|  |
| --- |
|  |
| Uměle inteligentní doporučovací systémy |
| Autor: Patrik Müller  Ilustrace: Dominik Müller, Patrik Müller |
| 2024 |

|  |
| --- |
| Poděkování  Děkuji Bogdanu Walkovi za vedení této diplomové práce a zprostředkování potřebných zdrojů, stejně tak Pavlovi Smolkovi za zprostředkování serveru pro testování, evaluaci a provoz. Děkuji mému bratrovi Dominikovi za poskytnutí ilustrací pro neformální část této knihy vytvořené pro popularizaci tématu. |

OBSAH

[Poděkování 2](#_Toc177021738)

[OBSAH 3](#_Toc177021739)

[ÚVOD 7](#_Toc177021740)

[AUTOR 7](#_Toc177021741)

[PŘEDMLUVA 8](#_Toc177021742)

[1 Neformální úvod do umělé inteligence 12](#_Toc177021743)

[1.1 Co je to umělá inteligence? 12](#_Toc177021744)

[1.2 Optimalizace 19](#_Toc177021745)

[1.3 Lineární regrese 21](#_Toc177021746)

[1.4 Základní strojová inteligence 23](#_Toc177021747)

[1.4.1 Piškvorky: ukázka primitivního přístupu k umělé inteligenci 23](#_Toc177021748)

[1.5 Logika a umělá inteligence 24](#_Toc177021749)

[1.6 Pravděpodobnost 26](#_Toc177021750)

[1.7 Redukce dimenzí 27](#_Toc177021751)

[1.8 Klasifikace 28](#_Toc177021752)

[1.9 Shlukování 30](#_Toc177021753)

[1.10 Evaluace 31](#_Toc177021754)

[1.11 Neuronové sítě 32](#_Toc177021755)

[2 Současné doporučovací systémy 37](#_Toc177021756)

[2.1 Doporučovací systému pro novinové články 43](#_Toc177021757)

[2.2 Současné aplikace zabývající se zobrazováním a doporučováním článků 45](#_Toc177021758)

[2.2.1 Google News 46](#_Toc177021759)

[2.2.2 Apple News 47](#_Toc177021760)

[2.2.3 Flipboard 48](#_Toc177021761)

[2.2.4 *Feedly* 48](#_Toc177021762)

[2.2.5 *Pocket* 49](#_Toc177021763)

[2.2.6 FlashNews 50](#_Toc177021764)

[2.2.7 Shrnutí 51](#_Toc177021765)

[2.3 Akademické práce a články zabývající se podobnou tématikou 51](#_Toc177021766)

[2.3.1 Real-Times News Recommender System 51](#_Toc177021767)

[2.3.2 News recommender system: a review of recent progress, challenges, and opportunities 52](#_Toc177021768)

[2.3.3 Offline and online evaluation of news recommender systems at swissinfo.ch 52](#_Toc177021769)

[2.3.4 User trends modeling for a content-based recommender systém 53](#_Toc177021770)

[2.3.5 Hybridní doporučující systém pro doporučování filmů s použitím expertního systému 53](#_Toc177021771)

[2.3.6 Multimodální doporučující systém 54](#_Toc177021772)

[2.3.7 Doporučování na Amazon.com (Item to item collaborative filtering) 54](#_Toc177021773)

[2.4 Knihy zabývající se doporučujícími systémy 55](#_Toc177021774)

[2.4.1 Practical recommender systems 55](#_Toc177021775)

[2.4.2 Algoritmy inteligentního webu 55](#_Toc177021776)

[2.5 Otázka autorských práv u doporučujících systémů s články 55](#_Toc177021777)

[2.6 Netflix 56](#_Toc177021778)

[3 návrh systému 59](#_Toc177021779)

[3.1 Optimalizace návrhu 60](#_Toc177021780)

[3.1.1 Optimalizace využívání paměti, optimalizace metod, nativní metody 60](#_Toc177021781)

[3.1.2 Využití databázových indexů 60](#_Toc177021782)

[3.1.3 Metody s podporou distribuovaného nebo paralelního zpracování 61](#_Toc177021783)

[3.1.4 Využití Redis 63](#_Toc177021784)

[3.1.5 Mikro servisní architektura 63](#_Toc177021785)

[3.1.6 Zasílání zpráv 64](#_Toc177021786)

[3.1.7 Kombinace online a offline přístupu doporučování jako cesta k lepšímu uživatelskému zážitku 65](#_Toc177021787)

[3.1.8 AJAX technologie 67](#_Toc177021788)

[3.1.9 Využití existujících redakčních systémů 68](#_Toc177021789)

[3.1.10 Využití PostgreSQL 68](#_Toc177021790)

[3.2 Use Case 68](#_Toc177021791)

[3.3 Architektura 69](#_Toc177021792)

[4 implementace systému 71](#_Toc177021793)

[4.1 Sběr dat 71](#_Toc177021794)

[4.1.1 RSS 71](#_Toc177021795)

[4.1.2 HTML 72](#_Toc177021796)

[4.1.3 Výstupy z Wikipedie 72](#_Toc177021797)

[4.2 Doporučení založená na uživatelích 72](#_Toc177021798)

[4.2.1 Uživatelská hodnocení 72](#_Toc177021799)

[4.2.2 Kolaborativní filtrování a SVD algoritmus 73](#_Toc177021800)

[4.2.3 Doporučení na základě klíčových slov 75](#_Toc177021801)

[4.2.4 Doporučení na základě oblíbených kategorií 75](#_Toc177021802)

[4.3 Doporučení založená na obsahu 75](#_Toc177021803)

[4.3.1 Datové sady 77](#_Toc177021804)

[4.3.2 Využívání výstupů z Wikipedie 79](#_Toc177021805)

[4.3.3 Předzpracování textu 80](#_Toc177021806)

[4.3.4 Podobnost slov a dokumentů 85](#_Toc177021807)

[4.3.5 Knihovna Scikit-Learn 87](#_Toc177021808)

[4.3.6 Knihovna Gensim 87](#_Toc177021809)

[4.3.7 Doporučení na základě četnosti a inverzní četnosti termínů 88](#_Toc177021810)

[4.3.8 Doporučení na základě podobnosti vektorů slov 91](#_Toc177021811)

[4.3.9 Doporučení na základě podobnosti vektorů slov a dokumentů 96](#_Toc177021812)

[4.3.10 Doporučení na základě společných témat 103](#_Toc177021813)

[4.3.11 Skládání výsledků 104](#_Toc177021814)

[4.4 Hybridní přístup 104](#_Toc177021815)

[4.4.1 Fuzzy hybridní přístup 106](#_Toc177021816)

[4.5 Klasifikační metody 110](#_Toc177021817)

[5 PRAKTICKÉ OVĚŘENÍ 112](#_Toc177021818)

[5.1 Ukázka a práce se systémem 112](#_Toc177021819)

[5.2 Tok uživatelských událostí a automatizované testování 113](#_Toc177021820)

[5.3 Validace a optimalizace obsahového přístupu 114](#_Toc177021821)

[5.3.1 Optimalizace výkonu TF-IDF 114](#_Toc177021822)

[5.3.2 Evaluace TF-IDF a dalších metod 120](#_Toc177021823)

[5.3.3 Evaluace Word2Vec 127](#_Toc177021824)

[5.3.4 Evaluace Doc2Vec 137](#_Toc177021825)

[5.3.5 Evaluace a optimalizace LDA 140](#_Toc177021826)

[5.4 Uživatelské testování 149](#_Toc177021827)

[5.4.1 Výsledky uživatelského testování a srovnání obsahových metod 150](#_Toc177021828)

[5.4.1 Výsledky uživatelského testování a srovnání uživatelských metod 151](#_Toc177021829)

[5.5 Evaluace klasifikátoru 152](#_Toc177021830)

[5.6 Možné problémy doporučujících metod 153](#_Toc177021831)

[6 ZHODNOCENÍ 155](#_Toc177021832)

[6.1 Otevřené záležitosti 155](#_Toc177021833)

[ZÁVĚR 157](#_Toc177021834)

[RESUMÉ 158](#_Toc177021835)

[SUMMARY 159](#_Toc177021836)

[SEZNAM POUŽITÉ LITERATURY 160](#_Toc177021837)

[SEZNAM POUŽITÝCH POJMŮ 179](#_Toc177021838)

[SEZNAM POUŽITÝCH SYMBOLŮ 182](#_Toc177021839)

[SEZNAM OBRÁZKŮ 184](#_Toc177021840)

[SEZNAM TABULEK 186](#_Toc177021841)

[SEZNAM PŘÍLOH 187](#_Toc177021842)

ÚVOD

Doporučovací systémy jsou v současné době možná vůbec nejvýznamnější aplikací umělé inteligence. Do životů (zejména mladých lidí, ale zdaleka nejen těch) zasahují skrze sociální sítě, internetové i kamenné obchody, služby pro sledování filmů nebo poslech hudby, či služby pro výběry zájezdů, hotelů nebo restaurací. Dokonce dnes zasahují i do našich nejintimnějších částí života skrze různé aplikace pro sjednávání rande online. Tento text slouží jako prakticky zaměřené nahlédnutí pod pokličku takovýchto systémů. Text by mohl dobře posloužit zájemcům o datovou vědu a umělou inteligenci, programátorům, manažerům IT projektů týkajících se tohoto tématu, ale snad i všem lidem z netechnologického prostředí, kteří se chtějí dozvědět více o systémech, které bez velkého přehánění, změnily tento svět a chování lidí.

AUTOR

Absolvent Informačních systémů a Aplikované informatiky na Ostravské univerzitě a SPŠ veFrenštátě p. R. Věnuje se webovým systémům a umělé inteligenci v akademické i firemní sféře. V minulosti publikoval také netechnologické internetové články o hudbě a kultuře.

PŘEDMLUVA

Co jsou to doporučovací systémy?

Představte si, že se na chvíli vrátíte do 90. let, či alespoň začátků tohoto století. Díváte se na televizi a zrovna přijde přerušení vašeho oblíbeného programu. Přichází blok reklam. Pokud jste tu dobu ještě stihli zažít před internetem, třeba to ty z vás s větší fantazií někdy dokonce napadlo. Jaké by to asi bylo, pokud by se místo reklam, které jsou zobrazovány všem sledujícím daného TV kanálu, někdo napojil na váš televizor a nějakým způsobem zjistil, co vás nejvíce zajímá?

Pro obchodníka by to bylo asi darem z nebes. Snem každého správného obchodníka je přece maximalizovat svůj zisk! Takřka ideálním nástrojem pro maximalizaci zisku se tedy jeví nástroj, pomocí něhož lze dokonale předpovídat potřeby zákazníka i odhalovat jeho skryté tužby.

Co ale „já“ jakožto divák? Vždyť někoho přece také nebaví dívat se na reklamy o „žrádlu“ pro pejsky, když má doma jen kočku. Nebo bydlí v „paneláku“ a nevlastní ani zahradu, přesto mu opakovaně ukazují reklamu na sekačky a bazény. Samotný divák/uživatel/konzument tedy z doporučovacího systému může také těžit. Tedy do té doby, než si možná začnete klást otázky jako: „Jak vlastně ‚oni‘ zjistí, co se mi líbí?“ nebo „Co o mně všechno ví?“ Jak se často ukazuje, z pohledu uživatele můžete doporučovací systémy milovat i nenávidět (nebo obojí zároveň). Podobný model interakce obrazovky se zákazníkem by byla vlastně jedna z nejlepších a zároveň nejhorších věcí, která by se vám jako zákazníkovi/uživateli mohla stát.

Když vidíme sociální sítě, internetové obchody, či aplikace pro sledování videí, seriálů a filmů nebo poslech hudby, nejsme této futuristické představě daleko. Vidíme jako první samotný obsah a příliš nepřemýšlíme na tom, proč tuto položku vidíme jako první a tu jinou jako čtvrtou. Zaměřujeme se hlavně na samotné fotografie z dovolených, psané příspěvky na sociálních sítích, videa našeho oblíbeného tvůrce videoobsahu, náš oblíbený seriál, produkty k nakoupení, nebo dokonce zpravodajské články. Teprve ve chvíli, kdy se nám na obrazovce objeví něco nečekaného – například očividná reklama, pro nás nerelevantní obsah, překvapivý, či odpudivý internetový obsah, který vidět nechceme, si uvědomíme, že před nás „něco“ musí tento obsah předhazovat. Dá se tedy říct, že teprve ve chvíli, kdy algoritmy doporučovacích systémů selžou a nabídnou nám nesprávný obsah nebo naopak příliš podbízivý, se probudíme z našeho běžného brouzdání internetem a uvědomíme si, že na pozadí mnoha oblíbených internetových služeb existuje jistý skrytý mechanismus, který nám obsah vybírá a řadí.

Tento mechanismus je naprosto klíčovým prvkem mnoha internetových služeb a hnacím motorem jejich zisku a lákadlem pro uživatele. Tento skrytý mechanismus ať chceme, nebo ne, určuje například náš pohled na sortiment zboží daného obchodu, současnou hudbu, ale může výrazně také ovlivnit náš názor na následující volby.

Při zjednodušení problematiky lze poukázat na tyto motivace zákazníků (uživatelů) a obchodníků týkajících se doporučovacích systémů.

* Uživatel chce, aby se mu zobrazoval relevantní obsah (tedy obsah, který jej zajímá).
* Obchodník chce maximalizovat svůj zisk.

Pokud by tedy situace byla takto jednoduchá a motivace uživatelů a obchodníků se navzájem nevylučovaly, doporučovací systémy by byly ideálním nástrojem pro filtrování obsahu na internetu. Jako poctivý farmář s vidlemi by pak stály nad obsahem, přebírali jej a obsah „nejlepší“ pro nás by nám předhodili. Bohužel, pod výše zmíněnými dvěma zjednodušenými body se skrývá mnoho problémů.

Lidská psychologie je velmi komplexní fenomén a samotná otázka, jaký vlastně obsah chce uživatel vidět (jaký je pro nás „nejlepší“), je velmi složitá. Každý má možnost své zařízení odložit kdykoliv, přesto to tak jednoduché není. Pokud při dnešní pokročilé podobě internetu trávíte na počítači, či menších chytrých zařízeních déle než pár minut týdně, možná jste i dostali pocit viny z toho, že jste nestrávili čas nějak lépe. Je také dost možné, že tento pocit se opakuje pravidelně, a to i přestože jste si plně vědomi svého času, který zpětně vnímáte jako promarněný (tedy do chvíle, než opět zasednete k zařízení nebo jej vytáhnete z kapsy).

Nemusíme se ani pouštět do čtení psychologických vědeckých článků a knih, abychom si uvědomili, že zde existuje velký rozpor mezi obsahem, který bychom si přáli, abychom konzumovali a shledáme jej smysluplný (například obsah pomáhající našemu vzdělávání nebo zlepšování poměrů ve společnosti) a obsahem, kterému často dáváme přednost, ale nejsme na něj natolik hrdí (nechť si každý sáhne do svědomí a uvědomí si zde své slabiny).

Jedna věc je konzumovat určitý obsah za účelem relaxace pro účely odpočinku, či pobavení s přáteli (dejme tomu jít do kina na film). Takový obsah, byť nás třeba ničím aktivně neobohacuje, může být prospěšný jak pro naše zdraví, tak sociální vazby. Druhou věcí je konzumovat určitý obsah, protože se mu zkrátka nedokážeme ubránit. Tento obsah nás přitom často ničím neobohatí a někdy dokonce ještě navíc více unaví. Ač je to stále nová záležitost a výzkumy teprve probíhají, dnes je již zřejmé, že přílišná konzumace určitého internetového obsahu může způsobovat pokles produktivity, ztrátu motivace a v krajním případě být spouštěčem, či jedním z faktorů, přispívajícím vážným psychickým poruchám zejména mladých lidí.

Jedná se o rozkol, který mnozí pociťují při konzumaci návykových látek i gamblerství. Všichni například víme, že pro zlepšení nálady i zdraví, by bylo lepší si udělat pár kliků a dřepů, ale zatímco začít cvičit v kanceláři působí společensky nepřijatelně, odběhnout si na cigaretu, bereme jako součást naší společnosti, a dokonce jako jistý společenský rituál. Ostatně, jestliže je na internetu přístup takřka k neomezenému dosavadními lidskému poznání, jak je možné, že mezi námi nechodí více géniů než jindy v minulosti? Jednou z odpovědí může být konkurence v podobě zábavního obsahu, který je lákavější. A v tomto mají doporučovací systémy také své prsty.

V tomto textu si tak buďme všichni vědomi nebezpečí doporučovacích systémů. Avšak buďme teď možná ďáblovými advokáty. Samotné algoritmy „zlé“ nejsou. Jsou to jen chytře poskládané postupy a matematické operace určené k maximalizaci nějakého cíle. Jen zkrátka (zatím?) neznají koncept morálky a snaží se optimalizovat co nejlépe dokážou – podle toho, jak jim to lidé navrhnou. Doporučovací systémy potřebujeme. Když si uvědomíme, že množství aktuálních informací na internetu neustále exponenciálně stoupá, je zřejmé, že již dávno nelze informace na internetu zobrazovat pouze pomocí jednoho určitého kritéria jako například datum vydání a nestačí to ani v omezeném měřítku nějakého webu nebo například v množině příspěvků přátel a sledujících na sociální síti. Z příliš primitivního typu zobrazování informací by tedy neměl velký užitek ani daná společnost nabízející obsah, ani uživatel. Jeden můj kamarád řekl, že doporučovací systémy jsou „nutné zlo“. Jedná se o dnes nutný filtr informací. Nemůžeme tedy tyto systémy vnímat rovnou jen jako něco, co nám přišlo okrádat peněženky (v lepším případě) nebo nás učinit závislými na technologiích, či dokonce politicky zradikalizovat. Stinnou stránku těchto systémů je nicméně potřeba mít vždy na paměti, a pokud si myslíte, že to jen trochu dokážete, je moudré se o těchto systémech něco dozvědět. Ať už je vaší motivací finanční zisk, nebo obavy z budoucnosti.

A nejvíce se o takových systémech dozvíte, když jeden takový vytvoříte. Pojďte se tedy v tomto textu nejprve podívat na základy umělé inteligence pro sjednocení základních znalostí (popř. tuto část přeskočte, pokud o tomto tématu již něco víte) a pokračujme k přehledu současných doporučovacích systémů a praktické části vytváření doporučovacího systému.

Patřím spíše mezi lidi praxe než teorie, avšak praxe bez teorie fungovat nemůže a často ukáže člověku nové možnosti a donutí jej přemýšlet o řešení problémů jinak. Součástí textu je tak stručný, neformální úvod do umělé inteligence, ve kterém jsou uvedeny nejnutnější informace nutné k pochopení problematiky. Můžete si tak získat základní přehled k tomu, jak výzkumníci a vývojáři došli nejen k doporučovacím systémům, ale i dalším současným aplikacím umělé inteligence jako chatbotům (např. tom představeném společností OpenAI v rámci jejich ChatGPT).

Protože existuje mnoho možných přístupů, metod a algoritmů, tento text se zaměřuje pouze na zlomek z nich, a to však pomocí projektu, který nepatří mezi nejjednodušší na realizaci. Cílem bylo vybudovat doporučovací systém pro jednu konkrétní oblast, kde se doporučovací systémy také vyskytují a která je obzvláště citlivá na možné nežádoucí jevy plynoucí z možného špatného fungování, či dokonce zneužití takového doporučovacího systému – zpravodajské servery a novinové články. Začali jsme přitom úplně od nuly bez jakýchkoliv dat.

Všichni, kdo umí číst kód si mohou kromě popisu zde prohlédnout použité algoritmy přímo napsané v jazyku Python v GitHub repositáři <https://github.com/patmull/recommender-system-for-news>.

Pozn. netechnologická, č. 1: V názvu a nadpisech se vyskytuje slovní spojení „doporučovací systémy“. Pravopisně korektnější je možná spíše název „doporučující systémy“. V češtině se však více uchytilo spojení s tvarem „doporučovací“, spíše tedy zde bude používáno toto slovní spojení.

Pozn. netechnologická, č. 2: V textu se vyskytují názvy knih, odborných a vědeckých článků, programovacích knihoven, jazyků, webových stránek, aplikací a služeb, implementací metod apod. Tyto názvy jsou většinou uvedeny *kurzívou.* Pro odlišení metod a implementací metod v programovacích knihovnách však jsou názvy metod ponechány standardním písmem. Výrazy a výstupy programovacího jazyka včetně útržků kódů jsou poté odděleny rozdílným fontem.

1. Neformální úvod do umělé inteligence

Doporučovací systémy jsou součástí umělé inteligence a více specifické metody uvedené dále nelze chápat, jestliže čtenář nemá alespoň základní přehled o metodách, které z velké části vycházejí ze statistiky, lineární algebry nebo logiky. Alespoň základní přehled o významných metodách se snad bude hodit jak běžným lidem (jakožto uživatelům různých aplikací využívající tyto metody nebo jejich základy), tak programátorům. Protože je toto téma na samostatnou tučnou knihu (resp. několik knih), je tato sekce pojata velmi neformálně. Tato sekce obsahuje shrnutí mého pohledu na oblast umělé inteligence nejkratším možným způsobem, jakým bych předal informace kamarádovi/kamarádce, kteří neabsolvovali kurzy zaměřené na umělé inteligence nebo vyšší matematiky. Míra abstrakce bude tedy vysoká a nebude zde vysvětlováno nic příliš detailně, aby velikost této sekce nepřesáhla únosnou mez a odborných slov bude použito co nejméně. Kde to jde, bylo snahou namísto dlouhých popisů nebo rovnic, přidat náčrtky, které by měly být (snad) alespoň zčásti srozumitelné i běžné populaci. Jedná se tedy o pokus představit toto téma komukoliv, a to v nejkratší a nejjednodušší možné formě (a snad i s trochou humoru). Za případné faktické chyby se tedy především znalejším čtenářům předem omlouvám.

Následující sekce bude v nejhorším případě příliš odborná pro laiky a příliš žoviální pro znalé, avšak vzhledem k popularizaci tohoto tématu, dle mého názoru, stojí za to tohle riziko podstoupit.

* 1. Co je to umělá inteligence?

