam oblíbených autorů (obsahující jméno a ID), počet přátel, počet skupin, počet hodnocení, počet recenzí, seznam přečtených knih.

O knihách jsou poskytovány tyto informace: Id knihy, název, ISBN, url obrázku, datum publikace (rok, měsíc, den), název vydavatelství, jazyk, příznak e-book (zda se jedná o e-knihu, či ne), textový popis, průměrné hodnocení,počet hodnocení, počet textových recenzí, autoři, seznam podobných knih (Id, název, počet stránek).

Další pomocný skript byl napsán ke stažení hodnocení uživatelů udělených jednotlivým knihám. Rozhraní poskytující tyto informace obsahuje data o numerickém hodnocení (rating) i textové hodnocení. Textové hodnocení však udělují uživatelé velmi málo, z toho důvodu nebyla tato informace zahrnuta do doporučovacího systému. Se změnou ochrany osobních údajů již textové hodnocení není ani skrz API goodreads dostupné. Z těchto dat byla poté vytvořena matice R uživatelských hodnocení.

Meta informace o uživatelích také nebyly do doporučovacího systému zahrnuty, neboť stažená data obsahovala velké množství chybějících hodnot v jednotlivých atributech.

[1] https://goodreads.com

Doporučovací systém tedy pracuje hlavně s meta informacemi o knihách, které jsou pro doporučování nejdůležitější.

5.2 Návrh Doporučovacího systému

Obrázek5.1 níže zobrazuje základní schéma navrženého knižního doporučovacího systému. Knižní popis je zpracováván pomocí embedding vrstvy a konvoluční neuronové sítě. Ostatní meta informace informace jsou vyvedeny jako vstup pro plně propojenou neuronovou síť.

Základ doporučovacího systému tvoří doporučení na základě faktorizace matice, tedy podle 2.3. Trénování probíhá postupným upravováním jednotlivých latentních vektorů uživatelů a položek, což umožňuje připojit neuronovou síť podle vzoru ConvMF 4.4.

Učení latentních faktorů U a V probíhá v *max_iter* iteracích. Nejprve je inicializována neuronová síť. matice U je inicializována náhodně a matice V je aktualizována výstupem neuronové sítě. Poté probíhá trénování pro každého uživatele zvlášť, kdy dochází k aktualizaci jednotlivých složek vektoru U podle [15]. Následně jsou aktualizovány složky vektoru V pro každou položku.

Obrázek 5.2 zobrazuje architekturu neuronové sítě. Navržena neuronová sít se skládá ze dvou částí. První část je tvořena konvoluční neuronovou sítí s Embedding vrstvou 3.2, která má za vstup textový popis knihy.

Konvoluční vrstva je tvořena Embedding vrstvou , Konvoluční vrstvou (Conv2D) 3.1, Maxpooling vrstvou (MaxPooling2D) 3.1 a plně propojenou vrstvou (Dense vrstva) 3.1. Embedding vrstva má velikost (300,50), kde 300 značí velikost slovníků slov a 50 počet dimenzí. Váhy této vrstvy jsou inicializovány náhodně a v průběhu trénování aktualizovány.

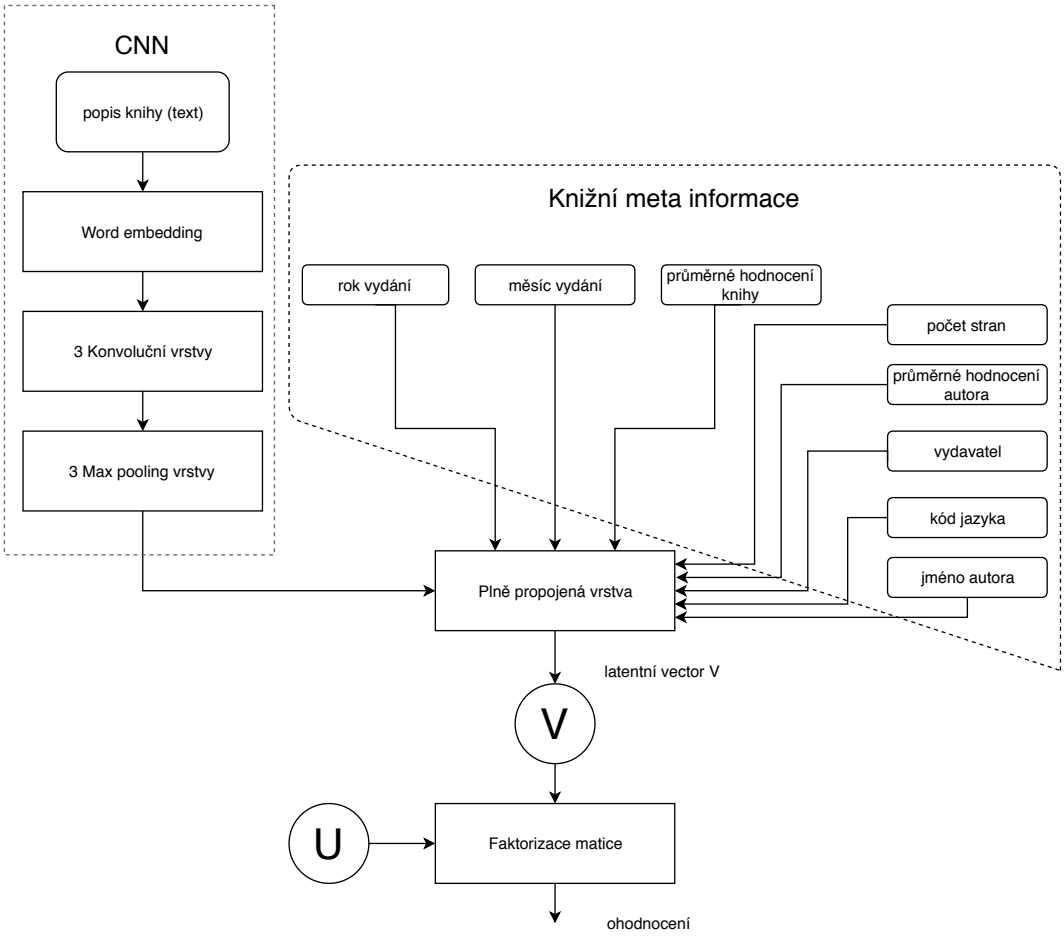
Nejprve byla použita konvoluční síť s jediným konvolučním filtrem 3.1 velikosti 3. Později byly přidány i filtry o velikosti 4 a 5. Konvoluční vrstva, jak bylo popsáno v 3.2, aplikuje filtr na celá slova. Konvoluční síť tedy zpracovává trojce, čtveřice a pětice slovních spojení. Max u pooling vrstvy znamená, že extrahuje pouze maximální kontextovou vlastnost z každého vektoru dle rovnice 4.8. Každá MaxPooling2D vrstva o velikosti 100x1x1 extrahuje 100 kontextových informací, tyto informace jsou potom spojeny do jednoho vektoru společně s meta informacemi pomocí *Concatenate* zobrazené na obrázku 5.2.

Společná síť pro obě části je tvořena dvěma plně propojenými sítěmi (Dense vrstvami), mezi které je přidána Batch Normalizace. První Dense vrstva obsahuje 128 neuronů, druhá Dense vrstva 64 a Dropout vrstva, vložena za Dense vrstvu, deaktivuje určitý počet neuronů a snižuje možnost přetrénování. Batch Normalizace by měla přispět k rychlejšímu trénování sítě. Batch Normalizace však přidává do sítě další vrstvu, čímž zpomaluje samotné trénování zvyšuje tak celkový čas trénování.

5.3 reprezentace vstupních dat

Jak bylo popsáno v kapitole 3.2 výše, vstupní data pro konvoluční vrstvu je potřeba nejprve jistým způsobem předzpracovat. Proto byl popis knih i knižní metadata převedeny do numerické reprezentace, která je vhodná jako vstup pro neuronové sítě. Jelikož popis knihy a metadata jsou zpracovávány různými neuronovými sítěmi, převod do reprezentace probíhal různým způsobem.