Odpovědět na to, co je to „skutečnou“ umělou inteligencí, je spíše podnět k filozofické debatě, pojďme se ale podívat na historické a současné vnímání umělé inteligence. Jeden ze zakladatelů počítačové vědy Alan Turing představil svůj slavný *Turingův test*, který říká, že inteligence se dá měřit pomocí času, který trvá lidskému operátorovi uhádnout, zda komunikuje s člověkem nebo počítačem. Čím delší je tento čas, tím má být stroj více inteligentní. Z pohledu dnešního vývoje umělé inteligence však vidíme, že tento test není úplně užitečný. Umělá inteligence v některých ohledech lidi již daleko převyšuje a je schopna v mnoha případech poskytnou odpovědi daleko rychleji a přesněji. Na druhou stranu stále selhává v některých oblastech lidského uvažování jako intuice nebo kreativita. Ač v poslední době vidíme zajímavé pokusy, kdy například umělá inteligence umí složit píseň nebo vytvořit obraz, tyto výtvory stále nejsou natolik ceněné (a snad i nikdy nebudou natolik ceněné) jako výtvory lidí.

Obrázek 1: Načrtnutí principu Turingova testu. Turing chápal strojovou inteligenci jako doba uhádnutí toho, zda komunikujete přes počítač s člověkem s počítačem nebo pouze s počítačem. Čím déle toto trvá uhádnout, tím větší měla být inteligence počítače.

Stejně tak umělá inteligence pohledem Karla Čapka a jeho (resp. bratrovi Josefovi) světoznámí *roboti*, kteří byli nakonec k nerozeznání od lidí, ve srovnání s dnešním vývojem tolik neobstojí. Roboti sice existují a již dnes jsou v některých případech až děsivě podobní lidem. Přesto se nejeví, že by byl masový zájem o to nahradit manžely a manželky robotami a robotkami.

Otázkou je samozřejmě, co nastane, až dojde k dalším vylepšováním. Vývoj, ale zatím zdá se, jde spíše tím směrem. Vypadá to, že chceme, aby umělá inteligence byla dobrá v určitých činnostech, které lidem tolik nejdou anebo se jim z nějakého důvodu nechtějí dělat. Máme sice již dnes robotické číšníky/servírky, ale ti/ty slouží ve většině restaurací jen k dovozu/odvozu občerstvení a restauratéři si je (až na pár restaurací, např. v Japonsku) spíše objednávají jako kuriozitu s trochou příslibu ulehčení práce lidským servírkám. Avšak nevypadá to zatím, že by lidští pracovníci v tomto odvětví ztráceli pozice kvůli robotické konkurenci. Stejně tak dnešní chatboty jako ChatGPT nebo Claude zvládají velmi dobře mnoho programátorských úloh, avšak nevidíme, že by byli programátoři masově nahrazováni. Spíše se jim naopak mohla ještě zvýšit porce práce, protože najednou jsou schopni sestavit programy složitější než jindy a rychleji. Kde jsme naopak mohli vidět úbytek, je například u poptávky lidí provádějící jednodušší jazykové služby na internetu s menší kvalitou – např. plnění webových stránek textovým obsahem, kdy nezáleželo tolik na kvalitě a originalitě. Je nám ale skutečně líto, že stejně už tak vlezlé, méně kvalitní marketingové texty, nejsou vygenerovány nějakým studentem marketingu, ale místo toho chatbotem? Alespoň se daný student toužící po přivýdělku bude více snažit, aby dodal do svých textů něco více než umělá inteligence, která zatím tvoří texty sice často překvapivě dobré, ale také poněkud nudné a nekreativní. Pokrok v této oblasti se tedy nevyvíjí tak, že bychom chtěli vytvořit z robotů nové lidi (jak to popisoval Čapek a vlastně i Turing) se všemi jejich dokonalostmi a nedokonalostmi. Spíše si nyní vyvíjíme nástroje, které dokážou nahradit věci, které se nám nechtějí dělat nebo zdokonalit věci, ve kterých nevynikáme tak jako umělá inteligence, či v těchto činnostech nejsme tak efektivní. Nehrozí tedy zatím natolik, že by nás nahradily stroje podobné nám, ale spíše, že bude umělá inteligence jakousi kouzelnou hůlkou, která bude umět vytvářet věci, které budou umět jiné lidi ohromit (v tom lepším případě) anebo ošálit (v horším případě). Již v roce 2024 je poměrně těžké rozeznat, zda je fotografie nebo zvuková nahrávka hlasu pravá či nikoliv. Toto nemusí být vůbec dobré, pokud někdo použije něco podobného, např. tak, že se bude snažit vygenerovat hlas podobný hlasu politiků nebo si nechá vytvořit video umělého teroristického útoku s vidinou poštvat společnosti proti určité skupině lidí. Jednou z takových „magických“ schopností dnešní umělé inteligence je také například schopnost řadit příspěvky na sociálních sítích.

Hranice umělé inteligence z historického hlediska tedy nejsou snadno určitelné. V roce 2024 se však můžete s tím, co se většinou myslí pod pojmem „umělá inteligence“, setkat zejména v těchto oblastech:

* Vyhledávače,
* doporučovací systémy (sociální sítě, internetové video nebo audio platformy apod.),
* překladače,
* hlasoví asistenti,
* videohry,
* automatické rozpoznávání obličejů a objektů na fotografiích (sociální sítě, vyhledávače),
* chatboty a generativní nástroje (pro generování nebo úpravu textu, či obrázků).

Problémy, které se firmy, univerzity nebo instituce snaží pomocí umělé inteligence řešit jsou ve zkratce tyto:

* **Logické odvozování**. Zejména v 80. a 90. letech byla práce v této oblasti zaměřena na tom, že počítač dokázal vykonat krok po kroku logickou dedukci (skoro jako digitální Sherlock Holmes). Problém tohoto přístupu byl, že ani lidé většinou nepoužívají k řešení problémů návod, kterého se drží krok po kroku, ale využívají také například intuici, kterou tento přístup nebyl schopen nabídnout. Podobná dedukce krok po kroku je pro počítače také velmi pomalá, pokud se pravidel objeví příliš mnoho. Logika také pracuje s něčím, co je jasně dané. Ale co je v našem složitém světě jasně dané? Že následujícího roku bude určitě nějaký den, kdy bude vidět na obloze Slunce? Pokud by dinosauři uměli myslet jako my a měli internet nebo televizi, asi by je na druhé straně planety nemile překvapila informace o tom, že se do mexického zálivu zřítil obrovský asteroid a na stovky let žádné slunce na obloze neuvidí kvůli množství impaktního prachu v atmosféře a požárům vzniklým horkým pulsem po dopadu. Později tak logiku doplnily koncepty nejistoty zachyceny pomocí pravděpodobností, že nastane (nebo nenastane) nějaká událost a vznikl také nový druh logiky – fuzzy logika, která umí pracovat s konceptem vágnosti definic (např. „velmi horká voda“ nebo „lehce zabrzdit“).
* **Znalostní inženýrství**. Tato oblast se zaměřovala na uchování dostatečně velké znalostní báze, která měla sloužit jako robustní základ znalostí pro následné odvozování. Znalosti tedy nemusely být uchovány pomocí vět logiky, ale mohly být uchovány například pomocí ontologií, ve kterých na sebe navazují objekty a tvoří také hierarchie. Dnes s nástupem neuronových sítí není tento přístup již úplně běžný, ale stále je součástí některých podnikových aplikací nebo jednoho nejmenovaného, zajímavého, ale pro mnohé odborníky kontroverzního projektu, snažícího se pomocí budování ohromné znalostní báze (propojené pomocí sítě znalostí), vybudovat uměle inteligentní software zahrnující, co největší množství znalosti o tomto světě, popřípadě o konkrétní firmě nebo instituci za účelem dalšího odvození znalostí nebo řešení problémů. Někteří se domnívají, že právě toto je tou pravou cestou k vyvinutí tzv. obecné umělé inteligence, která se nebude zaměřovat pouze na konkrétní úkol, ale bude umět řešit problémy z mnoha různých odvětví a snad i řešit problémy příliš složité, které my lidé dlouhodobě řešit nedokážeme.
* **Plánování a rozhodování**. Zatímco předchozí zmíněné metody jsou zjednodušeně jakousi automatizací předchozích známých matematických, resp. logických metod, které často znali již dávno v minulosti, vytvoření konceptu „agenta“, který je vypuštěn do nějakého prostředí a dokáže se mu přizpůsobit, je již něco, co evokuje označení „umělá inteligence“ přece jen o něco více. Můžete si zde představit jednoduché roboty, kteří se snaží vyhýbat překážkám. Mají některé preference situací, do kterých se snaží dostat, a naopak se snaží vyhýbat jiným situacím s ohledem na dosažení cíle. V základních modelech tzv. *Markovových procesů* (nebo „řetězců“) pracujeme s pravděpodobností, že nějaká akce změní určitým způsobem stav a také s funkcí odměn. Každý stav má určitou užitečnost, ale akce pro dosažení stavu něco stojí. V každém stavu se může agent rozhodnout kam dále.  
  Tento přístup začne být zajímavý ještě více, jakmile je zde takových agentů více. Na těchto základních myšlenkách staví počítačové hry s různými postavami, které se samostatně pohybují po prostoru. Mnoho starých počítačových her je ostatně založených přesně na principech popisovaných výše.
* **Učení**: V minulém přístupu jsme byli již blízko konceptu strojového učení. Jestliže dodáme agentovi možnost učit se ze svých chyb a byl vymyšlen způsob (formální matematický model), jak toto zachytit (pomocí Markovových rozhodovacích procesů), vědci měli možnost vyvinout optimální projekci („mapování“) stavů agenta (robota) na akce, pro to, aby se agent správně rozhodl v každém momentu, kam dále. Metody, které toto umožňují, jsou již tím slavným spojením „strojové učení“, které se dnes tak často řeší i mimo počítačovou vědu. Máme ale také další druhy strojového učení (bude rozebráno později).  
  Vzhledem k vnímání konceptu „učení“ výše, někteří zašli dále a dokonce chápou pojem „umělá inteligenci“ v ve vyšší míře abstrakce od detailů jako jakousi (často velmi složitou) funkci, která se snaží na základě vstupů odhadnout výstup tak, aby pokud přijdou vstupy nové, dříve vůbec neviděné, byla tato funkce schopna zhruba odhadnout požadovanou hodnotu výstupy. Pokud pak máme alespoň některé vzorová data, u kterých víme, že hodnota výstupu pro daný vstup je správní (např. pokud jsme to v minulosti pozorovali a naměřili anebo nám to někdo ručně vyplnil dle jeho perfektních znalostí dané problematiky), můžeme tato data využít k tomu, abychom následně, pokud již budeme mít pouze vstup, ale výstup nikoliv, dokázali „správný“ výstup odhadnout z existujících dvojic vstup-výstup.

Obrázek 2: Umělá inteligence z matematického pohledu. Vnímání umělé inteligence jako aproximace funkce je možná značné zjednodušení problematiky. Zejména u učení s učitelem je však tento pohled na problematiku výstižný.

Představte si například, že máme data ohledně minulých sportovních výsledků týmů. Máme např. hodnoty jako počet zraněných hráčů v daném zápase, průměrnou kupní hodnotu hráčů týmu, minulý zápas, aktuální formu týmu apod. Teoreticky, pokud budeme mít dostatek takových údajů a tyto údaje budou dostatečně kvalitní a nebudou obsahovat různý šum nebo odlehlé hodnoty, můžeme sestrojit takový model umělé inteligence, který se na základě dat minulých, naučí data budoucí. Tomuto druhu strojového učení se říká *supervizované učení* (nebo také *učení s učitelem*, protože zde máme něco jako „učitele“, který zná správnou hodnotu).   
Pokud by byl schopen někdo vyvinout takovouto dokonalou funkci pro odhadování sportovních sázek nebo například akcií na základě různých faktorů, byl by jistě okamžitě miliardářem. Realita však (asi naštěstí) tak jednoduchá není.   
Ač to nyní může evokovat, že umělá inteligence je o nějakém (nepřesném) předpovídání budoucnosti, není tomu zdaleka jen tak. Někdy máme cíl ostatně jiný. Nesnažíme se něco předpovědět, ale snažíme se něco klasifikovat do určité skupiny. Někdy navíc ani správné hodnoty nemáme. Nemáme „učitele“ a nezbývá nám tedy nic jiného než být „samouky“. Podobně jako v případě agenta „očichávající“ a učící se nový prostor, do kterého byl vypuštěn výše. Příkladu s agentem výše se říká odborně „posilované učení“, protože agent je odměněn za to, když je k cíli (který je na rozdíl od prostředí známý) blíže. Toto posilování je samozřejmě nutné také popsat matematicky, a tak tvoří nějakou matematickou funkci. V jiném případě se však bez znalosti výstupů coby „správných“ hodnot spíše snažíme v datech hledat různé vzory a shluky, abychom porozuměli, jak spolu data souvisejí (například které objekty jsou si podobné).   
Některé dnes používané metody, například z oblasti zpracování přirozeného jazyka, nejsou snadno zařaditelné do kategorie supervizované / nesupervizované a tvoří spíše samostatnou kategorii mezi těmito typy učení. Někdy se uvádí, že existuje také kategorie *polosupervizovaného* učení nebo *samosupervizovaného* učení. Rozdíl lze chápat takto: zatímco nesupervizované učení (bez učitele), jak název napovídá koncept „supervizování“, tedy správných hodnot, vůbec nezná, samosupervizované učení tento koncept aktivně využívá, ale nepoužívá k tomu správné, „oštítkované“ hodnoty výstupů (např. zkrátka protože nevíme, co má být výstupem nebo nemáme finanční prostředky nebo dostatek odborníků, aby byla data „oštítkovaná“). Správné štítky si tedy takové samosupervizované metody snaží určit „svépomocí“ přímo ze vstupů. Často je tomu využíváno právě u textů, protože ač nemáme žádnou správnou hodnotu danou předem, na které bychom mohli natrénovat, můžeme se například podívat na to, v jakém kontextu se nachází slova ve větách. Je tak neoddiskutovatelné, že jsou toto také ty „správné“ hodnoty – stačí tento koncept okolí slov ve větách vhodně uchopit matematicky. Ale nejsou to přitom hodnoty výstupní. Pokud pak poskytneme dostatek takových vět, může model zjistit pouze ze samotných dodaných textů, že je daleko větší pravděpodobnost, že se slovo „pes“ vyskytne v okolí slov jako „kočka“, „vodítko“, „štěkání“ než v okolí slov jako např.: „banka“, „úřad“ nebo „byrokaracie“. Stejně jako v učení bez učitelem tedy scházejí správné (štítkové, cílové) hodnoty, ale snažíme se takové hodnoty aktivně zjistit z dosavadních vět a poté je postupně při trénování využívat na nových a nových větách. Štítek pro samosupervizované učení je tedy například v případě metody pracujíc s převodem textu na vektory *Word2Vec* středové slovo ve větě předpovězené z kontextu slova. To většinou ale není samotným cílem toho, když nějaká forma nebo jednotlivec chce využít tenhle model v praxi. Konečným cílem je využít takového modelu například k tomu, aby navrátil podobné dokumenty nebo podobná slova na základě zadaných, či model uměl odpovídat na otázky.  
U polosupervizovaného učení, které je také někdy nazýváno jako „slabé“ učení pak hovoříme o případu, kdy máme k dispozici správné hodnoty výstupů *y* pouze u některých dat a jindy zase ne (buď jich máme málo, anebo máme intuici, že záměrné opomenutí některých výstupních hodnot nám může pomoci).   
A aby to nebylo ještě dostatečně zamotané, některé z dnešních úspěšných metod strojového učení také využívají kombinace výše zmíněných metod (protože to zkrátka funguje nejlépe). Nejprve je například využito samosupervizovaného učení a poté shlukování. Jestliže se ale níže budeme bavit o metodách redukce dimenzí nebo shlukování, mělo by být (alespoň v případě nejznámějších metod) jisté, že se bude jednat o nesupervizované učení (bez učitele).

* **Zpracování přirozeného jazyka**. Ze zkušenosti z mého okolí můžu tvrdit, že mnoho programátorů nepatří zrovna mezi příznivce literatury a češtiny. Počítače jsouvšak s jazykem spojené neodmyslitelně. Programovací jazyky dávají schopnost programátorům dávat počítači instrukce, které nejsou zase tolik rozdílné od běžného lidského (přirozeného) jazyka. Obzvláště mezi lety 2021-2023, kdy došlo k výraznému pokroku v rámci velkých jazykových modelů, majících schopnost generovat text na základě požadavků uživatele se ukazuje, že je možné plynule přejít od lidského jazyka skrze jazyk programovací až ke strojovému kódu a bitům (resp. bajtům), coby elektrickým impulsům v rámci hardwarových součástek počítače nebo optického signálu internetových kabelů. Spojení počítačů a jazyka tak zaujalo již v průběhu historie i lingvisty (a jiné jazykové vědce), kteří rovněž přispěli svým pohledem k pochopení toho, co „jazyk“ znamená z pohledu počítačové vědy a jak můžeme využít strojového učení pro vytvoření modelů, které budou umět pracovat s jazykem. Všiml si toho ostatně také známý filozof a lingvista Noah Chomsky, který významně přispěl také do teorie jazyků počítačové vědy. Jednou z přelomových myšlenek v oblasti zpracování přirozeného jazyka (ke které přispěl výrazně také například absolvent ČVUT Tomáš Mikolov) bylo nalezení různých metod pro převedení textu na vektory. Jakmile je text převeden na vektory, je možné měřit úhel vektorů nebo vykonávat s vektory matematické (resp. algebraické) operace. Zatímco hledání podobnosti textového dokumentu je těžko uchopitelné, hledání podobnosti vektorů se matematika, statistika a lineární algebra zabývá dlouhá léta. Pokud je tato vektorová reprezentace textového dokumentu správná (například článku), můžete pak například hledat podobné dokumenty.

V poslední době se můžete také setkat s dalšími příbuznými označeními pro problémy představené výše či níže. Umělá inteligence je samozřejmě všechny nějakým způsobem zaštiťuje, ale dnes se často také mluví o dalších, zatím nezmíněných oblastech jako například datová věda, datová analýza, dolování dat. Každá z těchto oblastí má svou určitou specializaci, ale všechny tyto obory jsou si příbuzné. Stejně tak se například můžete setkat s názorem, že umělá inteligence je pouze aplikovaná statistika. Na příkladech výše lze ale vidět, že do umělé inteligence spadá mnohem více (např. i logika). Je však pravda, že dnešní umělá inteligence je skutečně tvořena z velké části statistickými přístupy.

* 1. Optimalizace

Abychom našli vhodné řešení problému, který se snažíme řešit pomocí strojového učení, je nutné optimalizovat. Co budeme optimalizovat, záleží na problému, který řešíme. Optimalizaci známe i z našich běžných životů. Mnoho moderních fitness aplikací se například snaží optimalizovat cvičební programy uživatelů. Ani tady to však není tolik jasné a uživatel si musí zvolit, zda chce například optimalizovat váhu nebo růst svalů, protože mít jako cíl obojí, je příliš obtížné. Pokud by se nám podařilo například vyvinout takový model, který by dokázal na základě zadané výšky, váhy, věku a požadovaného cíle optimalizovat tréninkový program a výživový plán, utrhli by nám ruce nejen jednotlivci toužící po lepší postavě, ale také bohaté sportovní týmy.

Znovu je ale nutné připomenout, že realita není tak jednoduchá. Aby to fungovalo alespoň dostatečně dobře, potřebovali bychom nejspíše velké množství údajů, neměli bychom jistoty, že je uživatel změří nebo zadá správně a je zde mnoho dalších problémů. Uživatel například může mít stresující práci nebo skrytou nemoc, kterou nelze odhalit, či genetické vlastnosti a predispozice ke sportu, které mají vliv na jeho cíle v oblasti fitness. Proměnných je zde zkrátka příliš mnoho a některé faktory nejsou snadno měřitelné.

Proto dnes mnoha uživatelům zůstane místo vysněných výsledků po využívání takovýchto aplikací jen díra v peněžence, a kdo chce pokračovat lepším způsobem, raději si najme osobního lidského trenéra z masa a kostí, který má sice své lidské chyby a možná některé ne úplně objektivní nebo správné názory, ale pořád je schopen se na základě své zkušenosti a intuice rozhodnout lépe než stroj.

Existují dnes však i v této oblasti slibné projekty, které jsou například využívané profesionálními fotbalovými kluby, které dokážou vyhodnotit různá data a doporučit následující postup tréninku, či nákupy hráčů do týmu a některé z takových systému přinášejí užitek skutečným trenérům pro optimalizaci tréninku sportovců.

Zde se mimochodem může jednat (příznačně) o dvojí výskyt optimalizace a tréninku. Taková aplikace pomáhající *optimalizovat* *tréninkový* plán, může obsahovat nějaký model strojového učení, který byl tzv. *natrénován* na nějakých minulých změřených datech sportovců. Tento model musí být *optimalizován* pro to, aby byl schopen dodávat vhodná cvičení s vhodnou zátěží ve správný čas.

Optimalizace je každopádně velmi těžký matematický problém (jak bylo naznačeno na příkladu výše). Lze si jí představit jako kráčení na horách v naprosté mlze, kdy se snažíte sejít do nejnižšího možného údolí, kde je nejtepleji (nebo nejvíce kyslíku).

Představte si, že se horolezec ztratí ve vysokých horách a operátoři záchranné služby mu sdělí, že vrtulník bude mít v takové nadmořské výšce problémy udržet, a jestli chce maximalizovat své šance na přežití i kvůli přísunu kyslíku, musí se dostat do nejnižšího možného údolí, jaké najde. Chce se dostat do údolí, jenže nevidí ani na krok. Jakmile však kráčí v mlze, ví, jestli aktuálně stoupá nebo klesá, a jak moc je kopec strmý. Když by se horolezec rozběhl a dlouho neměnil svůj směr, může se mu stát, že údolí prošvihne a už se nevrátí zpět. Jakmile půjde příliš pomalu a bude příliš přemýšlet nad tím, jestli nezměnit směr, může se stát, že vyhladoví nebo umrzne. Jakmile se do nějakého údolí konečně dostane, neví pak, jestli je to skutečně to nejnižší údolí anebo za dalším kopcem se nachází ještě nižší – více optimální pro přistání vrtulníku.