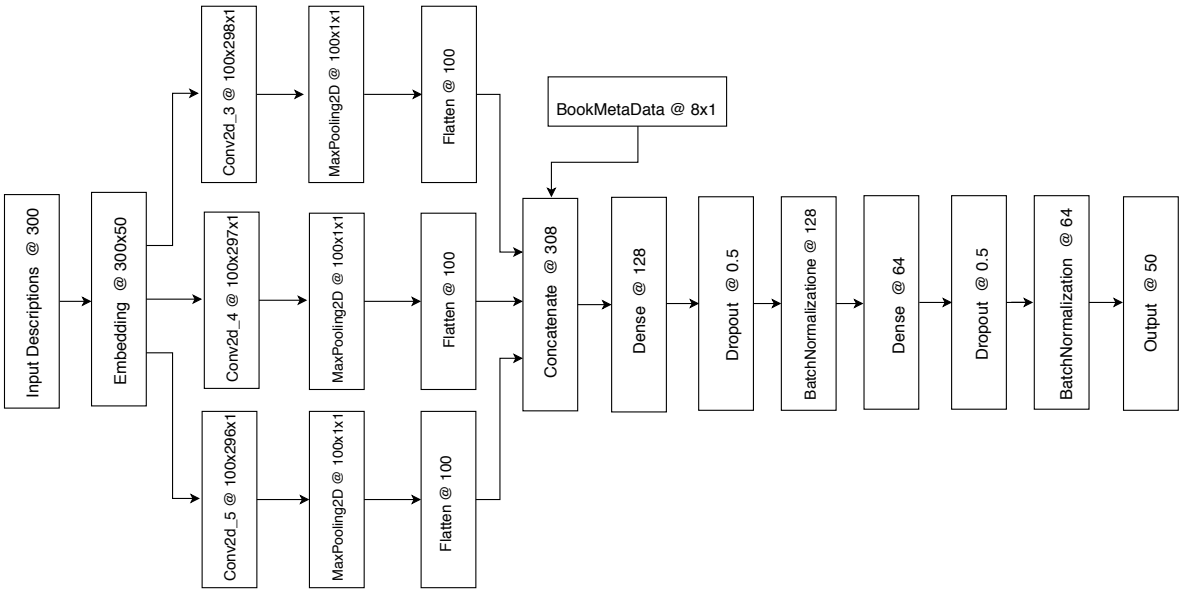
Textový popis knihy byl převeden do číselné podoby pomocí metody TF-IDF 3.2. K tomuto účelu mi posloužila funkce *TfidfVectorizer* z knihovny *sklearn.feature_extraction.text* Frekvence výskytu slov byly počítány pro slovník čítající 300 slov, které se nejčastěji vysky-



Obrázek 5.1: Navržené schéma Knižního doporučovacího systému se konvolučními sítěmi a plně propojenými vrstvami.

tují v daném korpusu. Slova ze slovníku lze nalézt v příloze níže. Slova, které mají nízkou kontextovou informaci (stop words), jako jsou spojky, předložky, jsem ze slovníku odstranil. Stop words slova jsem použil z *nltk.corpus* a lze je nalézt také v příloze níže. Frekvence jednotlivých textových popisů knih byly převedeny na vektor o velikosti 300 pomocí funkce *sequence* z knihovny *keras.preprocessing*.

Pro reprezentaci meta informací, kterých je konečný počet (vydavatel, jazyk, autor), jsem použil *LabelEncoder* z knihovny *sklearn.preprocessing*, který vytváří tolik tříd, kolik je různých hodnot v jednotlivých meta datech. Jedná se tedy o obdobu reprezentace dat pomocí one-hot vektoru, kdy je například autor reprezentován vektorem délky n obsahující n-1 nul a 1 hodnotu jedna. Číslo n značí počet různých autorů knih. Spojité hodnoty (průměrné hodnocení knihy, průměrné hodnocení autora, počet stran knihy..) jsem normalizoval do užšího intervalu <0,1> (pomocí *MinMaxScaler* z *sklearn.preprocessing*), který je vhodnější jako vstup pro neuronové sítě [20].



Obrázek 5.2: Architektura konvoluční neuronové sítě

5.4 Implementace

Při implementace jsem použil jazyk Python (verze 3.5) a řadu knihoven dostupných prostřednictví open-source distribuce Anaconda, která je hojně využívána v aplikacích týkající se zpracování dat a strojového učení. Pro práci s maticemi a s datovou sadou jsem použil knihovny *numpy* a *pandas*. Pro vytvoření neuronových sítí jsem použil *Keras*. Implementace probíhala ve vývojovém prostředí Pycharm. Výpočty probíhaly na strojích Metacentra, virtuální organizace pro akademickou obec, která sdružuje výpočetní a úložné kapacity hostované v institucích jako jsou CESNET (Praha, Brno), CERIT-SC (UVT MU), FI MU, Fyzikální ústav AV ČR a další. Pro skripty spouštějící a řídící výpočet na strojích Metacentra jsem použil jazyk *bash*. Pro výpočet jsem využíval zejména stroje, které obsahovaly grafické karty s podporu CUDA a knihovny cuDNN. Jednalo se o stroje NVIDIA Tesla K20m.

Kapitola 6

Experimenty a Výsledky

V této kapitole jsou popsány experimenty, které byly prováděny na vytvořené datové sadě a výsledky, které z experimentů vzešly. Nejprve jsem prováděl experimenty na menší datové sadě čítající 75881 uživatelů 79554 položek knih a 874035 hodnocení (ratings). Tato datová sada obsahuje pouze ty uživatele, kteří udělili 15 a více hodnocení. S ohledem na počet hodnocení v této sadě a zdroje dat, nazvěme tuto sadu *Goodreads-874k*. S touto menší datovou sadou jsem prováděl experimenty zejména ve fázi, kdy docházelo k ladění systému a optimalizaci neuronové sítě. V pozdější fázi jsem prováděli experimenty na větší datové sadě s větším počtem uživatelů a knih obsahující i uživatele s menším počtem hodnocení. Konečná větší datová sada *Goodreads-1.5m* obsahuje 1 522 224 hodnocení od 116975 uživatelů, kteří udělili nejméně 3 hodnocení a dále 134246 knih.