Naše hory v tomto případě jsou nějaká (často) složitá matematická funkce – tzv. *ztrátová* nebo *nákladová funkce*, kdy nejnižší bod znamená nejmenší náklad nebo ztrátu – což často znamená nejmenší chybu při odhadu správné předložené hodnoty při učení s učitelem. U dalších typů učení (např. bez učitele), je poté situace složitější, nechme to tedy stranou. Zatímco lidským okem bychom po vykreslení nějakého hezkého 3D hornatého modelu (nebo plastické mapy) byli schopni poměrně rychle určit, kde se nachází minimum, matematicky to tak jednoduché vůbec není. Neexistuje (alespoň tedy zatím) takový matematický nástroj, který by se podíval na údolí, ukázal prstem a řekl: zde je nejnižší bod. Při využívání matematických nástrojů, je dobrým nástrojem v kapse nikoliv kompas, ale *derivace*. Tu si lze představit jako výše popsanou znalost toho, jestli klesáme nebo stoupáme, a jak rychle. Poté můžeme využít například často využívané metody tzv. *gradientního sestupu*, která umí spolehlivě najít pomocí (někdy fixních, někdy proměnlivých velikostí kroků), kde se nachází nejnižší bod v dané „krajině“. Tato krajina však nemůže být příliš velká, a pokud máme až příliš velký prostor k prohledávání (to pravděpodobně znamená, že máme před sebou funkcí popsaný nějaký hodně složitý problém), často je dobré prohledávaný prostor prostě omezit a spokojit se s ne nejlepším, ale tzv. *heuristicky* optimálním řešením, tzn. že nesejdeme úplně do nejnižšího bodu v celém prostoru, ale pouze do údolí poblíž začátku hledání.

Obrázek 3: Naznačení gradientního sestupu do minima trojrozměrné funkce tvořící složitější povrch s více minimy a maximy [143].

Tento sestup nám tak nezaručí, že se nacházíme skutečně v tom nejnižším bodu. V rámci strojového učení se často musíme spokojit s tím, že jsme „snad v dostatečně nízkém“ (optimálním) bodu (a doufat, že helikoptéra to při naší záchraně zvládne i z neoptimální polohy, pokud se vrátíme k původnímu přirovnání). Pokud by funkce byla jen jednoduchá (jako například prostá funkce ve tvaru písmene „U“, určit minimum je jednoduché i matematicky, ale ne při hodně „pahorkaté“ funkci, a ještě k tomu ve čtyřech nebo více dimenzích.

Obrázek 4: Naznačení problému lokálních a globálních minim a maxim. Při optimalizaci často dojde k uváznutí v lokálním minimu.

Zajímavou oblastí, která přináší strategie, jak předejít některým z uvedených problémů jsou mimochodem evoluční algoritmy inspirované biologickými procesy (např. chováním hejn ptáků, mravenců nebo ryb). Tyto algoritmy jsou schopné často prohledávat prostor velmi efektivně. Jedná se o další inspiraci z přírody, pomocí které se daří řešit některé na první pohled těžké úlohy.

* 1. Lineární regrese

Lineární regrese se asi stěží dá označit za umělou inteligenci v tom slova smyslu, jak „AI“ chápe většina běžné populace. Je to však moment, kdy mnoha vysokoškolákům „docvakne“, že na základě předchozích dat se můžeme naučit nějaké zákonitosti mezi daty, což nám může dát do budoucna takřka „superschopnost“ předpovídat budoucí události. Moderní optikou strojového učení tedy můžeme označit lineární regresi za metodu učení s učitelem, kdy ke vstupům potřebujeme mít naměřené výstupy.

Tato metoda není natolik přesná a má velká omezení, ale někdy přece jen lze pomocí ní dosáhnout úspěchu, a to poměrně jednoduchým způsobem. Na pochopení principu této metody stačí bohatě základní škola. Na ose *y* máme tzv. *závislou veličinu*, na ose *x* máme proměnnou (ve složitějším případě více proměnných), která se pojmenovává také různými dalšími názvy jako „nezávislá proměnná“, „prediktor“, „regresor“, či „faktor“. Nejvýstižnější pro intuitivní pochopení je možná asi jedno z anglických pojmenování „explanatory“, tzn. „vysvětlující“ proměnná – protože vysvětluje, proč byly naměřeny takové hodnoty závislé proměnné (kterou můžeme nazvat také „vysvětlovanou“ nebo „predikovanou“) v rámci proměnné *y*. Pohledem funkcí si můžeme také lineární regresi představit jako funkci, která nám dostatečně dobře vyjádří vztah mezi vstupem *x* a výstupem *y.*

Skutečně větší sílu lineární regrese pak dostane, když použijeme více takových „vysvětlujících“ proměnných. Svět je totiž složité místo a použití pouze jedné takové proměnné často nestačí. Bohužel si poté již nelze tak dobře představit problém geometricky – naposledy tedy při přechodu do 3D prostoru si můžeme představit aproximaci dat místo přímky jako plochu. I proto se mimochodem používají metody pro redukce dimenzí později zde také popisované anebo se můžeme podívat zvlášť pouze na vztahy dvojic proměnných.

Cílem lineární regrese je na základě předchozích naměřených dat vytvořit takovou rovnici odpovídající v geometrickém pojetí rovnici přímky, aby byla minimalizována chyba, která je nejčastěji měřena pomocí čtverců, kdy strany čtverců odpovídají rozdílu mezi skutečnou hodnotou a hodnotou odhadovanou (v geometrickém pojetí hodnotou na přímce tvořící odhad vysvětlované hodnoty). Z původních diskrétních (nespojitých) hodnot tedy nyní máme spojitou přímku, která představuje hrubé odhady nových pozorování na základě dosavadních měření/pozorování.

Obrázek 5: Znázornění měření chyby u lineární regrese pomocí metody nejmenších čtverců pomocí internetového nástroje pro vizualizaci [144]. Přímka tvoří odhadované hodnoty a datové body jsou skutečnými naměřenými body. Rozdíl nám dává stranu čtverce. Součet těchto čtverců nám dává chybu modelu, kterou se snažíme minimalizovat (optimalizovat).

Pokud data lze zhruba protnout takovou přímkou, bavíme se o lineární korelaci. Bohužel, dvojice pozorování spolu často nějak souvisí, ale vztah mezi daty nelze tak jednoduše vyjádřit přímkou. Máme tedy korelaci, ale tzv. nelineární. Po vizualizaci bychom např. viděli různé prstence nebo shluky datových bodů – nějaká souvislost tedy zde očividně je a jako inteligentní lidé tuto souvislost chápeme, ale lineární regrese není rozhodně ten správný nástroj. Vědci a inženýři měli tak dlouho problémy takové závislosti dobře zachytit pro budoucí predikce. Nakonec se jim to však povedlo (viz dále).

Pokud se bavíme o různých metodách v rámci umělé inteligence nebo strojového učení, můžeme si je představit jako různé pracovní nástroje. Každý z nich se hodí na trochu jinou věc. Snaha o vyvinutí metod umělé inteligence je hnána právě tím, že ač lineární regrese někdy fungovala skvěle, nebyla dostatečně dobrým nástrojem pro mnoho problémů našeho světa.

* 1. Základní strojová inteligence

Pro lepší pochopení strojové inteligence (záměrně zde nazývaná jako „strojová“, pro rozlišení více obecných metod inteligence „umělé“ schopných řešit složitější, obecnější nebo různorodější problémy) je zde představen jeden z historicky významných programů známé jednoduché hry, na které bylo ukázáno, že stroj dokáže porážet lidi v piškvorkách.

* + 1. Piškvorky: ukázka primitivního přístupu k umělé inteligenci

Na této jednoduché hře, kterou znají všichni, si můžeme ukázat základní přístupy umělé inteligence, které byly používané v raných programech tohoto oboru.

Přístup 1: Můžeme jednoduše sestrojit pravidla, jak se má počítač chovat ve hře proti člověku. Můžeme tak vytvořit základní strategie, které mohou protivníka vyvést z míry. Např.: pokusíme se vždy označit dvě políčka po sobě nebo hned, jakmile bude mít protivník 2 políčka, zatarasit mu cestu na 3. políčku.

Přístup 2: Tenhle přístup je již „mazanější“. Využijeme toho, co umí počítač skvěle: uchovávat informace, zde konkrétně stav hry. Zatímco průměrný lidský mozek neumí tak dobře předpovědět mnoho následujících akcí, počítač můžeme nechat projít „nanečisto“ více různými variantami hry, a dokonce se pokusit předpovědět tahy protivníka.

Přístup 3: Sestavíme složitější model strojového učení. Na základě minulých her a možných nezdarů naučíme model reagovat lépe a lépe, až porážíme lidské hráče.

* 1. Logika a umělá inteligence

Trendem posledních let je vytvářet umělé neuronové sítě napodobující neuronové spojení lidského mozku. První takové modely byly přitom vyvinuty již v 60. a 70. let. Kvůli nedostatku výpočetního výkonu tehdejších počítačů, nedostatku dat, ještě ne tak vyvinutým programovacím jazykům a dalším důvodům, však tehdy ještě tyto metody neprorazily a nastalo období, kterému se někdy říká trefně „zima umělé inteligence“. „Žehlit“ reputaci strojové inteligence u veřejnosti tak musely přístupy založeny na logice, kterým se dnes paradoxně vývoj umělé inteligence již vyhýbá a berou se za zastaralé metody. Že nejsou tyto přístupy úplně kamarádi si měl možnost ostatně ověřit každý například i v chatbotu ChatGPT 3.5, který v matematice a logice zrovna neexceloval (musí se však uznat, že verze ChatGPT 4 nebo například chatbot Claude 3.5 si již vedou podstatně lépe, pořád ale v přesnosti zaostávají za daleko staršími programy řešícími logiku). Protože i na neuronových sítích vidíte, že trendy v umělé inteligenci se občas vracejí (naštěstí ne tak často jak je tomu v módě a kultuře), a protože někteří odborníci tvrdí, že bez logiky umělá inteligence možná nevyužívá dostatečně svůj potenciál, je dobré nevylučovat ani tuto oblast.

Základní oblast logiky tvoří *výroková logika* obsahující jednoduché logické spojky jako AND („a zároveň“), OR („nebo“), NOT („ne“, „není“) a implikace (dovolující definovat základní pravdy jako tzv. *modus ponens*: „Jestliže *A* implikuje *B* a *A* je pravda, pak *B* musí být také pravda.“). *Predikátová logika* pak dokáže vyjadřovat dokonce i vlastnosti a jednoduché věty lidského jazyka jako „Petr má rád historii“ nebo tzv. kvantifikátory jako „Každý Ital má rád pizzu.“ Nebo „Neexistuje Ital, který by měl rád pizzu s ananasem“. Na rozdíl od přirozeného (lidského) jazyka tyto věty však lze zapsat v podobě, jakou může počítač zpracovat, dát dohromady s ostatními zapsanými větami a z toho pak odvodit zajímavé nové znalosti a pokud je jich mnoho, lze z nich vytvořit stromové struktury, kdy jsou některé věty potomky rodičovských vět. V některých případech lze postupovat při odvozování nových znalostí jak dopředným způsobem (od faktů a dalších odvozených faktů až směrem k cíli), v některých ale i od cíle: máme nějaký cíl a snažíme se najít předpoklady, které jej potvrdí. Máme například symptomy nemoci (to je zde myšleno oním „cílem“) a snažíme se dobrat k příčinám nemoci.

Právě v medicíně se v 80. letech ukázaly jako možná první skutečně užitečné a v praxi použitelné využití umělé inteligence, tzv. *expertní systémy*. Ty byly nejčastěji založeny na „IF-THEN“ pravidlech (jak jim říkají programátoři) nebo modus ponens (jak by to nazvali matematici, resp. logici). Stovky a tisíce takových pravidel napsaných ve formě „jestliže je pravda toto, pak bude pravda toto“ se ukázaly jako (možná překvapivě) užitečné v poskytování doporučení k léčbě pacientů lékařům. Šlo tak zachytit cenné znalosti a zkušenosti lékařů, coby expertů do programů, které pak lékařům nebo přímo pacientům, kteří je ovládali, pomáhaly. Tedy do té doby, než prý jeden z takových systémů doporučil pacientovi předepsat kyblík penicilinu [145]. Jednoduchost těchto pravidel se ukázala jako největší síla, ale i slabost těchto programů. Tyto systémy nebyly schopné jakéhokoliv uvažování mimo tyto zachycené znalosti a jednalo se tedy o velmi primitivní formu umělé inteligence.

Když se ale pak do takových systémů zabudoval koncept tzv. *fuzzy logiky*, naskytly se zajímavé použití v rámci různých industriálních mechanických strojů, domácích spotřebičů, vlaků, tramvají nebo metra. V 60. letech si totiž Lofi Zadeh a ještě dříve Jan Łukasiewicz všimli, že klasická logika má podstatné omezení. Vnímá totiž, že něco buď pravda je anebo není. Toto binární vnímání světa je možná blízké počítačům, ale rozhodně nevyjadřuje dobře, jak funguje skutečný svět. Pokus vystihnout vágnost definice i řešení některých problémů však neskončila jen u konstrukce robustnější báze IF‑THEN pravidel, ale umožňuje také například aproximovat velmi složité, těžko předvídatelné funkce (například z oblasti časových řad jako chování burzy nebo počasí). Jestliže je tedy umělá inteligence o odhadu funkcí, obor fuzzy modelování toto zvládá často velmi dobře. Dnes asi nejúspěšnější metodou pocházející z fuzzy logiky jsou *fuzzy shlukovací metody*, které dokážou shlukovat podobné prvky bez ostrých hranic toho, co ještě je součástí stejné skupiny a co již není. Proč je dnes pole fuzzy logiky bráno jako spíše exotický a zastaralý obor je těžké odpovědět, ale než to svalovat na špatný marketing, pravděpodobně se může jednat o to, že spíše než potřeba zachytit „vágnost“, je potřeba zachytit v umělé inteligenci „nejistotu“ a k tomu fuzzy logika neslouží, ale slouží k tomu spíše statistika (pravděpodobnost, Bayesovská statistika apod.).

Výhodou při použití fuzzy logiky ve (fuzzy) expertních systémech bylo každopádně možné dosledování jakéhokoliv rozhodnutí programu. Takový „luxus“ dnes s neuronovými sítěmi, které něco rozhodují a generují bez našeho přesného chápání, fungující jako černé skříňky, které na základě vstupu navrátí nějaký výstup, bohužel nemáme a je to ožehavé téma současného a snad i budoucího výzkumu.

* 1. Pravděpodobnost

Jestliže jsme začali hovořit o fuzzy logice, pokusím se o základní popis rozdílu s pravděpodobností. Statisticky založená metoda by k problému předpovědi počasí a problému, zda: „Zítra nebude pršet“ asi přistoupila tak, že by se snažila identifikovat minulé dny, kdy také nepršelo. Součet pravděpodobností ale pořád počítá i s tím, že existují dny, ve kterých prše nebude a součet pravděpodobnosti, že bude nebo nebude pršet je přesně 1. Bereme zde také v potaz četnosti pozorování. „Fuzzy“ přístup by se oproti tomu snažil zjistit, zda zítřek bude zhruba takový jako dny, ve kterých nepršelo – tak jak je jako celek vnímá buďto meteorolog, nebo jak jsme to schopni zachytit z minulých pozorování. Měli bychom tedy například soubor pravidel jako: „Jestliže *atmosférický tlak* je *velmi nízký* a *rychlost větru* je *velmi vysoká*, pak *riziko bouřky* je *velmi* *vysoké.*“ Ohledně dnů, ve kterých neprší zde tedy ani nemusí být řeč (takový koncept zde neexistuje), stejně tak nebereme v potaz četnosti pozorování. Lze si tedy představit, že bychom mohli použít metody nejistoty ve chvíli, kdy bychom se snažili předpovědět počasí za 3 dny na základě dat počasí minulých dnů, ale koncept vágnosti („fuzzy“) v momentě, kdy bychom chtěli předpovědět počasí v následující hodině, a to na základě známých zákonitostí v počasí, které by nám například sdělil meteorolog. Nebo bychom tyto zákonitosti mohli vysledovat my sami z dat počasí minulých dnů.

Je potřeba se také rozhodnout, zda potřebujeme od programu definitivní odpověď nebo ne. Zatímco je nesmysl uvařené těstoviny definovat jako „uvařené na 85,6543 %“, ale raději byste chtěli vaření zarazit v momentě, pokud je máte rádi „měkčí“ nebo „tvrdší“, u samořídícího auta působí až černo-humorně definovat pravidla jako „zhruba člověk“.

Výše se bavíme o základním použití fuzzy logiky. Fuzzy modelování má, ale širší pole působnosti než jen zachycení „vágnosti“, které může zasahovat, a dokonce se nevylučovat a kombinovat s přístupem pravděpodobnostním (nejistoty).

Bayesovská statistika oproti tomu umožňuje vytvořit tzv. Bayesovké sítě. Do tabulky si můžeme nadefinovat pravděpodobnosti mezi 0 a 1 a definovat, do jaké míry bude mokrý trávník, pokud bude pršet a zavlažovat. Déšť má vliv na (automatický) zavlažovač. Dé.šť i zavlažovač má vliv na vlhkost trávy. Dnešní trendy v oboru dávají spíše přednost pravděpodobnostnímu přístupu, ale jak již bylo zmíněno s nadsázkou výše, i v tomto oboru se trendy někdy mění.

Obrázek 6: Znázornění principu Bayesovské statistiky.

* 1. Redukce dimenzí

Někteří odborníci tvrdí, že první „skutečně“ uměle inteligentní metodou bylo tzv. *PCA* (angl. *Principal Component Analysis*, česky *Analýza hlavních komponent*). Tato již poměrně stará metoda statistika otce moderní statistiky Karla Pearsona z roku 1901, je totiž schopná vzít mnohodimenzionální reprezentaci (tedy to, co si prakticky nedokážeme příliš dobře představit) dat a převést ji na méně dimenzí. Někteří tvrdí, že metody umožňující takto složitý a sofistikovaný úkone mají právo na to být označeny jako „inteligentní“. Praktické výhody to má následující: jak možná vytušíte, převod vyšších dimenzí na 3 nebo 2 dimenze nám dává možnost data zobrazit a pochopit a případně je prezentovat veřejnosti jako lidsky srozumitelné obrázky. Výhodné je to ale také například protože obraz dat v nižších dimenzích se dá lépe klasifikovat. I pro klasifikační metody strojového učení jsou příliš vysoké dimenze někdy příliš složité. Aplikací redukce dimenzí je poté velmi mnoho a používají ji často právě i doporučovací algoritmy.

K pochopení tohoto přístupu si tentokrát bohužel čtenář nevystačí s běžným chápáním matematiky základní a střední školy, ale pro zvědavé naznačme, že hlavní trik znějící poměrně možná pro neznalé trochu šíleně, spočívá ve vytvoření nového systému souřadnic dat. Již nemáme tedy tradiční *x* a *y* ze základní školy (a další písmenka abecedy, jestliže bychom byli ve vyšších dimenzích), ale jen pár tzv. *hlavních komponent*. Dosaženo tomu je pomocí triků lineární algebra se splněním několika statistických předpokladů. Jeden z těch snadněji pochopitelných předpokladů je například hledání největšího rozptylu dat, jehož bude dosáhnuto po promítnutí dat kolmo na nově vzniklé osy (hlavní komponenty).

Obrázek 7: Znázornění principu redukce dimenzí. *Var 1* a *2* znázorňuje proměnné, *PC1* hlavní komponentu (v tomto velmi zjednodušeném případě) je pouze jedna. *x̄* je průměr (výběru/vzorku dat) a *s2* („s na druhou“) rozptyl (výběru dat).

Problém redukce dimenzí je vcelku očekávaný. Při redukci dimenzí dochází defacto ke kompresi informací (podobně jako tomu je například u fotografií uložených v počítači v určitých formátech) a ač ne vždy ztratíme informace, některé informace, např. ohledně vzdálenosti objektů ve vyšších dimenzích, ztratit můžeme. Můžeme například ztratit informaci o vzdálenosti. To je samozřejmě velký problém, vzhledem k tomu, že měření vzdálenosti se velice často používá právě u doručovacích systémů jakožto indikátor podobnosti objektů. Chápat to můžeme jako dva body v tomto „šnekovi“ znázorněném níže. Body jsou blízko sebe, ale pokud bychom jej rozmotali, informační entropie. Informační byly by od sebe velice daleko. Není vzdálenost jako vzdálenost…

Obrázek 8: Znázornění problému měření vzdáleností u vysokých dimenzí.

* 1. Klasifikace

Asi nejčastější úkol v případě datových sad, ve kterých máme k dispozici správné hodnoty (oštítkovaná data), je klasifikace do 2 nebo více skupin. Můžeme například chtít na základě zadaných dat o domu dostat od modelu odpověď jednoznačnou: „ANO/NE“, zda se naší realitní kanceláři vyplatí koupit dům anebo zadat parametry sportovce a nechat si od natrénovaného modelu strojového učení rozřadit sportovce do výkonnostních skupin na základě změřených parametrů (např. čas sportovce v milisekundách na 30 m a maximální zvednutá váha v dřepu a podobně nám napoví, zda bude sportovec patřit do tréninkové skupiny „pokročilí“ nebo „začátečníci“).

Často využívanou metodou jsou například tzv*. rozhodovací stromy* nebo *lesy*. Model si umí vytvořit různá pravidla pro rozřazení skupin tak, aby tato pravidla seděla do předložených vzorových správných výstupů.

Máme například takováto data [142]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Den** | **Předpověď** | **Vlhkost** | **Vítr** | **Můžu hrát tenis** |
| 1 | Slunečno | Vysoká | Slabý | NE |
| 2 | Slunečno | Vysoká | Silný | NE |
| 3 | Zataženo | Vysoká | Slabý | ANO |
| 4 | Déšť | Vysoká | Slabý | ANO |
| 5 | Déšť | Normální | Slabý | ANO |
| 6 | Déšť | Normální | Silný | NE |
| 7 | Zataženo | Normální | Silný | ANO |
| 8 | Slunečno | Vysoká | Slabý | NE |
| 9 | Slunečno | Normální | Slabý | ANO |
| 10 | Déšť | Normální | Slabý | ANO |
| 11 | Slunečno | Normální | Silný | ANO |
| 12 | Zataženo | Vysoká | Silný | ANO |
| 13 | Zataženo | Normální | Slabý | ANO |
| 14 | Déšť | Vysoká | Silný | NE |

Tabulka 1: Data o počasí pro rozhodovací strom.

Výsledný rozhodovací strom by pak mohl vypadat nějak takto:

Obrázek 9: Rozhodovací strom sestavený na základě dat výše.

Konstrukci stromu dokáže model zvládnout pomocí konceptu *informační entropie*. Informační entropii si lze představit jako míru nepředvídatelnosti souboru dat. Soubor, který obsahuje pouze jednu třídu (například modrou), je extrémně předvídatelný: cokoli v něm je modré. Takový soubor má však nízkou entropii a příliš nám nepomůže v rozřazování dat, protože rozdíly mezi pravděpodobnostmi na základě dat jsou příliš malé a nemůžeme tedy dobře sestavit pravidla pro rozřazení. Soubor mnoha smíšených tříd je naopak nepředvídatelný: daný prvek může mít jakoukoli barvu! To by znamenalo vysokou entropii. Díky entropii můžeme zjistit informační zisk, jaký nám dá dané rozdělení dat na skupiny. Nejlepší rozdělení (nejlepší sestrojený strom s pravidly) je zvolen na základě maximalizace informačního zisku.

Rozhodovací stromy jsou jednou z mála současných metod, které můžeme implementovat tak, aby vypsaly uživateli přesně, jak dospěly k danému rozhodnutí (podobně jako například expertní systémy). Lze si tedy také představit použití v extrémně citlivých oblastech na chyby, kdy je nutná kontrola lidským expertem (např. medicínské odvětví). Nevýhodou je však, že nedokážou přesně aproximovat skutečné průběhy funkcí, jelikož rozhodovací stromy jsou diskrétní. Například i zmiňovaný fuzzy přístup oproti tomu může se správně zvoleným souborem pravidel funkce aproximovat daleko přesněji.

* 1. Shlukování

Pokud předem nemáme nebo nevíme správné hodnoty, na kterých bychom mohli model strojového učení natrénovat, neznamená to, že je vše ztraceno. Podle některých teprve tady začíná ta „pravá umělá inteligence“. Zatímco při předložení správných výstupů vlastně „jen“ aproximujeme nějakou složitou funkci s vidinou toho, že nám model na budoucích datech přinese užitečné odpovědi, zde si musí model strojového učení poradit sám. Sám si musí data zorganizovat a najít v nich různé vztahy a vzory bez znalosti toho, co je „správným“ nebo „chybným“ výstupem.

Asi nejčastěji se při učení bez učitele bavíme o shlukování dat (což v praxi znamená nalezení podobnosti mezi pozorovanými objekty s určitými vlastnostmi; např. můžeme mít digitální knihovnu a hledat nejpodobnější knihy z knihovny k jedné dané knize). Určí se tzv. *centroid* (středový bod, ono těžiště) a objekty se „shlukují“ do skupin podle tohoto centroidu, ke kterému mají nejblíže. Body zařazujeme do sousedství, které mají hranice. Aby bylo takové shlukování správné, je potřeba podobně jako tomu bylo u redukce dimenzí splnit soubor pravidel, jako například, že objekty nemohou být rozřazeny do shluků tak, aby nějaký objekt mohl být blíže objektu jiného shluku oproti objektům vlastního shluku. Jedná se tedy o proces, který není okamžitý a nějakou chvíli trvá, než je nalezeno konečné rozdělení prostoru na sousedství. Jestliže přidáme nějaké nové body (nebo klidně i jeden), musí se opět vše přepočítat. Polygony na obrázku níže vymezují hranice sousedství určující rozřazení objektů do shluků.

Obrázek 10: Náčrtek postupu shlukovací metody. Iterativně dochází k přepočtu těžiště s dodržením zásad shlukovací metody.

Aby to vše fungovalo, pracujeme kromě souboru pravidel, které je nutné dodržet, také s konceptem těžiště shluku a měření vzdáleností. Měřit můžeme například pomocí Euklidovské, Cosinovy nebo Manhattanské vzdálenosti. Ty jsou naznačeny níže. Manhattanskou vzdálenost si lze představit jako jízdu pozemním dopravním prostředkem skrze pravidelně uspořádané ulice newyorského centra. Euklidovskou vzdálenost jako přelet vrtulníkem z bodu A do bodu B vzdušnou čarou. Cosinova vzdálenost oproti tomu měří úhel mezi 2 vektory – v tomto případě by tedy vektory po vizualizaci ukazovaly na naměřené hodnoty („datové body“).

Obrázek 11: Nahoře znázornění Euklidovy a Manhattonské vzdálenosti, dole cosinovy podobnosti

* 1. Evaluace

V minulých částech, kde byly zhruba přiblíženy dnešní časté metody strojového učení, jsme opomněli ještě jednu důležitou věc: jak daná společnost, která vám předává pomocí umělé inteligence nějaký obsah, ví, že je to zrovna, jak se lidově říká: „to pravé ořechové“ než pustí model do reálného provozu mezi lidi a zapracuje jej například do své mobilní aplikace?

Základní myšlenkou je rozdělit data na 2 části: *tréninková data* a *testovací* *data*. Pro jednoduchost si představme učení supervizované (s učitelem), kdy máme určité štítky správných hodnot. Tedy hodnoty, které chceme naučit naši černou skříňku automaticky vygenerovat na základě různých faktorů (vstupů). Např. jsme realitní kancelář a na základě minulých dat firmy víme, že dům, který má 5 pokojů, byl postaven v roce 2004, má garáž a topení v podlaze, byl ohodnocen odborníkem na 4 miliony korun. Pokud máme dostatek takových údajů (a ideálně také nějaké další faktory, které jsou vzhledem k prodejní ceně relevantní, a pokud jsme spokojeni s dosavadními výnosy firmy), můžeme vytvořit tzv. *model*, což je ve zkratce nějaká složitá funkce, která se snaží co nejlépe transformovat vstupy *xi* na výstup *y* (možná vám opět pomůže pro pochopení nazvat vstupy a výstupy jako „vysvětlující“ proměnné a „vysvětlovanou“ proměnnou). Pokud bude takový model dostatečně spolehlivý, můžeme vzít úplně nový dům, zadat jeho údaje a nechat si modelem předpovědět odhadnutou cenu (a teoreticky tak ušetřit za odborníka provádějící odhady).

Abychom měli jistotu, že je tento model použitelný v praxi a zda se tyto údaje vůbec lze naučit a zda jsme například neměli jen moc velkou smůlu a nevybrali si zrovna špatnou porci dat (kdy by model dával příliš dobré nebo příliš špatné výsledky oproti pozdějšímu použití v praxi), využívá se například několikanásobné validace, křížové validace anebo se také ponechává navíc ještě jedna množina dat, kterou vůbec nepoužíváme do té doby, než je model odladěn na množinách pro trénování a validaci. Můžeme tak doladit model, či vybrat jeden z více modelů, a nakonec jej ověřit pro jistotu ještě jednou – to nám dá ještě trochu větší jistotu, že jakmile jej například představíme v rámci webové nebo mobilní aplikace, bude fungovat na nových datech dostatečně dobře.

Obrázek 12: Znázornění pokročilé metody ověření modelu, tzv. *křížové validace*.

* 1. Neuronové sítě

Pokud si vezmeme lineární regresi nebo základní klasifikační metody pomocí rozhodovacích stromů, narazíme na velký problém. Jestliže nejsou data tzv. lineárně oddělitelná (tedy oddělitelná čarou), dané metody nefungují. I pomocí lidského oka a mozku jednoduše oddělitelná data, jako například ta na obrázku níže vpravo (které by se daly oddělit v tomto případě kružnicí), nejsou pouhou přímkou oddělitelné. Ať už bychom chtěli tedy klasifikovat nějakou jednodušší metodou nebo předpovídat pomocí lineární regrese, měli bychom problém. Potřebujeme zjednodušeně řečeno jiné nástroj, které umí přímku „zohnout“.

Obrázek 13: Znázornění příkladu vizualizace lineárně oddělitelných dat a lineárně neoddělitelných.

Jestliže my, lidé (resp. náš mozek) dokážeme dobře vnímat vzory a snadno tak do takového obrázku dokreslit kružnici a data tak oddělit, někoho velmi chytrého napadlo, že by se dalo napodobit to, jak funguje nervová soustava inteligentních tvorů. Níže je uveden příklad s vizualizacemi nejjednoduššího modelu neuronové sítě – tzv. *perceptronu* a je znázorněn také přepočet hodnot. Tedy ono slavné „učení“. Objekt vizualizován trojúhelníkem byl nejprve zařazen do špatné skupiny. Poté vezmeme jeho správnou polohu a klasifikační pravidlo pro určení hodnoty *d* a upravíme rovnici přímky. Níže je vidět, že již přímka odpovídající rovnici nad správně klasifikuje tento předtím špatně zařazený objekt:

Obrázek 14: Parametry a vstupy v neuronové síti.

Obrázek 15: První část výpočtu.

Obrázek 16: Příklad klasifikace dat pomocí perceptronu.

Nápad je to jistě brilantní. Dokážeme nyní oddělit prostor elegantním způsobem, který si navíc bere inspiraci z toho, jak funguje lidský neuron. Moc navíc jsme toho však nedokázali a pole řešitelných problémů nám to zatím moc nerozšíří. Pořád můžeme oddělovat pouze přímkou. Revoluční myšlenkou perceptronu ale je, že nyní můžeme spojit více takových neuronů do neuronové sítě. A poté se teprve začínají dít zajímavé věci...

Pojďme ale postupně. Jestliže máme model umělého neuronu, pojďme si vytvořit model neuronové sítě s jednou vrstvou. Vlastně nic takového ani vytvářet nemusíme, protože něco velmi podobného existuje ve statistice již od 60. let a jmenuje se to *logistická regrese* a je to základní klasifikační metoda, která byla záměrně prozatím čtenáři zatajena.

Logistická regrese se dá chápat druhým pohledem jako jednoduchá, jednovrstvá neuronová síť. Jádrem této metody je křivka tvaru „logistického sigmoidu“. Právě tento unikátní tvar křivky dodává hned několik cenných vlastností. Oproti přímce můžeme lépe vyjádřit, že i malé rozdíly u středu jsou velmi důležité pro rozřazení, zatímco rozdíly mezi hodnotami ležící na číselné ose dále od 0 (velké záporné hodnoty a velké kladné hodnoty) nejsou až natolik důležité. Pokud například budeme chtít předpovědět, zda se na základě hodnocení testu má šanci daný student dostat na vysokou školu, můžeme „vystředovat“ bodování studentů. Blízko nuly budeme mít studenty, kteří měli průměrné hodnocení. Většinou je pak velký rozdíl, jestli má někdo lehce nadprůměrné nebo lehce podprůměrné hodnocení. To může zásadně rozhodovat o úspěchu/neúspěchu na vysoké škole. Naopak, pokud má někdo velmi špatné hodnocení, je nám již vcelku jedno, jestli obdržel z testu 4 body nebo 10 bodů ze 100. Předpokládáme, že takovíto studenti pravděpodobně neuspějí a nedáváme takovou hodnotu malým rozdílům. Stejně tak u velmi nadprůměrných studentů nám tolik nezáleží, jestli budou mít 98 nebo 94 bodů. Předpokládáme, že takoví studenti jsou velmi dobří a mají podobně velkou šanci uspět. Studenty na „chvostu“ při použití této metody úplně nezatracujeme, metoda však určí, že je velmi malá šance, že by studenti se 4 body ze 100 uspěli. Na základě tohoto principu dá v konečné implementaci model uživateli rozřazení do kategorií studentů s rizikem neúspěchu na vysoké škole a úspěchu.

Obrázek 17: Ilustrace sigmoidní křivky.

I obyčejná logaritmická funkce ze základní školy se ostatně chová podobně. Zatímco rozdíl vstupu 1 a 5 je na výstupu obrovský, pro velké vstupy již takové rozdíly mezi výstupy nejsou. Proto se často používá logaritmus nebo jeho příbuzné funkce v různých vzorcích, kde je potřeba zohlednit, že je velký rozdíl na nějakém úseku osy (nějakém rozsahu vstupů), zatímco např. když už jsou čísla příliš malá nebo velká, rozdíly už nás tolik nezajímají (rozdíl to pořád je, ale ne tak významný)

Obrázek 18: Vizualizace průběhu logaritmické funkce log(x). Zatímco je velký rozdíl mezi hodnotou na ose *y* u hodnoty na ose *x* 2 a 4, u vyšších čísel rozdíly nejsou tak významné.

Sigmoidní křivka je dobrým přítelem pravděpodobností. Tzv. *odds ratio* nám dává poměr šancí. Pokud máme 2 události A a B, máme šanci, že nastane událost A, když nastane také událost B lomeno šance, že A nastane bez B. Asi srozumitelněji řečeno, znamená to, že: čím větší je šance, že nastane událost, kterou chceme předpovědět, tím větší je tento poměr šancí.

Zdatnější znalci matematiky si mohou vzít jednoduchý vzorec pro výpočet „šance“: *p/(1-p)* udávající šanci, že nastane nějaká událost *Y* a ona geniální myšlenka za logistickou regresí spočívá v tom udělat z této „šance“ přirozený logaritmus. Rázem se z toho stane tzv. *logit funkce* mající sigmoidní tvar, která po troše úprav dává vzorec vyjadřující vztah mezi pravděpodobností a logit funkcí. Osa *y* je pravděpodobnost, osa *x* je hodnota logitna základě pravděpodobnosti.

Obrázek 19: Znázornění pravděpodobností a šancí u logistické regrese.

Po pár krocích odvozování bychom tedy dostali matematický zápis vztahu, která by ve vizualizované podobě udal vztah mezi pravděpodobností a „logit“ funkcí (tedy přirozeným logaritmem aplikovaným na vzorec pro poměr šancí):

Obrázek 20: Znázornění vztahu mezi funkcí logit a pravděpodobností.

Tato sigmoidní funkce je podobně jako v případě perceptronu lineární funkce. Zde však máme více sofistikovaný způsob a ukazuje se, že právě tento sigmoidní tvar dobře napodobuje prahový potenciál skutečných neuronových spojení v lidském mozku!

Mezi neurony v lidském těle dochází ke slabým elektrickým impulsům. Aby nervový impuls dorazil z jednoho neuronu na druhý, musí zde být změna elektrického potenciálu. Tento potenciál slouží jako „brána“ (práh) mezi dvěma neurony. Určuje sílu signálu propuštěného dále. To je tedy další důvod, proč je sigmoidní křivka výhodná. Víme, že to je dobrý základ pro učení, protože se něco v daleko složitější podobě děje v lidském mozku (v přírodě).

Obrázek 21: Rozdíl a podobnosti perceptronu a logistické regrese.

Obrázek 22: Logistická regrese se dá chápat jako jednoduchá neuronová síť.

I v jednoduchých modelech neuronů se přidává tzv. *bias*. Větší bias znamená při použití sigmoidní funkce „přísnost“ prahu. Jestliže máme například vstupy mezi ‑10 a +10 a výstup mezi 0,0 a 1,0 a přijde nám vstup +5, při menším menší biasu se sigmoidní křivka „zploští“ a tento vstup se promítne do výstupu 0,8. Při větším biasu se naopak promítne do skoro maximální hodnoty výstupu, např. někde k 0,97. Z pohledu elektrotechnického a biologického se toto dá chápat jako statický potenciál neuronového spojení. Tento klidový potenciál, v elektrotechnickém žargonu „předpětí“, určuje, jak silný elektrický impuls neuron potřebuje, aby dosáhl svého prahu aktivaci („vystřelení“) a vytvořil výstupní signál.

Obrázek 23: Bias ovlivňuje tvar sigmoidní funkce tvořící práh neuronové sítě.

Perceptron a logistická regrese v podobě modelu neuronové sítě je tedy prakticky stejný model, „jen“ se liší v této prahové funkci propouštějící signál na výstup. Tento malý rozdíl je však velice významný. U perceptronu to jednoduše buď je hodnota menší než 0 a signál dál nejde vůbec anebo je větší než 0 a pak je signál maximální (tedy 1) – žárovka buď svítí naplno nebo ne. U neuronového modelu logistické regrese dostaneme na výstupu hodnoty mezi 0,0 a 1,0 na základě výhodného tvaru sigmoidní funkce, který modeluje lépe, jak to funguje v lidském mozku.

Na těchto principech pak staví neuronová síť. Zde máme více vrstev a můžeme propojit neurony ve vrstvách navzájem (například každý neuron se všemi následujícími. Každá z vrstev většinou obsahuje svou aktivační funkci (často je to ale aktivační funkce stejného tvaru pro každou vrstvu, ale může to být navrženo i jinak – záleží ostatně jako obvykle v tomto oboru, co funguje nejlépe pro řešení konkrétního problému). V každé vrstvě máme jako vstupy výstupu z předchozí vrstvy. Každá vrstva (resp. spojení) v ní má své váhy a do jednotlivých uzlů sítě jde také bias. Stejně jako v případě samotného vstupu.

Pořád zde nevidíte to „učení“?

Učení spočívá v tom, že pokud máme vzorová data se správnými výstupy, můžeme po výpočtu výstupu neuronové sítě na základě vstupů, projít neuronovou síť zpět a poupravit tyto váhy tak, aby se neuronová síť poučila ze svých chyb a pokud by takový vstup obdržela znovu, již by příště navrátila výstup bližší tomu správnému. Jelikož se zde jedná o optimalizaci, využívá se často například výše zmíněné metody gradientního sestupu pro dosažení (alespoň lokálního) optima. Dostáváme se tedy zpět na úvod této kapitoly (pamatujete si ještě příklad s horolezcem?).

Dnešní výzkum v oblasti neuronových sítí se zaměřuje například na to, jak nejlépe vhodně propojit neurony nebo dokonce více neuronových sítí navzájem a jaké aktivační (prahové) funkce je výhodné použít a kde.

Obrázek 24: Náčrt principu neuronové sítě. Pozorujeme rozdíl mezi skutečnou (správnou) hodnotou v tréninkové sadě (pozorováním) a výsledkem modelu spočítaném neuronovém sítí. Na základě této chyby aktualizujeme váhy neuronové sítě.

Na těchto principech fungují tedy i dnešní nejpokročilejší velké jazykové modely známé z chatbotů, které vzbudily tak velký nedávný ohlas u veřejnosti. Rozdíl je v tom, že je zde složitější architektura neuronové sítě, kde je mnoho různých vrstev chytře spojených tak, aby model napodobil například lidskou krátkodobou a dlouhodobou paměť a lidskou pozornost, aby byla přidána důležitost různým částem věty a jiným naopak menší důležitost.

Tato neformální část tedy snad dodala intuitivní cestou potřebný základ pro následující části věnující se doporučujícím systémům. V případě výskytu omylů, budu rád za upozornění a případně budou opraveny v následujících vydáních.

1. Současné doporučovací systémy

Na doporučovací systémy můžeme z datového/statistického hlediska nahlížet jako na prakticky jakýkoliv jiný problém predikce hodnot na základě několika různých faktorů (což je podstatou modernějšího označení oboru „strojového učení“). Můžeme tedy uvažovat o některých z výše zmíněných metod. Jsou zde však jistá specifika, a proto existuje mnoho dalších, pokročilých metod, které se u doporučovacích systémů často využívají. Jeden z podstatných rozdílů je například velké množství chybějících hodnot. Někteří tedy na doporučovací systémy nahlížejí spíše optikou lineární algebry jako na problém hledání chybějících hodnot v tzv. řídké matici. Tato matice je tvořena typicky například hodnoceními uživatelů, kdy řádky a sloupce označují uživatele a položky (např. filmy, písně, příspěvky, články apod. v závislosti na konkrétním doporučovacím systému). Není těžké si poté představit, že taková matice všech uživatelů a filmů např. ve velké filmové databáze se všemi známými filmy bude hodně „řídká“, protože ani ti nejaktivnější uživatelé nebudou hodnotit mnoho z lokálně nebo méně známých filmů a zbytek hodnot bude tedy nulový.

Ještě větší problém s kvalitou informací ke zpracování pak nastane, když například z důvodu uživatelského rozhraní chceme, aby uživatel hodnotil pouze palci nahoru, a dokonce ani ne dolu. Máme poté najednou informaci o tom, co se uživateli líbí, ale není jisté, zda nehodnocená položka je neutrální, nebo se mu vyloženě nelíbí.

Další problém nastane, pokud nám chybí „správné“ hodnoty a nemůžeme přistupovat k problému okamžitě jako k učení s učitelem. Při startu firmy nebo při zavedení doporučovacího systému v rámci instituce (např. knihovna bude chtít doporučovat knížky svým čtenářům), nebudeme mít k dispozici žádná vzorová data a budeme budovat doporučovací systém takzvaně „ze zelené louky“.

K doporučení určitých položek chceme také většinou chceme přidat řazení. Není to tedy jen o tom určit, zda je položka relevantní nebo ne. Chceme také, aby uživateli byly zobrazeny ty nejvíce relevantní položky mezi prvními.

Bohužel, pokud se podíváme na současné vědecké práce zabývající se doporučovacími systémy, málokterá splňuje následující omezení:

* Neanglické texty. Současné práce k doporučovacím systémům pracují převážně s anglickými texty. Většina modelů a metod pro zpracování textů je přitom určená pro anglické texty.
* Malý počet položek v datové sadě. Mnoho ze současných prací používá velké datové sady, často dokonce různě „vyčištěné“ a zdokonalené. To nevystihuje dobře reálnou situaci surových textů, které mohou např. obsahovat gramatické chyby.
* Absence vzorových dat. Mnoho ze současných prací využívá datové sady, kde jsou známé hodnocení uživatelů příspěvků nebo jejich posouzení relevance Lépe se jim tak model trénuje a posuzuje. Reálná situace je přitom jiná. Sběr hodnocení/posouzení uživatelů stojí značné zdroje.
* Metoda nebo algoritmus, který je dobře implementovatelný a podporován komunitou vývojářů a datových vědců. Některé vědecké články představují nové metody, které vykazovaly při jejich experimentech dobré výsledky, avšak v odborném článku ani přidružených materiálech nemusí být dostatek informací ohledně toho, jak dané algoritmy naimplementovat. Softwaroví vývojáři (kteří často nemají znalosti vyšší matematiky na úrovni doktorského studia nebo ani navazujícího, či vůbec vysokoškolského studia) tak nemusí být schopni dle článků doporučovací systém „nakódovat“.

Přitom všechna tato omezení jsou velmi důležitá pro mnoho z menších společností a institucí, kteří by rádi doporučovací systém využily. Náš doporučovací systém má tedy všechna tato zmíněná omezení. Použité metody tedy sahají také do příbuzných oborů jako zpracování (přirozeného) jazyka nebo dolování znalostí. Místo honbou za co nejlepšími výsledky při evaluacích doporučovacích systémů, se raději zaměřujeme na použitelnost v praxi.

* 1. Příbuzné obory

Obecně můžeme říct, že problematika doporučovacích systémů se skládá z těchto příbuzných oborů a podoborů:

* Statistika.
* Strojové učení.
* Umělá inteligence.
* Dolování znalostí.
* Získávání informací.
  + Vyhledávače.
* Lineární algebra.
* Zpracování (přirozeného) jazyka.

Typické použití doporučovacích systémů v praxi zahrnuje například:

* Řazení příspěvků (sociální sítě).
* Doporučování na základě podobných hodnocení uživatelů (aplikace pro streamování video, audio materiálu, databáze knih apod.).
* Hledání podobných položek (např. e-shopy a srovnávače cen produktů, aplikace zaměřené na hoteliérství apod.).
* Aplikace pro hledání shodných profilů s jinými osobami na základě zájmů, (např. v rámci sociálních sítí, kariérních aplikací nebo aplikací pro digitální sjednávání rande).
* Vyhledávače. Současné populární vyhledávače používají pro řazení výsledků také personalizaci pomocí zeměpisných údajů uživatele nebo jeho historie předchozích hledání a prokliků. Mnoho ze současných vyhledávačů lze tedy také považovat za doporučovací systémy.