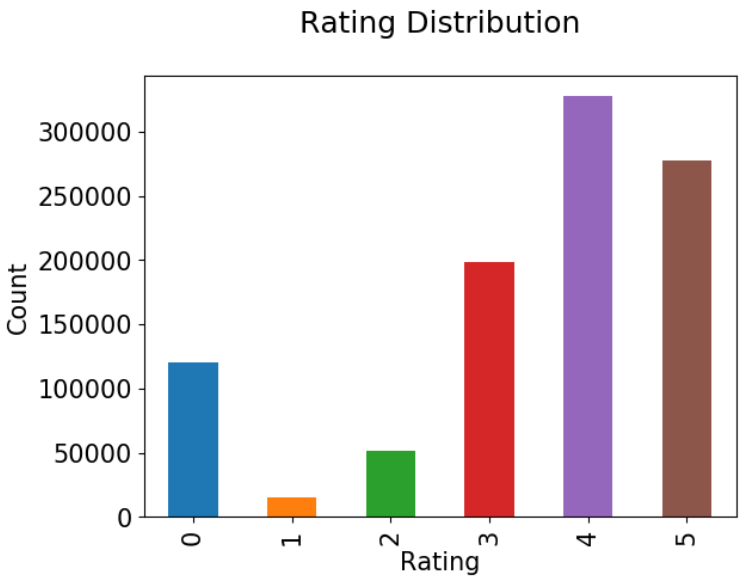
V této kapitole budou také uvedeny výsledky baseline metod, sloužící k porovnání výkonnosti navrženého doporučovacího systému. Jako baseline metoda byla zvolena metoda SVD 2.1, čistá faktorizace matice a dále statistické metody, tedy průměrné hodnocení knihy a průměrné hodnocení autora, ze kterých byla počítána chyba RMSE.

Výsledky baseline metod a všech navržených modelů jsou uvedeny v tabulce 6.1. Chyba RMSE u *avg_book* a *avg_book*, byla spočítána s průměrného hodnocení jednotlivých knih a autorů. RMSE = 1,233 bylo dosaženo při použití baseline metody SVD 2.1. MF ve výsledcích značí faktorizaci matice a výsledky pro různé velikosti dimenzí latentních faktorů U a V jsou uvedeny na řádcích *MF-10d*, *MF-30d* a *MF-50d*. *MF-meta* je model pouze s knižními meta informacemi a MF. *MF-meta-cnn* přidává i konvoluční neuronovou síť, avšak neobsahuje batch normalizaci. Nejlepšího výsledku bylo dosaženo u *MF-meta-cnn-batch*, kde byla přidána batch normalizace.

Experimenty s datovou sadou Goodreads-874k a Goodreads-1.5m

Na obrázku 6.1 je vidět graf s rozložením hodnot hodnocení (ratings), které udělili uživatelé goodreads.com knihám. Z grafu lze vyčíst, že rozložení hodnocení není úplně ideální, nejvíce uživatelé hodnotí skóre 4 nebo 5 a méně často udělují hodnocení 1 a 2.

Nejprve jsem prováděl experimenty pouze s hodnoceními uživatelů (ratings) bez informací o knihách, tedy pomocí faktorizace matice (MF) Výsledky těchto experimentů jsou uvedeny v grafu 6.2. Experimentoval jsem s různými velikostmi latentních vektorů. Nejhoršího výsledku bylo dosaženo s dimenzí 10, kdy byla nejlepší chyba na testovacích datech RMSE = 1,1598. Lepších výsledků bylo dosaženo při dimenzích 30 a 50. Při dimenzi 30 se RMSE = 1,683 a při dimenzi 50 se RMSE = 1,0596. Jak lze vidět z grafu 6.2, učení MF probíhá velmi rychle, od 20 iteraci docházelo už jen k malému zlepšení. Jedna iterace trvala

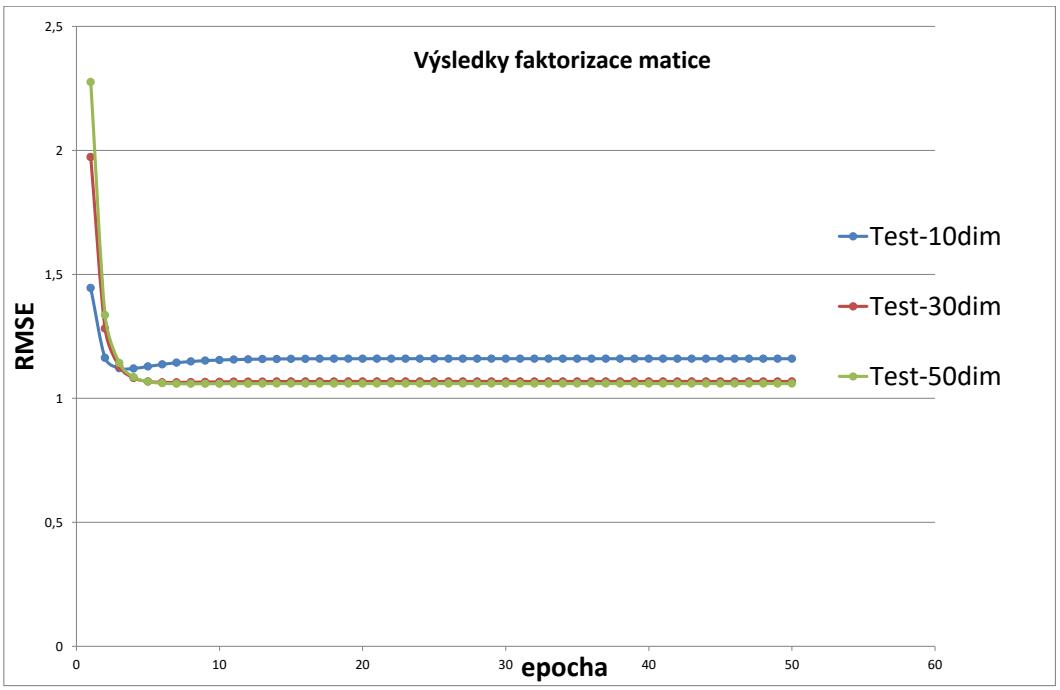


Obrázek 6.1: Graf obsahující rozložení hodnocení udělené uživateli knihám. Data pocházejí z menší datové sady. Hodnota nula byla při předzpracování dat odstraněna.

v průměru kolem 250 sekund. Celkový čas učení pro 20 iterací byl tedy cca 1,5 hodiny. Porovnání s baseline metodami a dalšími modely je uvedeno v tabulce 6.1.

Grafu 6.3 obsahuje výsledky experimentů s MF s přidanými meta informacemi o knihách v porovnání s MF využívající i text popisu knihy a knižní meta informace. Výsledky obou modelů jsou téměř shodné. RMSE u prvního modelu skončilo na hodnotě 1.0816 a u druhého modelu s texty na hodnotě 1.0863, což je něco horší výsledek, avšak druhá síť se dokázala naučit na nižší hodnotu RMSE = 0.88015, oproti RMSE = 0,9233 u modelu pouze s meta informacemi. Učení probíhalo s nastavenými parametry $\lambda_u = 10$ a $\lambda_v = 100$, které jsem nastavil experimentálně a docházelo při nich k nejlepším výsledkům. U těchto modelů už byl potřeba větší počet iterací. Graf 6.3 obsahuje 200 iterací, kde každá trvala v průměru 500 sekund, což pro 200 iterací dává cca 27 hodin.

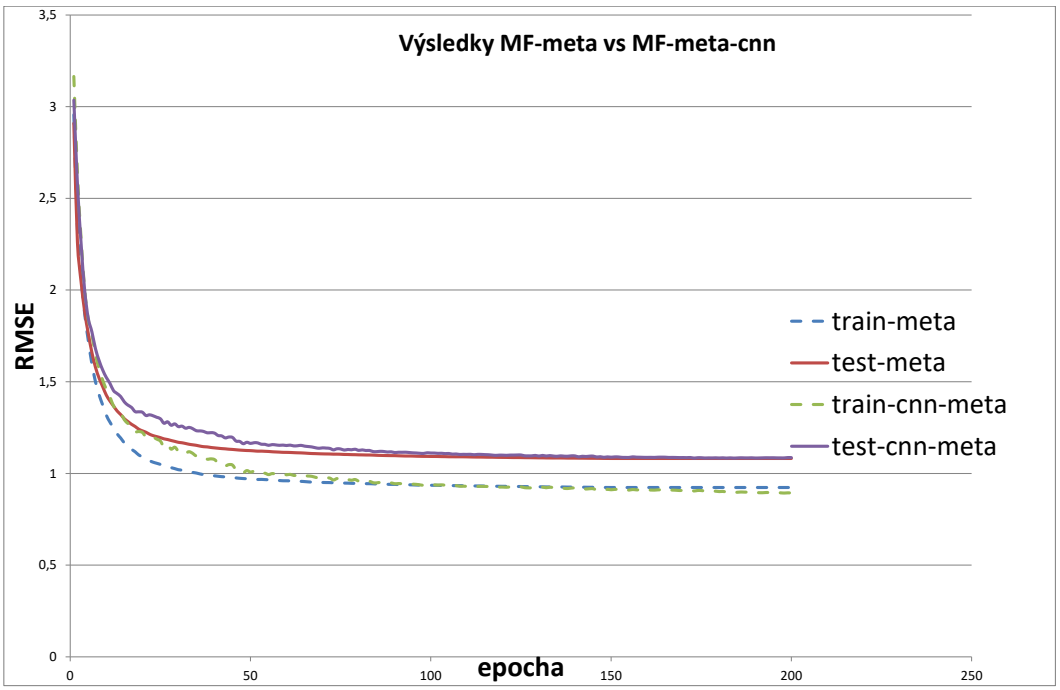
Všechny výše zmíněné experimenty jsem prováděl s datovou sadou *Gooddreads-874k*. U větší datové sady jsem čekal lepší výsledek, avšak po iteracích, 50 dimenzích, $\lambda_u = 10$ a $\lambda_v = 100$, jsem dosáhl na trénovacích datech RMSE = 0,933 a na trénovacích datech 1,098. V tabulce 6.1 uvedená jako *MF-meta-cnn-sbig*. S jiným nastavením parametrů λ bylo dosaženo poměrně horších výsledků, z čehož vyplývá, že největší vliv na úspěšnost hodnocení mají číselné hodnocení (ratings).