Obecně lze problémy řešené u doporučovacích systémů rozdělit na problém:

* **Problém predikce**. Zda uživatel shledá danou doporučenou položku relevantní.
* **Problém řazení**. Zda uživatel nalezne pro něj relevantní položky řazené na vyšších pozicích (ve výsledném uživatelském rozhraní zobrazené nahoře) než pro něj méně relevantní položky.

Doporučovací systémy můžeme dle povahy metod používaných k doporučování rozdělit na doporučovací systémy založené na:

* **Obsahu**. Nevýhodou doporučování na základě obsahu je, že přirozený jazyk je složitý a různorodý a pro strojové zpracování je často potřeba určitá forma převodu textů do jednodušší podoby (např. odstranění diakritiky, speciálních znaků, sjednocení tvarů slova v různých pádů do jednoho apod.). Doporučení na základě obsahu můžeme dále rozdělit na případy, kdy:
  + Samotné doporučované položky tvoří data používaná pro posuzování doporučení.
  + Profil uživatele tvoří data používaná pro posuzování doporučení.
* **Kolaborativní**. Na základě akcí uživatele (např. hodnocení filmů) jsou vyhledáni uživatelé s podobným „vkusem“.   
  Způsob sběru hodnocení od uživatelů má zásadní vliv na doporučovací systém. Např. ze sociální sítí známe hodnocení pomocí palců nebo symbolů srdíček. Toto tzv. *unární hodnocení* položek není však vůbec ideální pro zpracování dat a potřebné predikce pro uživatele. Jak již bylo řečeno výše, u nehodnocených položek nám splývají položky vnímané neutrálně a negativně. Hodnocení také nebudou tak přesná (např. někteří uživatelé, kteří by hodnotili položku jako středně dobrou hodnocením 3 v Likertově stupnici, by mohli hodnotit příspěvek ještě pozitivně, ale jiní zase ne, viz obrázek níže).   
  Je tu však i druhá strana mince. V některých aplikacích je tento způsob hodnocení preferován kvůli marketingovým důvodům, designu a uživatelskému rozhranní. Můžeme však předpokládat, že takové hodnocení může mít i podstatnou výhodu vzhledem k datové stránce. Uživatelé budou daleko ochotnější hodnotit příspěvky pouhým klikem (nebo ťuknutím na displej) než pomalejším rozmýšlením, zda ohodnotí položku např. 3 nebo 4. Velmi pravděpodobně tak nasbíráme dat více než u přesnějšího posuzování příspěvků.

Obrázek 25: Likertova škála (v uživatelském rozhraní nejčastěji zobrazeno jako hvězdičky) vs unární hodnocení (nejčastěji palce nahoru nebo symboly srdce, apod.).

Zatímco u klasifikačního problému, který se často řeší v rámci strojového učení si určitou část dat vyhradíme jako tréninkovou a druhou část jako testovací (případně ještě také třetí jako validační) a data si tedy můžeme představit jako tabulku s jedním sloupcem vyhrazeným pro „správné“ hodnoty, u kolaborativního filtrování data tvoří matici hodnot (připomínáme znovu, že ve většině doporučovaích systémů mnoho hodnot v této matici chybí, bavíme se tedy o tzv. řídké matici, angl. *sparse matrix*). Tuto matici tvoří například hodnocení uživatelů a řádky odpovídají uživatelům, zatímco sloupce samotným doporučovaným položkám (např. filmům). Pokud však máme pro to vhodná data, přičemž základní předpoklad je samozřejmě opět znalost *y*, coby hodnot „správného“ výstupu a pokud se spokojíme s tím, že nekompletní data doplníme z dat stávajících, k problému doporučování na základě uživatelských akcí můžeme přistoupit i jako k „tradičnímu“ klasifikačnímu problému. V takovém případě můžeme tedy použít v mnohých aplikacích úspěšné metody jako např. rozhodovací stromy, lesy nebo neuronové sítě.  
Srovnání těchto přístupů je načrtnuto na obrázku níže.

Obrázek 26: Ilustrace rozdílu v datech, které jsou vstupem metod založených na klasifikaci vs metod založených na kolaborativním přístupu.

Někoho však možná napadne, že pokud již data doplňujeme, dalo by se takového přístupu využít i u matice s chybějícími hodnotami. V „maticovém“ případě se používají často metody tzv. *(latentní) faktorové analýzy*. Základní myšlenkou je doplnit chybějící hodnocení na základě dvou hlavních nástrojů: redukce dimenzí a korelace hodnot v matici. Jednou z metod z této rodiny metod je metoda SVD (angl. *Singular Value Decomposition*, česky *singulární rozklad*), která bude více přiblížena blíže a byla implementována do našeho systému.

* **Znalostní** (angl. *Knowledge Based)*. Uživatel se aktivně zapojuje do specifikace svých preferencí. Skrze vhodné uživatelské rozhraní se ptáme uživatele na jeho preference, např. „Líbí se vám tento film?“ nebo „Zaškrtněte oblíbené žánry filmů“. Nevýhodou tohoto principu je, že ne vždy chtějí uživatelé takto aktivně participovat. Tento přístup může mít tedy negativní marketingové, a tudíž i negativní ekonomické dopady.
* **Hybridní**. Kombinují více různých přístupů dohromady s vidinou toho, že každý z přístupů má své slabé a silné stránky. Vhodná kombinace metod by měla využít silné stránky metody. Největším problémem kolaborativních metod je například počáteční absence dat uživatelů (tzv. *problém studeného startu*).   
  Pokud máme nový systém, nemáme vůbec data o uživatelích nebo máme příliš málo dat (např. hodnocení filmů uživateli). Je možné tedy buď doporučovat pouze pomocí obsahu (např. textu popisu filmů) nebo bude kolaborativní přístup tvořit jen část konečného výsledku a vážíme jednotlivé výsledky modelů. V nejjednodušším případě to může být např. pouhý součet numerické hodnoty vypočítaného doporučení na základě obsahu a kolaborativní metody. Numerická hodnota doporučení na základě kolaborativního filtrování tedy lehce může zkreslit výslednou numerickou hodnotu doporučení, ale druhou složku zde pořád tvoří také doporučení na základě obsahu, které by mělo fungovat v tomto případě stejně jako jindy.   
  Výsledky lze také míchat pokročilejším způsobem, např. určitou porci položek výsledků bude tvořit určitá metoda a tento správný poměr zjistíme pomocí strojového učení. Nad jednotlivými modely doporučovacích metod může být tedy další model, který zde bude pro správné „namíchání“ a řazení výsledků. Možný je také paralelní návrh systému. V takovém případě neskládáme výsledky z jednotlivých modelů, ale výsledky procházejí postupně více různými modely až k finálnímu, přičemž nakonec zde může být také model, který bude kombinovat výsledky. Do tohoto posledního modelu může dorazit ten, který projde skrze všechny předchozí (výstup předposledního modelu), ale také ještě vstup z jakéhokoliv jiného předchozího modelu (záleží na návrhu, tedy na tom, co bude fungovat nejlépe).  
  Na pomezí hybridních a kolaboračních metod pak stojí tzv. *metody založené na sousedství* (angl. *Neighborhood-Based Methods*) mezi které patří *modely založené na položkách* (angl. *Item-based models*). Ty oproti *hledání podobností hodnocení uživatelů* (*modely založené na uživatelích*, angl. *User-based models*), hledají také podobnosti mezi položkami a poté předpokládají, že podobné položky se budou líbit podobným uživatelům. Protože jsou takové metody často náročné na zpracování nebo musí pracovat se značně nekompletními daty, často se zde používají metody pro shlukování nebo redukce dimenzí nebo grafové metody [139] [140] [141]. Můžeme je rozdělit na:
  + **Kontextově senzitivní**. Doporučení se mění na základě různých kontextů týkajících se uživatele. Např. na základě aktuálního času a místa uživatele, na základě času stráveného interakcí s položkou (např. sledováním, prohlížením, čtením) apod. Nevýhodou tohoto přístupu je možné zasahování do soukromí uživatele a možná složitější implementace. Do této kategorie bychom pravděpodobně mohli zařadit také doporučovací systémy založené na taxonomiích (nebo ontologiích), kdy jsou související položky propojené a/nebo tvoří hierarchie (prvek např. může být rodičovským od jiného). Do hybridních doporučovacích systému jsou zde tyto doporučovací systémy zařazeny, protože často jsou součástí (jakýmsi doplňkem) obsahových nebo kolaborativních doporučovacích metod. Data si pak lze představit jako kvádr, kdy máme nejen řádky jako uživatele, sloupce jako položky, ale také jednotlivé matice vyjadřující např. rozdílné denní období – např. hodnocení filmů v 6 hodin večer a 10 hodin dopoledne. Takovéto skládání matic se nazývá *tenzor*. Tenzor hodnosti 2 je matice, hodnosti 1 vektor, hodnosti 3 tvoří „krychli“ nebo „kvádr“ hodnot a můžeme samozřejmě pokračovat i dále, avšak jako obvykle si vyšší dimenze již nedokážeme představit.

Ani pokud se nám podaří doporučovat s uspokojivou relevancí pro uživatele, nemusíme však mít vyhráno. Máme zde totiž další potenciální aspekty, které mohou ovlivnit kvalitu doporučovacího systému. Např.:

* **Doporučování nových položek.** Pokud bychom doporučovali přesně, ale pořád stejné výsledky, doporučování by pro uživatele mohlo ztratit smysl. Často je žádoucí, aby doporučované položky přinesly určitý pocit příjemného překvapení. Pokud by např. doporučovaí systém pro filmy doporučoval snímky, které uživatel již viděl nebo je zná z doslechu, nebyl by tak užitečný, jako když doporučí vhodný film, o kterém uživatel ještě nikdy neslyšel.
* **Diverzifikace doporučení**. Např. v doporučovacím systému pro novinové články by nebylo žádoucí, dokonce ani zodpovědné, doporučovat články s informacemi potvrzující uživateli jeho „světonázor“, díky kterým by se uživatel uzavřel do informační bubliny. Pokud bychom chtěli uživatele skutečně dobře informovat, měl by být čas od času vystaven také např. informacím ohledně jiné politické strany, než kterou aktuálně preferuje. Problém takové diverzifikace je však její špatná měřitelnost, a tím pádem i ověřitelnost.
  1. Doporučovací systém pro novinové články

Zatímco před digitální érou měly papírové noviny finanční hodnotu danou nejen obsahem, ale i fyzickým výtiskem, na dnešních zpravodajských nebo jiných portálech obsahujících články, je hodnota článku dána především:

* Prokliky do článků a časem stráveným na webové stránce článku. Zisk pro redakci poté tvoří především reklamy zobrazované uživateli.
* Pozitivním dojmem, který přispěje ke koupi předplatného (v případě, že má daný zpravodajský web určitou část článku uzamčenou pro předplatitele).

Priorita článků zpravodajských webů má dnes kromě novinářských zásad, svůj původ také v potenciálu masové popularity a je ovlivněna různými marketingovými zásahy, které mohou negativně ovlivňovat hodnotu obdržených informací. Některým zpravodajským portálům, internetovým magazínům nebo blogům přitom nestačí vybírat pouze obecně nejpopulárnější články nebo ty nejdůležitější vybrané a seřazené redakcí (jak tomu bylo a je u papírových novin), ale potřebují zachytit různorodost uživatelů, kterou majitelé potřebují udržet na svém webu a zobrazit mu nejen informace určené všem uživatelům, ale i konkrétnímu jednomu uživateli. Doporučující systémy, které hrají dnes zásadní roli v rámci e-shopů, sociálních sítí nebo služeb pro přehrávání multimediálního obsahu, se tedy jeví jako možné východisko i pro tento obor.

I pro čtenáře může přinést doporučující systém v této oblasti výhody. V éře, kdy jsou lidé zaplaveni informacemi, může být problémem obdržet relevantní informace z článků na internetu.

Doména doporučujících systému tedy nabízí možné řešení jak pro majitele zpravodajských serverů, magazínů nebo blogů, tak pro čtenáře (uživatele).

V této práci byl představen adaptivní doporučující systém pro aktuální články dostupný na adrese <https://moje-clanky.herokuapp.com> (pro nové uživatele dostupný pouze v závislosti na aktuálním stavu vývoje).

Účelem takovéhoto doporučujícího systému mělo být zobrazení článků v závislosti na preferencích uživatelů. Jelikož však doména digitálních článků není úplně totožná s běžnými doporučujícími systémy tvořenými pro produkty nejrůznějšího charakteru u doporučujících systému e-shopů, hudebních a filmových streamovacích služeb, kde mají v současnosti doporučující systémy využití největší, bylo u tvorby tohoto systému (ač se jedná o oborový přesah) brán ohled na metody založené pouze na obsahu. Toto vyvážení metod založených na uživatelích a na obsahu by mělo zajistit vhodný kompromis pro vystavení uživatele pestřejší paletě článků a přizpůsobením obsahu.

Cílem praktického projektu vytvořeného v rámci tohoto textu byl návrh a implementace webového doporučujícího systému pro automatizované doporučování a zobrazení vhodných aktuálních zpráv a článků. Systém na základě vstupních preferencí doporučil uživateli vhodné zprávy a články. Uživatel mohl články hodnotit, což mělo vliv na doporučující algoritmus. Hlavní součástí systému byl algoritmus, který adaptivně a průběžně navrhoval vhodné články na základě aktuálních preferencí uživatele a také historie přečtených článků v minulosti.

Systém byl prakticky ověřen na vybrané množině článků a zpráv a rovněž na reálných uživatelích.

Následující praktická část textu je členěna do těchto hlavních částí:

1. Analýza současného stavu.

2. Návrh systém.

3. Implementace systému.

4. Praktické ověření systému.

* 1. Současné aplikace zabývající se zobrazováním a doporučováním článků

Současné softwarové produkty zabývající se doménou zobrazování článků se dají rozdělit do dvou druhů:

* Agregátory zpráv.
* Aplikace pro ukládání zpráv.

Zatímco agregátory zpráv často obsahují nějaké doporučující algoritmy, aplikace pro ukládání zpráv většinou nikoliv a slouží pouze pro ukládání článků ze zvolených zdrojů do fronty typu LIFO („Last In First Out“), tzn. poslední uložený článek se uživateli zobrazuje jako první. Uložení článku probíhá například načítáním odkazu na RSS nebo někdy jen pomocí kliknutí na tlačítko doplňku prohlížeče.

V této podkapitole jsou uvedeny nejoblíbenější aplikace pro zobrazování a doporučování článků. Kromě popisu, kde je to možné, jsou rovněž naznačeny strategie zobrazování článků.

* + 1. Google News

Jedním z nejvyužívanějších agregátorů zpráv je *Google News* shromažďující články z různých webů. *Google News* shromažďuje pouze odkazy na články s fotografiemi, nikoliv samotné texty článků.

*Google News* si nejspíše uvědomuje, že k doporučování novinových článků se nemůže přistupovat jako k doporučování komerčních produktů, filmů nebo hudby. Proto jako první nabízí přehled „Události dne“. V této sekci jsou vybrány největší a nejdůležitější události, které se toho dne udály.

*Google* jen zřídkakdy vylučuje weby z výsledků vyhledávání, ale přiznává, že v dává přednost těm, které „nabízejí především aktuální zpravodajství nebo analýzu nedávných událostí“ [1].

Zaregistrování do služby musí provést samotní provozovatelé, resp. vydavatelé článků. Při registraci se ověřuje, zda je žadatel skutečným vlastníkem stránky, či pověřenou osobou. *Google* nabízí vydavatelům pravidla, v nichž jim radí, že pokud chtějí, aby jejich články byly posouzeny jako relevantní, musí psát “originální obsah, který je srozumitelný a bez gramatických chyb”. Mezi další kontroly kvality patří například jasné uvedení autorství, jasné označení případné zaujatosti autora a zahrnutí ručně přidaných hypertextových odkazů do kopie.

*Google* bohužel kvůli zřejmým podnikatelským důvodům hlídá své „know-how“ a není snadné nahlédnout pod pokličku jejich strategií a už vůbec jejich algoritmů více. Jejich oficiální texty jsou jen velice povrchní a obecné. Lze však vyvozovat, že *Google* bude při vybírání článků především využívat svá detailní data týkající se návštěvnosti webů. Jedna z mála důležitých informací, které sám *Google* poodkrývá je, že prý nepoužívá žádné zásahy “lidských” redaktorů, které by ovlivňovaly, jaké zprávy se uživateli zobrazí. Jediný zásah redakce se prý odehrává u specializovaných sekcí, které jsou věnované různým událostem jako mistrovství světa ve fotbale nebo předávání cen Oscarů [2].

Asi nejvíce informací o tom, jak vlastně *Google News* funguje, poodhalil produktový manažer *Google* Josh Cohen ve článku webu *Search Engine Land*. Jedná se sice o již poněkud zastaralé informace, ale nabízí alespoň základní náhled do toho, jak *Google News* funguje: „*V mnoha dgrafiskusích s vydavateli mluvíme o hodnocení, procesu shlukování a o tom, co se snažíme dělat. Existuje žebříček zpráv, kde zobrazujeme nejlepší zprávy dne v dané kategorii. Jaký je souhrnný zájem vydavatelů o daný příběh? Co mají všichni na titulní stránce? Co redaktoři společně považují za top příběh dne? Podle toho se budou řídit výsledky.“* [3].

Klíčové poznatky o strategiích *Google News* pro zobrazování článků jsou:

* Zprávy jsou shlukovány k sobě (to prakticky znamená, že se pod sebou zobrazují články podobných nebo dokonce stejných témat, viz *Obrázek 1: Shlukování zpráv na Google News*).
* Na postavení článku má vliv např. čerstvost zprávy nebo relevance pro danou lokaci.
* *Google News* využívají svá data z vyhledávání a tzv. page rank (koeficient říkající, jak si stojí stránka ve vyhledávání vzhledem k ostatním).
* Uchovává kromě toho také kliky na články v rámci samotného *Google News*.
* Algoritmy mají přednost před lidmi. Z části kvůli efektivitě, z části také kvůli tomu, aby služba dodala lidem větší důvěru v to, že informace budou vyvážené a nezávislé.
* Články v *Google News* není možné hodnotit.
* *Google News* nabízí titulek a někdy také výňatek z článku.
* Služba *Google News* je dostupná na webu, mobilní zařízení s OS Android a je také zabudovanou součástí dalších aplikací společnosti *Google* jako prohlížeč Chrome.

Obrázek 27: Shlukování zpráv na Google News [3]. Sekce zobrazuje články, které se týkají amerických voleb [146].

* + 1. Apple News

Zatímco v minulosti se *Apple News* nelišilo příliš od *Google News*, v současné době služba *Apple News*+ více akcentuje svou schopnost dodat uživatelům jakousi přidanou hodnotu v podobě lepšího zážitku ze čtení článků. *Apple News* na rozdíl od *Google News* rovněž více propaguje poslouchání zpráv v audio podobě. Větší rozdíl je také v tom, jak je *Apple News* svázáno s hlasovým chytrým zařízením od společnosti *Apple Siri*. Asi vůbec největší rozdíl je v tom, že zatímco *Google* se snaží lidský faktor eliminovat, *Apple* naopak využívá redaktorů z řad skutečných novinářů, vybírajících články pro uživatele, výrazně více [4].

*Apple News* je dostupné pro desktopová a mobilní zařízení *Apple*.

* + 1. Flipboard

*Flipboard* [5] je spíše než na myšlence agregátoru jako *Google News*, založen na myšlence digitálního časopisu, který si ovšem uživatel sestavuje sám. Má obsahovat sbírky zajímavých příběhů, které čerpají z vybraných zpravodajských zdrojů, blogů a dalších stránek, které může do *Flipboard* zapojit sám uživatel. Ve *Flipboard* je možné si vybrat buďto témata nebo konkrétní weby, který chcete sledovat (nebo obojí).

*Flipboard* se pak stará o jakési „zamíchání“ zdrojů dohromady. Podobně jako *Apple News*, *Flipboard* dává důraz na uživatelský prožitek. V případě potřeby lze vedle zpráv vkládat také různá jiná média jako podcasty, tweety a videa a na stránkách je vidět o něco větší snaha zapojení sociálních sítí než v případě *Google News*.

Jako první si uživatel po registraci zvolí témata, která jej zajímají, články si pak ukládá do vlastního magazínu (vlastního seznamu článků). *Flipboard* zobrazuje titulky a výňatky článků. Ve *Flipboard* je možné články označit hodnocením ikonou „srdíčka“. *Flipboard* je dostupný pro web, iOS a Android.

* + 1. *Feedly*

*Feedly* [6] by se dalo stručně a výstižně shrnout jako „chytrá RSS čtečka“. Hlavním účelem *Feedly* je skutečně načítat obsah z RSS a zobrazovat jej ve formě vlastního vizuálního designu, kdy si uživatel může přizpůsobovat vizuální vzhled článku (hlavně např. výšku písma nebo druh fontu). *Feedly* ale také nabízí doporučení na aktuální články dle vložených zdrojů RSS.

Výčet článků je rozdělen na dvě části: „Me“ obsahuje články ze zadaného zdroje, avšak setříděné algoritmy, část „Explore“ pak nabízí články další. Míra přizpůsobení v tomto případě je buďto velmi malá anebo vůbec žádná. Ač jsem například zadal zdroje v RSS z českých zpravodajských webů, *Feedly* žádný jiný český článek v této sekci „Explore“ nezobrazilo. Články tak dle informací zobrazených ve *Feedly* byly vybrány hlavně na základě čtenosti a jakési „trendovosti“, jak *Feedly* uvádí i ve své uživatelské sekci „Know-how“. Zdroje a způsoby zpracování takovýchto dat jsou však opět utajeny. Není tedy jasné, zda vedle dat o klicích samotného *Feedly*, využívá *Feedly* například i sledování výskytů článků na sociálních sítích nebo data o čtenosti samotného webu poskytnuté třetí stranou.

*Feedly* zobrazuje v závislosti na tom, co je dostupné v RSS. Většina českých zpravodajských webů poskytuje v RSS pouze nadpisy a výňatek, např. Aktuálně.cz pak i u některých článků celý obsah.

Články v aplikaci hodnotit nelze.

*Feedly* je dostupné na webu a mobilních zařízeních s Android a iOS.