Obrázek 6.2: Výsledky faktorizace matice (MF) s různými velikostmi latentních vektorů

Model	**RMSE-Test**	**RMSE-Train.**
avg_book	1,891	1,903
avg_author	1,877	1.890
SVD	**1.232**	**1,233**
MF-10d	1,015	1,159
MF-30d	0,908	1,068
MF-50d	0,898	1,059
MF-meta	0,923	1,081
MF-meta-cnn	0,906	1,138
MF-meta-cnn-big	0,933	1,098
MF-meta-cnn-batch	**0,880**	**1,086**

Tabulka 6.1: Tabulka obsahující výsledné chyby RMSE testovaných modelů na sadě Goodreads-874k



Obrázek 6.3: Graf obsahující výsledky pro doporučení pouze s meta informacemi a pro doporučení obsahující meta informace i konvoluční síť využívající textový popis knihy.

Kapitola 7

Závěr

V této práci byly analyzovány doporučování systémy, jak z hlediska vnitřní reprezentace, tak z pohledu technik, které tyto tradiční systémy používají. Od těch nejstarších využívajících pouze číselné hodnocení, po ty novější využívající i strojové učení a pokročilejší techniky zpracování textových a jiných dat. Dále jsou v práci rozebrány konvoluční sítě, které jsou schopny z textových dat extrahovat sémantické informace a nachází své uplatnění i v doporučovacích systémech. Nakonec v kapitole o doporučovacích systémech využívající neuronové sítě, byly analyzovány techniky, které se v této oblasti používají, a jsou vhodné pro zpracování přirozeného jazyka.

Cílem této práce bylo vytvořit vlastní datovou sadu, navrhnout a implementovat systém, který bude doporučovat knihy a bude využívat i textové informace, což bylo splněno zahrnutím popisu knih do výsledného systému. Datová sada byla vytvořena z knižní databáze *Goodreads*. Navržený systém je hybridních, neboť využívá k doporučení metodu kolaborativního filtrování i obsahové informace o knihách.

Výsledky ukázaly, že navržený systém dosahuje lepších výsledků než baseline metody. Na testovacích datech, obsahující 876 tisíc hodnocení, bylo dosaženo chyby 1,086, kdežto u baseline metody SVD pouze 1,233. Pokročilejší metody jako CTR nebo CDL dosahují RMSE chyby pod 1,0, avšak těchto výsledků je dosaženo na větších datových sadách.

Co se týče budoucího možného vývoje, bylo by zajímavé prozkoumat, zda by vedlo ke zlepšení zahrnutí celých textů knih do doporučovacího systému. Problémem je však tyto texty legální cestou získat, neboť pro trénování by bylo potřeba titulů v řádech sto tisíců či miliónů.

Literatura

[1] Blei; David, M.; Anddrew, N.; aj.: *Latent Dirichlet Allocation . Journal of Machine Learning Research*, 2003: s. 993–1022.

[2] Bohnert, F.; Schmidt, D. F.; Zukerman, I.: *Spatial Processes for Recommender Systems*. [Online; navštíveno 29.12.2017].

[3] Bottou, L.: *Large-Scale Machine Learning with Stochastic Gradient Descent . Proceedings of COMPSTAT'2010*, September 2010: s. 177–186.

[4] Brownlee, J.: *How to Use Word Embedding Layers for Deep Learning with Keras*. 2017.
URL https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

[5] Cheng, T.; Koc, L.; Harmsen, J.; aj.: *Wide deep learning for recommender systems In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 7–10. .*

[6] Collobert, R.; Weston, J.; Bottou, L.; aj.: *Natural Language Processing (Almost) from Scratch* . August 2011: str. 24932537.

[7] Davidson, J.; Liebald, B.; Liu, J.; aj.: *The YouTube Video Recommendation System. .*

[8] Gao, J.; he, X.; Huang, P.-S.; aj.: *DSSM* .

[9] Goldberg, Y.; Levy, O.: *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method* .

[10] Gomez-Uribe, C. A.; Hunt, N.: *The Netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS) 6, 4 (2016), 13* .

[11] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; aj.: *GAN* .

[12] Hao, M.; Yang; Haixuan; aj.: *Sorec: social recommendation using probabilistic matrix factorization . Proceedings of the 17th ACM conference on Information and knowledge management*, 2008: s. 931–940.

[13] Herlocker, J. L.; Konstant, J. A.; Loren, G.; aj.: *Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems*, January 2004: str. 5–53.

[14] Hinton, G. E.; Salakhutdinov, R. R.: the Dimensionality of Data with Neural Networks. *Science*, ročník 313, July 2006: s. 504–507.

[15] Kim, D.; Park, C.; Oha, J.; aj.: *Convolutional Matrix Factorization for Document Context-Aware Recommendation . RecSys '16, Proceedings of the 10th ACM Conference on Recommender Systems*, ročník 10, September 2016: s. 233–240.

[16] Kim, Y.: *Convolutional Neural Networks for Sentence Classification . EMNLP 2014*, August 2014.

[17] Koren, Y.; Bell, R.; Volinsky, C.: *Matrix factorization techniques for recommender systems . Computer*, ročník 42, August 2009: s. 267–274, ISSN: 0018-9162.

[18] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: *ImageNet Classification with Deep Convolutional Neural Networks . Advances in Neural Information Processing Systems 25 (NIPS 2012)*, January 2012.

[19] Lam, X. N.; Lam, X. N.; Le, T. D.; aj.: *Addressing cold-start problem in recommendation systems . ICUIMC '08 Proceedings of the 2nd international conference on Ubiquitous information management and communication*, January 2008: s. 208–211, ISBN: 978-1-59593-993-7.

[20] LeCun, Y.; Bottou, L.; Genevieve, B. O.; aj.: *Efficient BackProp .*

[21] Lu, J.; Wu, D.; Mao, M.; aj.: *Recommender System Application Development s: A Survey .*

[22] Madadipouya, K.; Chelliah, S.: *A Literature Review on Recommender Systems Algorithms, Techniques and Evaluations . BRAIN: Broad Research in Artificial Intelligence and Neuroscience*, ročník 8, č. 2, July 2017.

[23] Mikolov, T.; Karafiát, M.; Burget, L.; aj.: *Recurrent Neural Network Based Language Model . INTERSPEECH 2010*, September 2010: s. 26–30.

[24] Mikolov, T.; Sutskever, I.; Chen, K.; aj.: *Distributed Representations of Words and Phrases and their Compositionality . Advances in Neural Information Processing Systems 26*, 2013.

[25] Okura, S.; Tagami, Y.; Ono, S.; aj.: *Embedding-based News Recommendation for Millions of Users. In Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. .*

[26] P., L.; de Gemmis M.; G., S.: *Content-based Recommender Systems: State of the Art and Trends. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook.* Springer, Boston, MA, 2011, ISBN 978-0-387-85819-7.

[27] Pazzani, M. J.; Muramatsu, J.; Billsus, D.: *yskill 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) .*

[28] Pennington, J.; Socher, R.; ; aj.: *GloVe: Global Vectors for Word Representation.* 2014.

[29] Piech, C.; Bassen, J.; Huang, J.; aj.: *Deep knowledge tracing . Deep knowledge tracing. In Advances in Neural Information Processing Systemspp*, 2015: s. 503–513.

[30] Purushotham, S.; Liu, Y.; Kuo, G.-C. J.: *Collaborative Topic Regression with Social Matrix Factorization for Recommendation Systems .* June 2012.

[31] Resnick, P.; Iacovou, N.; Suchak, M.; aj.: *GroupLens: an open architecture for collaborative filtering of netnews .*

[32] Rich, E.: *User Modeling via Stereotypes. Cognitive Science .*

[33] Ruck, D. W.; Rogers, S. K.; Kabrisky, M.: *The multilayer perceptron as an approximation to a Bayes optimal discriminant function.* Dec 1990: s. 296 – 298.

[34] Sarwar, B.; Karypis, G.; Konstan, J.; aj.: *Item - based collaborative filtering recommendation algorithms* .

[35] Sarwar, B. M.: *Sparsity, scalability, and distribution in recommender systems* . *Doctoral Dissertation*, 2001, ISBN: 0-493-04207-5.