* + 1. *Pocket*

*Pocket* [7] je doplněk prohlížeče vyvíjený v současnosti společností Firefox. *Pocket* vznikl z dnes již neexistující aplikace *Read It Later* [8]. *Pocket* původně neměl zapracován v žádné ze svých částí doporučující algoritmus, ale později začal zobrazovat doporučené články na základě kategorií vybraných uživatelem. Tuto aplikaci je nutné také zmínit z důvodu jednoduché, ale uživatelsky lákavé myšlenky možnosti si jedním kliknutím uložit článek na později a mít pak k dispozici uložiště takových článků, které není závislé na původním zdroji zpráv.

*Pocket* se vypořádává s autorskými právy na text, kterým musí čelit každá podobná služba tím, že tím, kdo ukládá články je uživatel, a nikoliv samotná služba [7]. Podobně jako u čteček RSS obsahu si tak může dovolit zobrazovat text článku samotný bez přesměrovávání uživatele na původní zdroj jako v případě *Google*. Zatímco však RSS zpravodajských webů často neobsahují celé texty článků, zde je zobrazován a průběžně ukládán do mezi paměti aplikace i celý text článku.

Zajímavá se zde jeví také samotná technika čtení textu z HTML článku, kdy *Pocket* automaticky rozezná, co je text článku a co je už další sekce webu obsahující např. odkazy na další články nebo reklama a filtruje tak pouze text článku.

Články lze uložit do aplikace pouhým klikem v případě prohlížeče Firefox, kdy bývá *Pocket* nainstalován jako doplněk již při stáhnutí prohlížeče pro desktopovou verzi. Pro mobilní verzi je pak nutné si stáhnout samostatnou aplikaci *Pocket*. V jiných případech (např. při používání jiného prohlížeče) lze do aplikace články přidat zkopírování a vložením odkazu na článek.

Články v aplikaci lze hodnotit ikonou “hvězdičky”, avšak nikoliv pro potřeby doporučujících metod, ale pouze pro vlastní účely třídění a ukládání.

Aplikace je dostupná jako doplněk prohlížeče Mozilla Firefox, články se zobrazují na webu https://getpocket.com/ nebo jako mobilní aplikace.

* + 1. FlashNews

Českého zástupce na tomto trhu tvoří aplikace *FlashNews* vyvíjená skupinou okolo sportovního výsledkového webu Livesport [9]. Rozložení a design aplikace je nápadně podobný nástroji *Pocket*, *FlashNews* je však svým obsahem agregátor zpráv -- z tohoto hlediska tedy spíše podobný *Google News*. *FlashNews* kombinuje články ze zpravodajských webů s příspěvky na sociálních sítích, kdy však většinou vybírá nikoliv „obyčejné“ uživatele, ale účty vládních orgánů, institucí, např. městských úřadů a také známých osobností.

Aplikace *FlashNews* je dostupná na webu, Android nebo iOS.

Články lze hodnotit palcem nahoru a lze také hodnotit osobnosti, jejichž profily se v aplikaci vyskytují.

* + 1. Shrnutí

Z výše uvedených systémů lze pozorovat rozdílné přístupy k doporučování článků. Větší služby i přes svá propracovaná komplexní algoritmická řešení využívají také různých redakčních zásahů, aby například lépe reagovaly na aktuální události. Pro menší projekty je však takové řešení pomocí zapojení jakési lidské redakce, která pomáhá moderovat zprávy, špatně dosažitelné. Některé aplikace jako *FlashNews* obsahují tlačítka hodnocení. U těch, které nevyužívají, explicitní hodnocení, lze předpokládat, že využívají pro posouzení relevance prokliky na články a obzvláště v případě *Google* i mnoho uživatelských dat přebíraných z prohlížečů týkajících se jeho chování na internetu i jeho osobních údajů.

Aplikace *Pocket* přináší zajímavý způsob ukládání textu celých zpráv bez porušení autorských práv a licenčních podmínek, což by se mohlo hodit pro případné budoucí rozšíření tohoto nebo jiného podobného projektu snažící se o zavedení doporučujícího systému pro články nebo zprávy. Populární jsou dnes také aplikace, které lze nazvat jako „chytré čtečky“, jenž zobrazují pouze články ze zadaných zdrojů, avšak třídí je a seřazují je uživateli pomocí doporučujících algoritmů.

* 1. Akademické práce a články zabývající se podobnou tématikou

V této části jsou představeny klíčové akademické práce užitečné pro tvorbu tohoto doporučujícího systému. Tyto práce se často zabývají jinou problémovou doménou, avšak mnoho ze strategií a algoritmů je použitelných i pro doménu článků.

* + 1. Real-Times News Recommender System

V článku *Real-Time News Recommender System* [10] se autoři ze slovinského *Institutu Jožefa Stefana* zaměřují na doporučování novinových článků skrze systém zaměřen na uživatele. Jejich doporučující systém funguje na základě minulých návštěv uživatele, kontextu a popularitě článků. V rámci obsahových doporučení je kladen důraz na doporučování na základě kategorií namísto jednotlivých položek. Autoři tím chtějí zabránit nerelevantním doporučením vedoucím z přílišné specifičnosti doporučení položek. Dalším důležitým faktorem pro doporučení je pro autory čerstvost článků, která má v rámci jejich algoritmu přednost před relevancí. Pro kolaborativní filtrování byl v práci využita metoda SVM.

* + 1. News recommender system: a review of recent progress, challenges, and opportunities

Článek *News recommender system: a review of recent progress, challenges, and opportunities* (autoři Raza, Ding) je věnován vývoji doporučujícího systému článků založeného na neuronové síti [10]. Pro vyvážení výhod a nevýhod přístupů založených na uživatelích a na obsahu, autoři práce využívají hybridní přístup. Kromě technologické stránky se práce zaměřuje rovněž na sociologický aspekt doporučování novinových článků inteligentními systémy. Pro odbourání nepříznivého vlivu na uživatele autoři navrhují zajistit transparentnost a vysvětlování výsledků; zapojit uživatele do volby toho, co chtějí číst (nehnat se pouze za jeho kliky) nebo například ctít soukromí uživatele a jeho údaje o chování na internetu, které jsou zjistitelné z prohlížeče.

* + 1. Offline and online evaluation of news recommender systems at swissinfo.ch

Článek *Offline and online evaluation of news recommender systems at swissinfo.ch* se zaměřuje na kombinaci online a offline přístupu trénování modelů strojového učení a evaluace. V offline případě se využívá statického datové sady, zatímco v online systému je model znovu trénován nebo aktualizován novými články za běhu systému.

* + 1. User trends modeling for a content-based recommender systém

Více netradiční způsob je zde zastoupen článkem *User trends modeling for a content-based recommender system* [11] týmu z íránské *Shahrood University of Technology*. Autoři se snaží k doporučování článků přistupovat jako k problému modelování trendů a tento přístup navrhují z důvodu dynamické povahy měnícího se prostředí aktuálních článků, obzvláště v kombinaci se sociálními sítěmi. Pro řešení využívají u doporučujících systémů ne tak tradičního statistického přístupu v podobě procesu čínské restaurace. Tento proces inspirován situací z reálného světa má 2 základní pravidla: 1. první zákazník vždy vybere první stůl. 2. „n-tý“ zákazník si vybere první neobsazený stůl s určitou pravděpodobností a obsazený stůl s jinou pravděpodobností. [12] Čím je stůl více obsazený, tím je populárnější – odtud tedy autoři vychází pro modelování popularity článků. Do výpočtu pravděpodobnosti přitom zasahuje například počet osob sedících u daného stolu. Metody byly evaluovány na krátkých výňatcích zpravodajských serverů *New York Times*, *BBC* a agentury *AP* zveřejněných na jejich profilech v rámci sociální sítě Twitter.

* + 1. Hybridní doporučující systém pro doporučování filmů s použitím expertního systému

Na *Ostravské univerzitě* a *Katedře informatiky a počítačů* vnikla diplomová práce Vladimíra Fojtíka zabývající se doporučujícím systémem pro doménu filmů. Z této diplomové práce *Systém pro podporu rozhodování v prostředí webových aplikací* poté vznikl i článek *A hybrid recommender system for recommending relevant movies using an expert system*.

Výstupem tohoto systému byl poté web https://app.predictory.dev.

Tento doporučující systém se dělí na základní části:

* Uživatelské rozhraní.
* Doporučující modul.
  + Kolaborativní filtrování.
  + Metody obsahového filtrování.
* Expertní systém.
* Sběrač informací.

Systém využíval open-source databázi filmů *MovieLens* sestaven a podporován výzkumným týmem *GroupLens* z *University of Minnesota* [13].

Zatímco využití kolaborativního filtrování a obsahového filtrování pro tuto doménu je součástí i mnoha dalších akademických i komerčních prací, tento systém nabízí zajímavou vlastnost z hlediska kombinace těchto tradičních doporučujících metod a metod expertních systémů.

* + 1. Multimodální doporučující systém

Matěj Jakimov z *Masarykovy univerzity* v Brně implementoval ve své diplomové práci *Multimodální doporučovací systém* [14] „proof-of-concept“ obecného doporučujícího systému, kde jsou popsány nejpoužívanější doporučující algoritmy založené na uživatelích i obsahu a je naznačena implementace škálovatelného a obecně použitelného doporučujícího systému.

* + 1. Doporučování na Amazon.com (Item to item collaborative filtering)

Článek z časopisu *Industry Report* [15] Grega Lindena a Jeremyho Yorka pracujících v době vzniku pro jedny z průkopníků doporučujících systémů v *Amazon*, nabízí nahlédnutí do jednoho z prvních doporučujících systému svého druhu. Pro velké prodejce, jako je *Amazon*.com, je důležitý doporučující algoritmus škálovatelný na velmi rozsáhlé zákaznické základny a katalogy produktů, který vyžaduje pouze rychlé zpracování pro generování online doporučen a který je schopen okamžitě reagovat na změny v údajích uživatele a vytváří přesvědčivá doporučení pro všechny uživatele bez ohledu na počet nákupů a hodnocení. Na rozdíl od jiných algoritmů je kolaborativní filtrování mezi položkami schopno tuto výzvu splnit.

* 1. Knihy zabývající se doporučujícími systémy

V této podkapitole jsou představeny knihy zabývající se doporučujícími systémy obsahující užitečné poznatky vzhledem k této práci.

* + 1. Practical recommender systems

Jedním z nejobsáhlejších titulů na téma doporučujících systémů je *Practical recommender systems* Kima Falka [16]. Tato kniha popisuje používané techniky a strategie od těch známých (algoritmy kolaborativního filtrování) po ty méně známé jako například sledování času stráveného na stránce, sledování tlačítek sdílení, rychlosti akcí, které uživatel provádí na stránce a podobně.

* + 1. Algoritmy inteligentního webu

*Algorithms of the Intelligent Web* [17] je dalším příkladem knihy sloužící pro získání znalostí k vytváření aplikací, které shromažďují, analyzují a využívají obrovské množství dat, jež uživatelé při používání webu zanechávají.

V knize je čtenář seznámen s doporučujícími systémy, vyhledáváním a řazením, automatickým seskupováním podobných objektů, klasifikací objektů nebo prognostickými modely. V knize jsou rovněž představeny knihovny s otevřeným zdrojovým kódem v jazyce Java.

* 1. Otázka autorských práv u doporučujících systémů s články

Z hlediska ochrany duševního vlastnictví není etické, ani legální přebírat veškerý obsah článků webů, pokud jej web nenabízí ve svém RSS kanálu. Proto také i mnoho v minulé části uvedených služeb pro agregování článků neobsahuje články v celém znění a obsahuje např. pouze nadpisy a výňatky zpráv. Z hlediska doporučujících algoritmů toto může být nevýhodou v případě algoritmů založených na obsahu, které by mohly využít i celý obsah článku a na srovnání celého textu článků.

Zákony nejsou v tomto příkladu zcela průhledné, lze ale předpokládat, že překročení hranice mezi použitím „fair-use“ a porušením autorských práv se rovněž týká ukládání dat do databáze, a to i bez ukázání dat veřejnosti [18] [19].

Jako teoreticky možné řešení se jeví ukládání obsahu webů pouze do mezi paměti při vykonávání algoritmů – tedy stažení obsahu článků na pouze určitou dobu. S tím se ale pojí nevýhoda ztráty už tak cenné výkonosti a náročnější technologické řešení.

* 1. Netflix

Vzhledem k tomu, že komerční doporučující systémy podléhají firemním tajemstvím, není leckdy lehké nahlédnout do jejich vnitřní architektury. Některé velké technologické společnosti však pomyslnou pokličku poodkryly více. *Netflix* odtajnil část své architektury již kvůli v oboru strojového učení slavné soutěži *Netflix Prize*. V dalších letech ale pokračoval se sdílením znalostí i například v rámci svého blogu *Netflix Tech Blog* [20].

Některé zajímavé informace poskytnuté v rámci tohoto blogu jsou například takovéto:

* Online výpočty podléhají časovým omezením HTTP požadavků. Např. seřazení položek podle žánru. Aby byl doporučující systém co nejvíce flexibilní, existuje trénování modelů strojového učení prováděné online [21] (v takovém případě se provádí nad menšími datovými sadami).
* Offline výpočty – provádí se nad daty čítající i miliony položek. Zde se provádí hlavní trénink modelů strojového učení. Využívá se dávkové zpracování [21] a provádí se zde vědecké experimenty [22].
* „Nearline“ vrstva slouží např. pro aktualizace doporučení po otevření nové položky. Tato vrstva má na starosti ukládání a načítání z mezi paměti („caching“) [21].
* „Předvýpočty“ probíhají offline, poté následují méně náročné části online [21] prvku do konkrétní shluku již může být provedeno online. Stejně tak například v případě shlukovacích algoritmů se provedou identifikace shluků offline, ale přiřazení prvku do konkrétní shluku již může být provedeno online.
* Některé výsledky doporučení však bývají kompletně předem vypočítané offline, popř. mohou být předem vypočítány, ale aktualizovány pro přidání čerstvosti výsledků [21]. Ač není zmíněn konkrétní příklad, zde se nabízí například možnost předem vypočítat několik doporučených článků offline, jejich uložení a následně seřazení řadícími algoritmy třídy learn-to-rank za běhu (více o nich v následujících částech práce).
* Pro cache využívá *Netflix* například technologie jako *Cassandra* nebo *EVCache*, což je mezi paměť založená na ukládání párů klíč-hodnota („key‑value“) [21]. Jako modernější alternativa se nabízí například *Redis*, který postupně *Netflix* rovněž integroval do své architektury [23]. Pro další data *Netflix* využíval například relační databázi *MySQL* (jako pro uložení dat, které budou zpracovány později).

Pro doporučující systémy byla průlomovou historickou událostí slavná soutěž *The Netflix Prize*. Společnost *Netflix* se chtěla zasadit o zlepšení svého doporučujícího systému filmů a seriálů, a tak nabídla milion dolarů týmu, či jednotlivci, který dokáže překonat jejich tehdejší doporučující algoritmus *CineMatch* alespoň o 10 %. Dokázal to nakonec mezinárodní tým *BellKor*, tehdy pomocí jejich algoritmu *Pragmatic Chaos*, jenž dokázal při testování na skrytých položkách filmů a seriálů (neposkytnutých při zadání oproti množině využitelné k testování a trénování modelů strojového učení) dosáhnout chybové hodnoty odmocniny ze střední kvadratické hodnoty (RMSE) 0,9525 [25]. *Netflix* však jejich řešení nakonec nevyužil z důvodu přílišné komplexnosti řešení [26] (jakožto druhý důvod by se daly identifikovat tehdejší výrazné změny na trhu, kdy streamování začalo nahrazovat DVD, což znamenalo celkové podstatné změny v jejich podnikání včetně prezentace a marketingu). Ani společnost jako *Netflix* si nemohla dovolit vyhradit zdroje na vytvoření hladké produkční verze „ideálního“ vítězného algoritmu a jejich řešení kombinující více doporučujících přístupů zároveň. Z toho lze vyvozovat další záludnost doporučujících systémů – nejpřesnější řešení často není možné použít, protože by bylo příliš časově nebo prostorově náročné.

Z těchto poznatků si lze převzít, že ač má *Netflix* k dispozici široké možnosti v rámci zdrojů (myšleno např. servery, ale i programátorské týmy a finance), některé doporučující metody mohou být natolik náročné, že i přes jejich přesnost musí dělat kompromisy. Není možné postavit systém, který by se takřka v reálném čase adaptoval na změny a určitá část výpočtů, či algoritmů, musí být vykonávána na pozadí. Vhodné je identifikovat ty části systému, jejichž výpočty a algoritmy nejsou tak časově, či prostorově složité a ideálně za využití mezi paměti je nechat reagovat v reálném čase (online) na akce uživatele.

V závislosti na potřebě se u doporučovacích systémů používá buďto ověřování typické pro strojové učení, kdy máme rozdělení dat na trénovací a testovací (popř. také validační) a popř. také ještě můžeme provádět křížovou validaci anebo také alternativní způsoby. U *Netflix Prize* byl však také navržen nový způsob, který měl lépe ověřit výkon doporučovacích systémů. Pro trénovací sadu bylo možní využít více než 95 % objektů a malá porce dat poté měla rozhodnout o výkonnosti modelu. 1,36 % bylo tedy v tréninku využito pro selekci modelu a dalších 1,36 % pro ověření, které však bylo ještě rozděleno na to, kdy výsledek evaluace modelu byl ukázán a kdy dokonce ani výsledek nebyl ukázán.

1. návrh systému

Oblast doporučujících systémů je poměrně náročná na návrh systému. Uživatelé vyžadují od doporučujících systémů nízkou odezvu, a aby jejich interakce se systémem probíhala zdánlivě v reálném čase. Mnohé doporučující algoritmy jsou však časově (i prostorově) náročné.

Řešení návrhu doporučujícího systému v rámci této práce bylo přizpůsobeno rozsahu tohoto projektu, a možnostem finančních. Vzhledem k tomu, že je však pravděpodobné, že v budoucích letech bude pokračovat trend, kdy doporučující systémy již nejsou doménou technologických gigantů, může tohle řešení posloužit jako inspirace pro menší doporučující systémy určené menším a středním firmám, příspěvkovým organizacím apod.

Při snaze implementovat tento doporučující systém se ukázalo, že i přes čím dál dostupnější možnosti hostingů serverů a cloudů a výhodné nabídky, které poskytují zejména velké technologické firmy jako *Amazon* nebo *Google* studentům, může být návrh takového systému v doméně s minimálními finančními prostředky a při vyvíjení jednotlivcem, či menším, méně zkušeným týmem, prakticky velmi náročný úkol. Samotný kód je totiž možné optimalizovat pro lepší výkon, stejně jako optimalizovat náklady spojené se servery, či optimalizovat kód pro menší využívání zdrojů serverových poskytovatelů a tím pádem minimalizovat i náklady na provoz. Takovéto optimalizace však mají zase za následek prodlužování doby vývoje řešení, což může být problém zejména pro začínající start-up firmy, které si mohou dovolit pracovat na do produkce zatím nevydaném softwaru, tedy nevýdělečně, pouze omezenou dobu (uvádí se například 9 až 12 měsíců [24]).

Většina moderních doporučujících systémů používá hybridní přístup kombinace doporučeních založených na obsahu, kolaborativního filtrování (doporučení založená na uživatelích) a dalších přístupů. Hybridní systémy ukázaly v praxi lepší výsledky než pouze orientace na jeden ze způsobů doporučování [25] [26]. Hybridní přístupy pomáhají řešit některá zásadní úskalí doporučujících systémů, jakými jsou například problém studeného startu (jak bude systém doporučovat na začátku, jestliže ještě nemá data o chování uživatele) [16], problém řídkosti dat (jak bude systém doporučovat na základě ostatních uživatelů, jestliže je ostatních uživatelů málo) nebo problém úzkého hrdla znuvupoužitelnosti ve znalostním inženýrství (jak můžeme znovupoužít znalosti pro nový úkol) [27]. Tento hybridní přístup byl tedy ve vhodné podobě do této práce také zahrnut.

* 1. Optimalizace návrhu

Možné způsoby, jak se potýkat s časovou a prostorovou složitostí doporučujících systémů jsou představeny v následující části. Jedná se o znuvupoužitelnost zavedených knihoven, optimalizace využívání paměti, využívání nativních metod knihoven pro rutinní operace namísto programování vlastních, využití databázových indexů, využití moderních technologií pro dávkové zpracování dat apod. Řada z těchto technik jsou obecnými metodami využívanými v oblasti softwarového inženýrství. Další z těchto technik jsou ty speciálně užitečné pro oblast doporučujících systémů založené na znalostech sdílených vývojáři stávajících doporučujících systémů nebo zkušenostech z tvorby praktické části této práce.

* + 1. Optimalizace využívání paměti, optimalizace metod, nativní metody

Týká se např. rutinního procházení prvků, prohledávání. Při vyvíjení složitějších systémů, mezi které doporučující systémy patří, je vhodné využívat vyšší úrovně abstrakce nástrojů a znuvupoužitelnost [28]. Pokud jsou využívány knihovny třetích stran, doporučuje se využívání jejich nativních metod, které bývají optimalizovány ve většině případů lépe než “ručně” kódované metody rutinních činností [29] [30]. Nativní metody jsou ve své implementaci často paralelizované. Toto platí napříč různými programovacími jazyky.

* + 1. Využití databázových indexů

Jedná se o typický způsob urychlení prohledávání databáze nad častými dotazy, který je časově i implementačně nenáročný a není při něm nutné měnit současnou strukturu dat.

* + 1. Metody s podporou distribuovaného nebo paralelního zpracování

Pro méně zkušené programátory může být lákavé využít uživatelsky přívětivějších řešení populárních v různých návodech, ale často i knihách. Jedná se ale o další z rozporuplných problémů v oblasti doporučujících systémů (a v tomto případě i datové analytiky). Méně zkušený jednotlivec, či tým, chce co nejrychleji doručovat řešení a doručovat je do produkce. Některá řešení se však mohou ukázat časem jako nevhodná a také náročná na změnu, kdy se nemusí jednat jen o problém refaktoringu, ale i architektury systému. Typickým případem je například hojně využívaná knihovna programovacího jazyku *Python Pandas*, která poskytuje datové rámce v rámci datové struktury datových rámců („dataframe“). „Dataframe“ nabízí ukládání i zpracování dat před tím i po tom, co jsou data vyhodnocena, či zpracována nějakým algoritmem a tato knihovna se stala velice populární v oblasti datové analýzy a strojového učení [31]. Výhodou této knihovny je především její flexibilita použití spojená s vysokým množstvím metod pro zpracování dat, které nabízí a široká podpora dalšími knihovnami často používanými v toto oblasti jako například matematickou knihovnou *Numpy* (např. pro možnosti převádění datových struktur *Pandas* na datové struktury *Numpy*), kterou zase využívá mnoho implementací modelů a metod strojového učení (vstupní datové struktury *Numpy* jsou například používány v rámci knihovny *Sklearn*).