[36] Shambour, Q.; Lu, J.: *hybrid trust - enhanced collaborative filtering recommendation approach for personalized government - to - business e - services, International Journal of Intelligent Systems* .

[37] Shardanand, U.; Maes, P.: *Social information filtering: algorithms for automating "word of mouth"* .

[38] Stutz, D.: *IUnderstanding Convolutional Neural Networks. Seminar Report*, August 2014.

[39] Sutskever, I.; Hinton, G. E.; Taylor, L. W.: Recurrent Temporal Restricted Boltzmann Machine . *Advances in Neural Information Processing Systems 21*, 2008.

[40] Takács, G.; Pilászy, I.; Pilászy, I.; aj.: *Matrix factorization and neighbor based algorithms for the netflix prize problem . RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems*, 2008: s. 267–274.

[41] Uria, B.; Coté, M.-A.; Gregor, K.; aj.: *Neural Autoregressive Distribution Estimation . Journal of Machine Learning Research 17*, September 2016: s. 1–37.

[42] Wallach, H. M.: *Topic modeling: beyond bag-of-words. ICML '06 Proceedings of the 23rd international conference on Machine learning*, June 2006: s. 977–984.

[43] Wang, H.; Wang, N.; Yeung, D.-Y.: *Collaborative Deep Learning for Recommender Systems. KDD '15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2015: s. 1235–1244, ISBN: 978-1-4503-3664-2.

[44] Willmott, C. J.; Matsuura, K.: *Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance . Climate Research*, December 2005: s. 79–82.

[45] Wu, C.; Xie, M.; Guo, X.: *Resolving the Sparsity Problem in Recommender Systems Using Association Retrieval* .

[46] Zhang, C.; Li, P.; Sun, G.; aj.: *Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks*. February 2015.

[47] Zhang, S.; Yao, L.; ; aj.: *Deep learning based recommender system: A survey and new perspectives*.

Příloha A

Obsah přiloženého paměťového média

- DeepBookRecommendationDIP - zdrojove kódy, stažená data, výsledky
- DeepBookRecommendationDIP/data stažená datová sada a výsledky
- DeepBookRecommendationDIP/recommendersystem - zdrojové soubory obsahující hlavni třídu RunLearning.py spouštějící trénování
- DeepBookRecommendationDIP/recommendersystem/dataFormated - zformatovana datovaá sada
- VIDEO_LINK.txt - video prezentující tuto práci

Příloha B

použitý slovník a stop-words

stop words = will, did, aren, you've, yourselves, herself, aren't, our, by, me, same, below, yourself, isn, all, that'll, they, in, nor, doing, wasn't, both, weren't, under, their, during, over, wouldn't, wasn't, few, myself, but, mos..., between, my, couldn't, from, her, it's, shouldn't, own, wouldn, being, has

slovník = science, real, get, around, still, set, original, look, em, case, school, every, everything, heart, much, show, american, read, soon, nature, come, last, body, reveals, earth, mystery, better, sense, keep, comes, next, back, english, one, human, explores, must, public, see, story, extraordinary, contemporary, works, night, change, begins, account, land, first, another, mother, popular, things, wife, family, women, later, de, includes, country, us, power, moving, made, shows, four, classic, language, rich, master, left, provides, city, characters, many, full, part, questions, play, love, day, second, tells, go, become, beautiful, dead, secret, since, tale, town, girl, twenty, award, bestselling, finally, called, written, business, light, today, magic, self, wants, winning, others, takes, live, parents, mind, everyone, finds, learn, book, lost, personal, collection, brilliant, best, edition, far, times, literature, events, people, along, books, cultural, even, black, search, man, relationship, son, young, true, early, hope, guide, whose, historical, age, culture, three, published, la, year, including, based, well, house, life, author, un, without, art, truth, big, take, information, que, introduction, use, politics, national, dark, often, across, among, daughter, ways, research, secrets, unique, journey, writing, nothing, woman, york, may, writer, experience, career, volume, home, important, friend, deep, turns, greatest, america, political, past, friends, anyone, series, god, away, years, making, make, society, boy, help, different, way, powerful, days, always, behind, understanding, husband, death, new, small, men, world, fascinating, white, becomes, en, something, amp, known, stories, father, ancient, lives, literary, social, murder, knows, makes, offers, child, major, reading, beyond, want, work, find, ever, hard, thought, yet, future, know, right, would, el, could, living, place, two, writers, short, together, little, great, end, brings, food, also, need, understand, funny, dangerous, within, might, mysterious, five, turn, children, old, king, really, long, face, high, seems, free, century, history, novel, discover, perfect, marriage, adventure, john, readers, war, modern, time, good, br, reader, fiction, found, like, never