Nevýhodou datových rámců *Pandas* je však její horší výkonost, která se může projevit obzvláště v případě potřeby zpracovávání velkého množství dat v dimenzích označovaných poměrně vágním pojmem “Big Data” (řekněme desítky tisíc záznamů a více). Některé alternativní knihovny pro datové struktury *Pandas* pro *Python* jsou například knihovny *Dask* nebo *Ray*, popř. ukládání a operace dat pomocí datových struktur komplexních platforem jako distribuovaná platforma pro ukládání událostí a zpracování proudů dat *Apache Kafka* (resp. implementace API knihovny pro *Python* *PyKafka*) nebo distribuovaný open-source nástroj pro dávkové zpracování *Apache Spark* (resp. pro *Python* *PySpark*) přitom nabízejí nejen lepší výkon nad velkými daty [32] [33] [34]. Další možností je rozšíření architektury systému o efektivní zpracování velkých datových struktur například pomocí software pro distribuované zpracování a ukládání velkých dat jako *Hadoop* nebo *Hive*. Tyto dva software nejsou, jak by se mohlo zdát, konkurenty všech výše zmíněných platforem. *Hadoop* je možné brát díky velkému průniku funkcionalit jako alternativu k *Apache Spark*, ale *Hadoop* naopak může spolupracovat s *Apache Kafka* v rámci jednoho systému, kdy se např. *Apache Kafka* může starat o zpracování proudů dat „v reálném čase” a pro uložení, následné další zpracování velkých dat, monitoring apod. se může využít *Hadoop* [35].

Určité východisko mohou být v tomto případě knihovny *Python*u jako *Modin*, které lákají na možnost využití efektivních datových knihoven založených na paralelním zpracování jako *Dask* a *Ray* a to za pomocí jen několika řádku kódu (změny importování třídy datových rámců z knihovny *Pandas* na třídu datových rámců od *Modin*). Tato knihovna převede automaticky metody *Pandas* na metody *Dask*, či *Ray*. Pokud intergace není možná, ponechají se pomalejší metody *Pandas*. Bohužel, jako se ukázalo i v případě tohoto projektu, změna nemusí být vždy tak přímočará a jednoduchá, pokud je projekt již v pokročilejší fázi. *Modin* totiž nepokrývá některé metody *Pandas*. Intergace *Dask* a *Ray* do vývojového prostředí bylo tak natolik časově náročné, že bylo odloženo na případné pozdější rozšíření systému.

Paralelního přístupu bylo v tomto projektu tedy využito až v kombinaci s technologií *Advanced Message Queuing Protocol* (dále AMQP) a implementací *RabbitMQ* (viz část 2.1.6) při zachytávání akce uživatele v tomto toku událostí:

* Uživatel udělá akci, která má ovlivnit doporučení (např. ohodnotí článek).
* JavaScript Object Notation (dále JSON) obsahující data uživatele se uloží do fronty *RabbitMQ*.
* Fronta je obsloužena konzumentem.
* Je vytvořeno vlákno.
* Program zpracovává nová doporučení na základě předchozích akcí uživatele.
  + 1. Využití Redis

*Redis* je v jádru implementačně lehký a optimalizovaný nástroj pro ukládání a práci s daty [36]. Základní použití *Redis* je mezi paměť s ukládáním informací v jednoduché struktuře klíče a hodnoty (podobně jako např. u slovníku *Python*u). Možnosti *Redis* jsou ale také jiné a mnoho z případů užití má svůj průnik s doporučujícími systémy. *Redis* například obsahuje datové struktury jako množiny („sets“) a množinové operace. Blogový článek Davida Maiera na webu *No SQL Geek* [37] například ukazuje, jak je možné implementovat jednoduchý doporučující systém s využitím pouze metod a datových struktur *Redis*. Byť má tento způsob pochopitelná omezení (např. při integraci s modely strojového učení by musela být data převedena do jiné datové struktury), jedná se o zajímavý způsob využití *Redis* nejen jako uložiště, ale i jako plnohodnotné datové struktury s vlastními matematickými nástroji.

V tomto projektu je *Redis* využito jako jednoduché a rychlé mezi paměti. V případě, že uživatelem není provedena akce, která ovlivní načítání výsledků, nemá smysl volat externí API („Application Programming Interface“), ani vykonávat dotazy nad relační databází, místo toho lze využít načítání z *Redis* mezi paměti. Teprve po vykonání akce, která změní výsledky (např. přidání klíčového slova), jsou data v *Redis* aktualizována programovým skriptem (popř. záložní API). Pomocí struktury SET a množinových operací je u filtrování aktuálních článků rychle vybrán průnik mezi relevantními články podle doporučující metody a nejnovějšími články.

* + 1. Mikro servisní architektura

Mikro servisní architektura je výhodná pro menší a středně velké projekty a korespondující vývojové týmy vzhledem ke snadnější správě jednotlivých modulů. Další výhodou je snadnější kontrola využívání placených prostředků serverů, reorganizace nasazených částí projektu, dočasné odstavení nefungujících částí, testování menších částí, či dokonce využití různých poskytovatelů hostujících služeb pro jejich výhodnější ekonomické využití (namísto jedné finančně nákladné aplikace s mnoha procesory a velkou pamětí je možné využít více levných, či dokonce zdarma dostupných serverových prostorů). Nevýhodou této architektury je pomalejší komunikace skrze internet, při stahování souborů, či při komunikaci různých modulů po síti. Jedná se o rozpor obvyklých uživatelských nároků pro fungování systémů v „reálném čase“. Tato nevýhoda je částečně řešitelná například s využitím předchozí zmíněné technologie *Redis*, či pomocí offline vyplňování údajů do relační databáze (podobně jako to dělá i *Netflix*, viz předchozí kapitola).

Poněkud opačným problémem u konkrétního projektu poté je, zda při distribucí za účelem ušetření finančních prostředků nedošlo k nedostatečnému využívání výpočetních kapacit.

S mikro servisní architekturou souvisí i následující podkapitola o zasílání zpráv.

* + 1. Zasílání zpráv

V rámci doporučujících systémů se objevuje mnoho případů použití zasílání zpráv mezi programy. Zasílání zpráv je vhodné využít v případech, kdy je vyžadována spolehlivost pro doručování zpráv [38].

Pro tento úkol je možné využít množství platforem. Dnes je mezi programátorskými týmy populární například platforma *Kafka*, *IBM M*, či *Active MQ*. Pro účely tohoto projektu bylo využito další populární platformy *RabbitMQ* operující na protokolu AMQP [39] založeném na komunikaci rolí (vydavatel – spotřebitel, angl. publisher‑consumer) a to konkrétně pro zaslání zpráv buďto parsovacím skriptem, který po stažení aktuálních nových textů článků uvědomí část systému nazvanou interně jako „Prefillers“ (či „Updaters“). Jedná se o „předvyplňovací“ skripty, které předem zpracují text, či přímo najdou pomocí modelů založených na obsahu podobné články a doplní je v podobě datové struktury JSON („JavaScript Object Notation“) do databáze. Tato technologie také zachytává akce uživatele jako ohodnocení nebo přidání oblíbených klíčových slov konzument fronty pak vyvolá vhodné části programu obsahující doporučující metody, které mají reagovat na danou akci.

Struktura JSON byla zvolena z důvodu zachování integrity s komunikací v rámci REST rozhraní, která probíhá rovněž pomocí JSON a je v rámci tohoto systému také využívána v podobě záložního API, pokud nejsou k dispozici předem vyplněné údaje dostupné ani v databázim ani v *Redis* cache. JSON formát je zavedená technologie, nabízí navíc možnosti zavedení rozšíření JSON-LD, jenž je spravováno komunitou W3C, jenž se při vývoji tohoto formátu zaměřuje na použitelnost pro oblast umělé inteligence a strojového učení [40].

* + 1. Kombinace online a offline přístupu doporučování jako cesta k lepšímu uživatelskému zážitku

Strategie doporučování se odvíjejí od vlastností jednotlivých modelů. Obecně se dá využít třech základních přístupů:

* Doporučovat za běhu – tj. nové, čerstvé výpočty provedené na vyžádání zahrnující například nové procházení a porovnání všech položek datové sady (např. všech textů článků).
* Uložení předem zjištěných doporučení a uložení do databáze (uložení podobností).
* Uložení přímých „surových“ výstupů modelů v podobě vektorů, či matic – do databáze nebo do rychlé mezi paměti (viz např. výše popisovaná technologie *Redis*), viz např. přístup představován samotnou úspěšnou technologickou firmou *Netflix*.

Často se uvádí pro rozdělení těchto druhů doporučování pojmy „off-line“ a „online“. Toto označení však může být jak pro laickou veřejnost, tak pro programátory matoucí – skutečně „off-line“ se může odehrávat například předzpracování textu nebo někdy trénování modelů, i tady ale dříve nebo později musí docházet ke stažení aktuálních datových sad nebo nahrání aktualizovaných výpočtů a zpracování. U výpočtu nových doporučení je interakce s informacemi obsaženými v databázi ale ještě častější; vhodnější je v tomto ohledu tedy například anglický výraz „on the fly“ pro doporučení za běhu a rozdělení konkrétních činností prováděných „off-line“ např. na část experimentů, evaluací a srovnání modelů strojových učení, „předvyplňování“ vypočtených doporučení do databáze apod.

Například, pokud jsou zrovna staženy nové články; v takovém případě je vzhledem k povaze aktuálnosti článků zpravodajského serveru vhodné články zobrazit rychleji i při riziku, že by nové články nemusely být zahrnuty do doporučení vůbec anebo by doporučení podobných článků mohlo trvat o něco déle.

Zasílání zpráv poté umožňuje realizovat oddělení doporučujících algoritmů a metod. Rychlejší (ale spíše méně přesnější) metody a algoritmy je možné využít pro zpracovávání doporučení na vyžádání uživateli okamžitě (např. přes REST rozhraní), zatímco výpočetně náročnější modely a hybridní kombinace modelů lze postupně předem vyplňovat do databáze.

Problém takového přístupu vězí jednoduše v tom, že při zásadním vylepšení modelů, či zpracování texů je nutné znovu zpracovat všechny textové dokumenty (v tomto případě články) a k nim všechny potřebná načítání souborů, matematické výpočty, řazení výsledků a další činnosti. Spolu s využitím distribuovaného zpracování se však jedná o jedno z nejzásadnějších vylepšení časové náročnosti problému doporučování relevantních textů a uživatelského zážitku. Na rozdíl od distribuovaného zpracování za pomocí počítačových clusterů, je vypočítání doporučení předem a jejich uložení možné realizovat i u menších projektů.

Protože v takovém případě uživatel nečeká na výsledek za běhu systému, můžeme například zčásti obětovat časovou náročnost ve prospěch menšího využívání paměti a procesoru při výpočtech. Například budeme iterovat nad databázovým objektem po jednotlivých dokumentech, namísto nahrání všech textů najednou do paměti.

U velkých datových sad je však tento problém vypočítávání hodnot předem přesto vhodné řešit pomocí distribuovaného zpracování. *Netflix*, či *Google* například potřebují neustále aktualizovat své výsledky doporučování, či vyhledávání, a přitom nabízet uživateli rychlé výsledky. Toho je možné dosáhnout pomocí:

* Co nejlepší optimalizace algoritmů.
* Co nejvhodnějšího rozdělení doporučení – většinou kombinace již předem vypočítaných podobností a aktualizovaných výstupů modelů (matic, vektorů), jenž poskytují rychlejší výpočty za běhu [21].
* Využití indexace, mezi paměti a technologií jako *Redis* (viz výše).
* Distribuovaného zpracování – jak výpočtů za běhu, tak předem vypočítaných výsledků.

V datové sadě asi 8 tisíc článků například i po optimalizacích trvalo vypočítání podobností článků s využitím výstupů metod TF-IDF („Term Frequency‑Inverse Document Frequency“, Doc2Vec, Latentní Dirichletovy alokace (dále LDA) a Word2Vec v rozmezí 5 vteřin až 70 vteřin v závislosti na nastavení modelů (např. omezení počtu témat, omezení počtu slov, která se berou v potaz apod). Algoritmus TF‑IDF nevyžadující natrénovaný model pak dokázal při experimentování spočítat podobnosti v rozmezí asi 5 až 25 vteřin. Více komplexní varianty modelů (např. LDA natrénováno na velkém množství témat) se tedy nacházely i za běžnou hranicí časového limitu HTTP protokolu, který bývá u většiny serverů nastaven na původní hodnotu 30 vteřin. I po obejití tohoto problému však nastává problém uživatelský. Průzkumy z oblasti internetového marketingu například ukazují, že uživatelé nechtějí čekat na webových stránkách týkající se e-komerce déle než 2 až 2,5 vteřiny před objednáním zboží – až poté jsou ochotni déle čekat (10 až 15 vteřin) na další části nákupu jako zpracování platby [41] [42].

V rámci domény novinových článků není průzkumů tolik, chyba v načítání způsobena akcelerátorem načítání stránek *Google* AMP, ale poskytla zajímavá data, kdy až 53 % uživatelů velkých internetových zpravodajských a magazínů jako Forbes, Washington Post nebo The Guardian odmítlo čekat déle než 3 vteřiny na načtení stránky [43].

* + 1. AJAX technologie

Pro zajištění interaktivity se systémem a lepší uživatelský zážitek je vhodné použít technologii AJAX založenou na jazycích JavaScript [44] a XML díky které nemusí docházet k obnovení stránky po každé uživatelské akci. Obzvlášť užitečné je využití tohoto v případě opakujících se uživatelských voleb preferencí jako uživatelského hodnocení, přidávání oblíbených kategorií, oblíbených klíčových slov a podobně.

* + 1. Využití existujících redakčních systémů

Pro účely této práce byl využit redakční systém vyvíjen za účelem mé bakalářské práce, kdy sloužil pro správu varovných zpráv [45]. Redakční systém poskytne jednoduchý způsob spravování příspěvků například v případě potřeby odstranit některé příspěvky. Existence takového systému pro správu obsahu (CMS systém) při rozšíření rozsahu projektu možné například využít administrativních pracovníků namísto programátorů pro správu příspěvků a moderování příspěvků.

* + 1. Využití PostgreSQL

Ač bylo v původním redakčním systému Varovného systému ČR využito systému pro správu databáze *MySQL*, vzhledem k levnějším nákladům, či dobré podpoře nástrojů datové analýzy a dolování dat [46] byla zhruba ve třetině tvorby projektu provedena migrace na *PostgreSQL*. Migrace z *MySQL* na *PostgreSQL* byl v případě tohoto projektu poměrně hladkým a rychlým procesem.

*PostgreSQL* je u podobných projektů vhodné využít i díky uvolněných licenčních omezení, resp. riziku „vendor lock-in“ [47].

Ač je *PostgreSQL* finančně výhodnou volbou pro menší projekty, tento systém pro správu databáze využívají i větší společnosti jako *Netflix*, Uber, Spotify, Meta Inc. (pro Instagram), Reddit apod [48].

Mimo *PostgreSQL* bylo pro účely trénování některých modelů strojového učení, sestavování slovníků nebo při využívání evaluačních metod, využito také dokumentově orientované databáze *MongoDB*, která byla pro tento účel vhodnější.

* 1. Use Case

Níže přiložený use case diagram zachycuje případy užití pro 3 základní role v systémů: administrátor, správce obsahu a uživatel. I v rámci menšího doporučujícího systému je vhodné obsahu rozdělit technickou administrátorskou roli a administrativní, či redakční roli správce obsahu vzhledem ke zjevným rozdílům kvalifikace takových případných pracovníků. U středních a větších systémů se nabízí role analytiků, kteří by měli přístup například k nástěnkám shrnujícím statistiky systému.

Obrázek 28: Use case diagram tohoto projektu

* 1. Architektura

Systém se skládá z těchto hlavních částí, pro které byl pro rozlišení zvolen název:

* **News-Parser**: modul založen na jazyku *Python*, využívající knihovny jako *BeautifulSoup*, sloužící pro stahování RSS obsahu z *Idnes* a následně plného textu z HTML.
* **Core**: hlavní modul systému obsahující doporučující metody, reagující na předvyplnění některých polí po stáhnutí článků a na akce uživatele. Obsahuje konzumenty zpráv z *RabbitMQ* fronty, které pak spustí metody v rámci dalšího kódu systému.
* **Fresh API**: modul obsahující některé časově méně náročné doporučující metody. Nyní slouží v případě nefunkčnosti nebo zaneprázdnění části *Core*. Vzhledem k časovým omezení HTTP protokolu zde však nejsou všechny doporučující metody.
* **Moje Články:** na PHP založený web a jednoduchý redakční systém, který zdědil část funkcionalit z projektu Varovný systém mé bakalářské práce. Úkolem je zobrazování článků a hodnocení, administrátorská správa, odeslání zpráv do *RabbitMQ* fronty ohledně nových akcí uživatele.
* **Databáze:** *PostgreSQL* relační databáze obsahující jak údaje o článcích, uživatelích, hodnoceních, tak některá aktualizovaná doporučení, která jsou vypočtena modulem *Core*.
* **Redis:** slouží jako mezi paměť pro web Moje články a pro rychlý přístup k doporučením.
* **CI/CD Pipeline:** pro průběžné doručování a testování nových změn slouží GitHub, testovací prostředí *GitHub Actions*, či *Heroku Pipeline*. Testování je prováděno technologiemi *PHPUnit*, *Laravel Dusk*, *Pytest* nebo *Tox*.
* **Off-line experimenty, testování, trénování**: na počítači nebo na serveru bylo využito například technologie *Docker* a *MongoDB* pro tvorbu slovníků, byly zde provedeno natrénování slovníků a modelů strojového učení, které nepotřebují aktualizaci, či off-line testování.

Vzhledem ke komplexnosti systému bylo nutné využít vhodné úrovně abstrakce (podle principu RUP/OpenUP Elevate the Level of Abstraction [28]) pro zjednodušení systému, ale zároveň zachycení všech aktivních součástí systémů a použitých technologií. Bylo proto tentokrát upuštěno od standardů a UML diagramů a raději byl využit neformální diagram, který tomuto účelu posloužil dobře. Ve větším rozlišení si lze tento diagram prohlédnout v příloze.

Obrázek 29: Neformální diagram systému (v plném rozlišení v příloze)

1. implementace systému

Pro důvody zmíněné v kapitole 2 byl zvolen hybridní přístup založen na kombinaci více různých metod doporučování. V doporučování založených na uživatelích hrála zásadní roli metoda SVD. V doporučeních založených na obsahu bylo testováno a evaluováno několik metod zpracování přirozeného jazyka (TF‑IDF, Word2Vec, Doc2Vec, LDA), které byly v kombinaci s metrikami podobnosti (Kosinova podobnost a Jensen-Shannonova podobnost) využity pro hledání podobných textů na základě různých faktorů. Větší důraz byl přitom kladen právě na doporučení založených na obsahu.

* 1. Sběr dat

Pro sestavení vlastní datové množiny ze zpravodajského webu *Idnes*.cz bylo využito RSS kanálu webu a pro doplnění údajů technik webového scrapování a procházení („web scraping“, „web crawling“). Pro větší datovou sadu byly využity výstupy české *Wikipedie*. Důvody využití textů *Wikipedie* jsou uvedeny v podkapitole 3.3.2.

* + 1. RSS

RSS kanál webu *Idnes*.cz je členěn na hlavní zpravodajský kanál a následně má každá kategorie zpráv svůj oddělený kanál. Při stahování textů je tedy možné rovnou členit příspěvky do kategorií, což se dá využít jako podstatná informace pro metody doporučování zmíněné níže. RSS kanál *Idnes*.cz obsahuje v rámci jednotlivých položek obsažených v rámci RSS tagu *<item>*: titulek, odkaz, popis, datum publikování článku, popis (perex), název kategorie, komentář (zde je však uveden pouze odkaz na diskuzi) a následně různé meta informace týkající se multimédií (pro tento doporučující systém však irelevantní).

Obrázek 30: RSS kanál zpravodajského serveru iDNES.cz

Protože obsahuje *Idnes* poměrně velké množství kategorií, kdy má například každý kraj nebo sport svou kategorii, byly některé kategorie sloučeny do jedné tak, aby nedocházelo k přílišnému roztříštění článků a v některých kategoriích se nenacházelo příliš málo článků.

* + 1. HTML

Pokud došlo k určitým nekonzistentnostem v obsahu RSS a na webu (např. RSS obsahovalo rozdílný, neaktualizovaný text), či došlo na straně *Idnes* k chybě a text nebyl nahrán, či zde bylo špatné kódování, bylo využito i navíc scrapování HTML obsahu *Idnes*.cz za pomocí knihovny *Python*u *BeautifulSoup* [49].

* + 1. Výstupy z Wikipedie

Výstupy textů mezinárodních variant *Wikipedie* byly staženy ze serveru švédského spolku *Academic Computer Club* fungujícího v rámci *Umeå University* [50]. Dostupné jsou archivy v komprimovaném formátu *xml.gz2* a *xml.bz2*. Více o zpracování těchto výstupů a o jejich využitích se nachází v části věnované metodě LDA (3.3.10 a 4.3.4).

* 1. Doporučení založená na uživatelích

Doporučení založená na uživatelích poskytují příležitost pro identifikaci uživatelů s podobnými preferencemi. Pro svou rychlost a přesnost byl využit algoritmus SVD.

* + 1. Uživatelská hodnocení

Uživatelská hodnocení jsou v systému prováděna:

* Na základě Likertovy škály (1=nejhorší; 5=nejlepší).
* Na základě posouzení relevance (binární; 0 =relevanntí; 1 =irelevanntí).

Pro uživatelskou přívětivost je možné takto hodnotit články v případě Likertovy škály pomocí ikon „hvězdiček“, na které jsou uživatelé zvyklí z jiných systémů. Posouzení relevance může probíhat v testovacím „playground“ systému anebo pomocí ikony palce nahoru/dolů.

Hodnocení bylo rozděleno na tyto druhy, aby binární hodnocení posuzoval spíše relevantnost obsahu vůči jinému obsahu, hodnocení 1 až 5 poté může vyjadřovat celkovou spokojenost s článkem zahrnující více různých faktorů jako například zda nejenže článek odpovídá obsahu článku druhého, ale do jaké míry uživatele článek zajímá (článek například může být obsahově podobný s jiným, ale uživatele nezajímá, protože se mi nelíbí, jak je sepsán). Likertova stupnice je vhodnější pro přesnější identifikaci podobných uživatelů (tedy pro algoritmy založené na uživatelích), binární hodnocení zase pro posouzení relevance obsahových doporučení.

* + 1. Kolaborativní filtrování a SVD algoritmus

Kolaborativní filtrování pracuje se základním předpokladem, že pokud se jednomu uživateli líbí nějaká položka a druhému se ta stejná položka líbí také, je pravděpodobné, že pokud se v takových případech shodnou víckrát, bude se druhému uživateli také líbit položka, kterou hodnotil zatím pouze jeden z uživatelů. Kolaborativní filtrování je součástí velkých doporučujících systémů e‑shopů, hudebních a video streamovacích služeb i sociálních sítí. Typickým příkladem populární služby, která je ve velké míře založena právě na kolaborativním filtrování je například YouTube, kdy se využívá historie zhlédnutí videí, historie hledání a kontextu uživatele (věk, pohlaví…) [51]. Častý algoritmus využívaný v rámci je algoritmus SVD založen na maticovém rozkladu. U doporučovacích systémech se nejčastěji bavíme o tzv. „user-item“ matici, tzn. matici „uživatel-položka“. Nejčastěji se setkáváme s variantou, kdy sloupce jsou filmy, řádky uživatelé a hodnoty matice jsou hodnocení položek. U doporučovacích systémů je velmi často taková matice řídká, protože neobsahuje ani zdaleka všechny hodnocení. Často je také tato matice plná šumu. Pokud bychom existovali v ideálním světě, taková matice uživatelů a položek, např. filmů by dokázala dokonale reprezentovat určité druhy uživatel nebo filmů, jako např. „fanoušek sci-fi“ a „sci-fi filmy“. V idealistickém případě by pak šlo také pro nové uživatele s chybějícími hodnotami předpovědět hodnocení ještě nezhlédnutých filmů jako jednoduché lineární kombinace předchozích hodnocení uživatelů.

Obrázek 31: Naznačení geometrického významu SVD a dvou rotací.

V reálném světě se ale musíme potýkat například s tím, že si nejsme jsiti s jakou motivací uživatelé hodnotí položku (např. zda hodnotí, že to není jejich „šálek kávy“ nebo se snaží o „objektivní“, kritické zhodnocení kvality), stejně tak se uživatel mohl prostě překliknout a omylem ohodnotit položku příliš kladně/záporně Není tedy náhodou, že SVD metoda má své aplikace kromě mnoha jiných aplikací, také v oboru zpracování signálu nebo pro počítačové vylepšování obrázků, Je totiž velice cenná pro práci s šumem., Speciálním rozkladem na tři speciální matice s geometrickým významem: rotace v původní bázi (doméně), roztažení, rotace v nové bázi (.. SVD metoda pro doporučovací systémy byla ve vylepšené verzi algoritmu *SVD++* využita i v případě vítězného doporučujícího algoritmu v rámci *Netflix Prize* [52] a pro svou jednoduchost a rychlost byl využit i v této práci.

Zjevným problémem SVD je problém studeného startu („cold-start problem“). Jestliže se zde nacházejí hodnocení malého počtu uživatelů, metoda SVD nevyužívá dostatek informací pro smysluplné predikce hodnocení článků na základě množiny zatím neohodnocených článků. Částečnou záplatou problému chladného startu v tomto systému je samotné využití metod založených čistě na obsahu, kdy se uživatel zkrátka může přiklonit k doporučením na základě takových metod. Ačkoliv je SVD uváděno jako metoda patřící do skupiny kolaborativních [51], při experimentálním použití bez uživatelů (resp. když byly články hodnoceny pouze autorem textu) vykazovalo SVD poměrně dobré výsledky. Stálo by tedy za zvážení a průzkum, zda se nedá v případě nízkého hodnocení článků SVD omezit pouze na jediného uživatele. Aby však metoda SVD mohla být takřka plnohodnotně využita i pro srovnání více uživatelů, byla do databáze vygenerována hodnocení. Protože neexistuje žádná stávající datová sada, která by šla použít přímo pro hodnocení článků z *Idnes*.cz a čistě náhodně vygenerovaná hodnocení by mohla mít velmi nepředvídatelné výsledky, či dokonce učinit metodu SVD nepoužitelnou, hodnocení byla vygenerována na základě dat *American Press Institute* a *Reuters* [53][54]. Tyto instituce uvádějí čtenost článků v rámci různých demografických skupin. Údaje o čtenosti byly normalizovány a převedeny na hodnoty odpovídající Likertově škále a vhodně projektovány na kategorie, které odpovídají kategoriím z *Idnes*.cz (někdy byl vztah zřejmý, jindy, např. v případě, že v průzkumu byla kategorie „Science and Technology“ a *Idnes* přitom obsahuje oddělené odpovídající kategorie „Věda“ a „Technologie“, byla stejná data využita pro obě tyto kategorie). Pro randomizaci hodnocení byla poté k datům přičtena náhodná odchylka odpovídající Gaussovskému šumu.

* + 1. Doporučení na základě klíčových slov

Doporučení na základě klíčových slov probíhá na základě zadání množiny klíčových slov v sekci uživatelského profilu systému po registraci a přihlášení. Zadaná klíčová slova jsou poté srovnána s články pomocí metody TF-IDF založené na obsahu (viz samostatná kapitola) a výsledky jsou zobrazeny na přední straně webové stránky. Metoda TF‑IDF byla vybrána speciálně pro klíčová slova vzhledem k jednoznačnosti výsledků. Při zadávání klíčových slov není nutné brát tolik ohled na širší kontext jako v případě porovnávání článek – ostatní články a uživatel naopak očekává přesnou shodu textu s jeho klíčovými slovy.inverz

* + 1. Doporučení na základě oblíbených kategorií

V nastavení uživatelského profilu si uživatel po registraci a přihlášení může přidat také oblíbené kategorie. Po zvolení se mu zobrazí nejlépe hodnocené články z dané kategorie.

* 1. Doporučení založená na obsahu

Podle oxfordského slovníku je inteligence definována jako: „Schopnost získávání a aplikace znalostí a dovedností.“ [55]. V rámci bádání v oblasti umělé inteligence pak bylo vynaloženo již velké množství úsilí zejména pro sestavení takových metod a vytvoření takových modelů, které dokážou rozpoznat míru podobnosti mezi objekty. V rámci tohoto projektu byly využity metody, algoritmy a modely, které si s tímto problémem do určité míry dokázaly poradit. Některé z těchto metod (jako např. metoda LDA) byly sestaveny nebo se používají i pro jiné případy užití a než je hledání podobnosti v rámci textu, např. pro výzkum v oblasti genetiky.

Český jazyk přináší určitá omezení pro použití znovupoužitelných komponent pro implementaci metod podobností. Mnoho jazyků, zejména *Python*, sice poskytuje knihovny pro zpracování textu, které jsou nezbytné pro lepší výsledky algoritmů podobnosti, většina těchto knihoven však češtinu (a další mezinárodně méně používané jazyky) podporuje jen částečně nebo vůbec. Níže zmíněné přístupy ukazují, jak se s těmito omezeními do jisté míry vypořádat.

V rámci doporučujících systémů a doporučení založených na obsahu lze hovořit o těchto problémech:

* Hledání podobných slov, vět, dokumentů.
* Problém řazení výsledků („Learn-to-rank“ problém).
* Vyhledávání společných témat článků.

Lze si přitom povšimnout, že se jedná o podobnou problémovou doménu jako doménu internetových textových vyhledávačů (*Google*.com, Seznam.cz, Yahoo.com…) a je zde velký průnik mezi problémy řešenými vyhledávači a doporučujícími metodami založenými na obsahu. Pokud se budeme bavit o tomto konkrétním případy užití doporučujících systémů pro doporučování článků, na rozdíl od vyhledávání je k dispozici větší množství textu, tudíž je k dispozici více indicií o tom, jak vyhledávat podobné články. Zatímco pak u internetových vyhledávačů je většinou kladen důraz na co nejpřesnější shodu slov ve vyhledávaném textu, u článků je důležité zaměření i na téma článků, tj. články nemusí být natolik podobné slovy, které obsahují, ale musí být podobné zejména svým hlavním tématem. Další významný rozdíl spočívá v tom, že doporučení založená na obsahu jsou často kombinována s jinými metodami doporučovácích systémů. Tento rozdíl však nemusí být tak velký, protože u moderních vyhledávačů však dochází rovněž ke kombinaci obsahové podobnosti a dalších faktorů založených na uživateli (personalizaci výsledků).

Vedou se debaty o tom nakolik například *Google* svůj textový vyhledávač přizpůsobuje jednotlivým uživatelům (všeobecně známé je např. přizpůsobení na základě polohy uživatele nebo historie minulého vyhledávání). Asi největší rozdíl mezi vyhledávači a doporučujícími metodami (i pokud je omezíme pouze na ty založené na obsahu) je, že zatímco u vyhledávání uživatel ví, co hledá (nebo si to alespoň myslí), u doporučujících systémů by měl naopak systém nabídnout nové, neobjevené položky, které budou pro uživatele zajímavé. Vyhledávače tedy spíše souvisí s oborem získávání informací (obor v angličtině označován jako „information retrieval“), zatímco doporučující systémy spíše souvisí s filtrováním a objevováním informací (někdy označováno jako „information filtering“ nebo „information discovery“).

* + 1. Datové sady

Databáze systému obsahuje velké množství článků (zpráv) jednoho z největších zpravodajských portálů v České republice *Idnes*.cz. Tento zpravodajský portál je třetím nejnavštěvovanějším zpravodajským portálem v České republice. Zde je uveden počet článků v jednotlivých kategoriích v experimentální fázi. Později nicméně přibývají do databáze aktuální články.

|  |  |
| --- | --- |
| **Kategorie** | **Počet článků** |
| Sport | 2013 |
| Domácí | 1080 |
| Ostatní | 926 |
| Zahraničí | 850 |
| Ekonomika | 543 |
| Regionální | 485 |
| Kultura | 419 |
| Hry | 351 |
| Celebrity | 252 |
| Technologie | 220 |
| Auto-moto | 217 |
| Věda | 209 |
| Vztahy | 192 |
| Finance | 180 |
| Móda | 137 |
| Zdraví | 120 |

Tabulka 2: Srovnání kategorií na článcích převzatých z iDNES.cz

Některé modely natrénované na samotných textech z *Idnes*.cz však nenavracely tak dobré výsledky. Důvodem může být, jak menší počet článků vzhledem k *Wikipedii*, tak flexibilita této oblasti, kdy zpravodajské servery vydávají velké množství článků určitých témat, jako jsou aktuální témata (pandemie, později např. válka na Ukrajině) nebo sportovní články, kdy počet článků z této kategorie výrazně převyšuje ostatní kategorie.

Vzhledem k charakteristikám textu bylo k trénování modelů strojového učení (u těch algoritmů, kde se strojové učení využívá) využito také textů z české *Wikipedie* dostupných například ze serveru švédského *Academic Computer Club* z *Umeå University* [50]. Texty *Wikipedie* mají dobré vlastnosti vzhledem k velkému množství článků a rozmanitým tématům. Je zde tedy potenciál, že model natrénovaný na těchto textech bude dobře generalizovat články (viz následující podkapitola 3.3.2).

Zde jsou shrnuty všechny datové sady, které byly v rámci této práce (resp. její experimentální části) použity.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Zdroj** | **Název datové sady** | **Počet dokumentů** | **Střední hodnota délky dokumentu** | **Předzpracování** | **Odstranění nadbytečných slov** |
| *Idnes*.cz | *Idnes* Preprocessed | 8194 | 440.60 | ANO | ANO |
| cs.*Wikipedia* | *Wikipedia* Raw | 457108 |  | NE | NE |
| cs.*Wikipedia* | *Wikipedia* Preprocessed | 457108 |  | ANO | ANO |

Tabulka 3: Přehled využitých datových sad

* + 1. Využívání výstupů z Wikipedie

Běžnou praktikou v rámci *Gensim*, ale i jiných knihoven zabývajících se strojovým učením v rámci přirozeného jazyka (např. knihovna SpaCy), je využít pro trénování výstupy z *Wikipedie* [56] [57] [58].

Důvodů, proč se využívá výstupů zrovna z *Wikipedie* je několik [59] [60]. Je to například:

* Dostupnost textů z *Wikipedie* pod licencí „creative commons“.
* Dostupnost velkého množství textů ve formátu připraveném k použití (není nutné vyvíjet vlastní program pro stahování článků z *Wikipedie*).
* Dostupnost článků v širokém množství různých jazyků.
* Dobrá kvalita článků z gramatického hlediska. Díky revizím zde není příliš překlepů, či gramatických chyb.
* Články mají pravidelnou, jednotnou strukturu.

Další velkou výhodou jsou znalosti uchované ve *Wikipedii*. *Wikipedia* obsahuje velké množství témat. Je to defacto největší encyklopedie na světě [61] obsahující velké množství lidských znalostí – nejspíše se dá považovat i za místo s největší koncentrací lidských znalostí ve zpracovatelné formě na jednom místě. Natrénovaný model na ní tedy má potenciál být dostatečně univerzální a kvalitní pro odvozování nových znalostí. To se ostatně udává jako jedna z nejdůležitějších charakteristik inteligentních systémů, viz např. hypotéza „znalosti jakožto síly“ od Buchanana a Feigenbauma: *„Schopnost inteligentního programu dobře plnit svůj úkol závisí především na množství a kvalitě znalostí, které o tomto úkolu má.“* [62] [63].

Díky organizované struktuře článků z *Wikipedie* každý článek z *Wikipedie* představuje určitý koncept. Jednotlivé články jsou řazeny do kategorií. Empirické výsledky potvrzují, že tato znalostně náročná reprezentace přináší výhody pro kategorizaci textů v rozmanitých souborech dat [64].

Samotné využití *Wikipedie* však pochopitelně automaticky nezajišťuje dobré vlastnosti inteligentních systémů. Jelikož jsou znalosti z *Wikipedie* zapsány textovou formou přirozeného jazyka, je nutné systém přizpůsobit pro porozumění této formy znalostí. Ač pak *Wikipedie* obsahuje široké množství různých znalostí, autoři článků stále předpokládá, že lidský čtenář rozumí určitým obecným znalostem zdravého (selského) rozumu získaným z dřívějších znalostí, kterými ale inteligentní systém nedisponuje [64]. Naučení systému takovýmto obecným znalostem je samotným jádrem bádání v oblasti umělé inteligence a odborníci se tímto problémem zabývají od prvopočátků výzkumů v oblasti umělé inteligence. V roce 1958 byl například vydán článek *Programs with Common Sense*, který je někdy považován za vůbec první akademickou práci zabývající se umělou inteligencí [65]. Užitečné je si povšimnout, že tato práce považuje za (uměle) inteligentní počítačový program právě ten, který nabývá určité „common sense“ (zdravého/selského rozumu).

* + 1. Předzpracování textu

Příprava dat může přímo souviset se schopností nebo neschopností modelu předpovídat [66]. Pro strojovou čitelnost textu, a tedy přesnější výsledky podobnosti článků je vhodné využít techniky pro předzpracování textu. První z často využívaných metod předzpracování textu, která byla využita i pro účely této práce, je metoda odstranění nadbytečných slov (někdy se toto nazývá jako „stop slova“, angl. „stopwords“).

**Odstranění nadbytečných slov**

Tento soubor nadbytečných slov byl jednak shromážděn z volně dostupných internetových zdrojů [67] a také ručně doplněn po analýze nejčastějších slov ve článků, kdy byla odstraněna ta slova z nejpoužívanějších slov ve článcích, která neposkytovala žádné užitečné informace pro další části algoritmů podobnosti (nejčastěji to byly spojky nebo předložky).

**Vyčištění textu**

Po odstranění nadbytečných slov byla odstraněna diakritika (české háčky a čárky i speciální znaky jiných abeced). Dále byly odstraněny číslovky (římské číslice jako „ii“ však byly ponechány vzhledem k tomu, že mohly referovat např. na jména historických postav, popř. názvy uměleckých děl). Dále byly odstraněny některé občas vyskytující se chyby textu jako „zatoulané“ HTML značky. K tomu bylo využito vlastních metod založených na regulárních výrazech a také utility knihovny *Gensim* (viz následující kapitola) „Simple Preprocess“, která na rozdíl od některých jiných nástrojů této knihovny podporuje i češtinu.

**Extrakce klíčových slov**

Dalším důležitým krokem předzpracování textu je použití nástrojů pro odvozování klíčových slov a frází z názvů článků a textů. Klíčová slova mohou upozornit na důležitá témata z textu a poskytnout užitečné informace pro vyhledávání podobných článků. Tato implementace kombinuje několik algoritmů pro extrakci klíčových slov v anglickém jazyce.

Pro extraktor klíčových slov byl využit extraktor klíčových slov *MultiRAKE* (Multilingual Rapid Automatic Keyword Extraction), tj. metoda automatické extrakce klíčových slov založená na kombinaci různých přístupů (*Rake, Topic Rank*, TF-IDF) a *Summa Keywords*, jenž je implementací algoritmu *TextRank* [68] (více v části Řazení výsledků **Chyba! Nenalezen zdroj odkazů.**).

**Stemming: odtrhávání koncovek slov**

Odtrhávání koncovek slov bylo v praktické části práce později nahrazeno lemmatizací. Jedná se však o jednoduchou možnost předzpracování textu, která může vést v některých případech také k dobrým výsledkům.

Odtrhávání koncovek pro český jazyk bylo vyvinuto Lilijanou Dolamic ze švýcarské *Univerzity Neuchâtel* v jazyce Java a o převod kódu pro programovacího jazyku *Python* se postaral Luís Gomés z *Lisabonské univerzity* [69].

**Lemmatizace: Hledání kořenů klíčových slov**

Další důležitou součástí předzpracování textu je lemmatizace. Tato metoda by měla zajistit, aby se několik tvarů téhož slova, například „lepší“ a „dobrý“ nebo „ostravský“, „ostravští“, „ostravských“, spojilo do jednoho slova (např. „ostrava“ nebo „ostravský“). To je důležité zejména pro češtinu, která má obrovské množství slovních tvarů. V češtině má stejné slovo několik různých tvarů na základě sedmi gramatických pádů jednotného a množného čísla a v tomto je čeština komplexnější než např. jazyk Anglický (ale i spousta dalších ve světě více používaných jazyků). Čeština je tedy označována za morfologicky bohatý jazyk, jenž svou morfologickou složitostí přesahuje jazyk anglický [70]. Zatímco některé z těchto gramatických pádů nemění syntax slova, není neobvyklé, že totéž slovo má devět různých tvarů (například slovo „stát“ má tvary: stát, státu, státe, státě, státě, státy, států, státům, státech, státech).

Vzhledem k tomu, že implementovat vlastní řešení by vydalo i na několik samostatných diplomových prací, k řešení tohoto problému byl použit morfologický analyzátor *Majka* [71] vyvinut v brněnské *Masarykově univerzitě* obsahující morfologické databáze k českému jazyku (a dalších 14 jazyků).

**Bigramy a tokenizace**

Pokud to daná metoda vyžaduje (jako např. LDA), texty jsou rozděleny na jednotlivá slova. Aby se v takovém případě zabránilo ztrátě slovních spojení, je možné vytvořit dvojice nebo trojice slov („bigramy“ a „trigramy“), které spojí často se vyskytující slovní spojení v rámci korpusu dokumentů do jednoho řetězce separovaného oddělovačem, nejčastěji znakem podtržítka („\_“). Nedojde tak ke ztrátě některých podstatných informací. Na základě výskytů u sebe často vyskytujících se slov je tedy možné sestavit i slovní spojení jako „Václav II.“ nebo „dopravní přestupek“, tzn. reálně bude po předzpracování textu a převodu na bigram vypadat výše uvedené spojení jako „vaclav\_ii“ a „dopravní\_prestupek“. Protože je „Václav II.“ podstatně rozdílná informace než „Václav III.“, či dokonce „Václav Klaus“, dojde v tomto případě k přímému rozlišení historických postav.

Vzhledem k tomu, že by bylo za normálních okolností kvůli předzpracování textu číslovka „II“ odstraněna jako nežádoucí znak, je proto vhodné provést vytvoření bigramů nebo trigramů na textech s ne tak striktním předzpracováním, či vytvořit bigramy nebo trigramy dříve a poté vynechat slova s oddělovačem od předzpracování.

**Shrnutí části předzpracování textu**

Text, který byl vstupem do modelů založených na článcích z *Idnes*.cz zmíněných v následujících kapitolách je řetězec spojen z částí:

* Název kategorie.
* Titulek.
* Perex článku.
* Klíčová slova.

Části titulku a perexu a v případě *Wikipedie* celé texty, jsou poté předzpracovány dle těch technik již zmíněných výše:

1. Všechna písmena jsou převedena na malá.
2. Odstranění HTML značek.
3. Odstranění odkazů (URL).
4. Odstranění teček, čárek, středníků, podtržítek, dvojteček apod. Kompletní výpis odstraněných znaků je:

!"#$%&'()\*+, -./:;<=>?@[\]^\_`{|}~

1. Odstranění numerických znaků.
2. Rozdělení řetězce na tokeny (které by měly být slovy).
3. Dodatečné odstranění pro ujištění, že zde stále nepřebývá něco nežádoucího:
   1. Odstranění tokenů, které nejsou slovy, ale jsou tvořeny pouze znaménkem nebo více znaménky.
   2. Odstranění jednoznakových slov.
   3. Odstranění slov tvořených pouze číslovkami.
4. Lemmatizace podle českého morfologického slovníku *Majka*.
5. Odstranění nadbytečných slov (českých a obecných).
6. Spojení tokenů (tentokrát by již měly být tvořeny pouze slovy) do řetězce odděleného mezerou.
7. Odstranění diakritiky. (S tímto krokem bylo různě experimentováno; někdy bylo použito, někdy byla diakritika zachována.)
8. Vytvoření bigramů nebo trigramů
9. Tokenizace

Důležitost předzpracování je například vidět na tomto příkladu sportovního fotbalového článku. Konkrétní skóre a statistiky zápasů nám pro náš případ užití nedávají prakticky žádnou užitečnou informaci o obsahu článku, stejně jako např. znaky využívané pro zápis sestavy a statistik utkání. Pokud tyto znaky odstraníme, zůstanou nám naopak užitečné informace o názvech týmů a jména fotbalistů:

'Náhradníci:\r\nVítek,', 'Marek', '–', 'Stáňa,', 'Chwaszcz,', 'Ulbrich,', 'Burda,', 'Zúbek.', 'Náhradníci:\r\nVaňák', '–', 'Pajkrt,', 'Lauko,', 'Šimon,', 'Velich,', 'Kocourek,', 'M.', 'Kubista.', 'Žluté', 'karty:\r\n20.', 'Otrísal,', '39.', 'Jeřábek,', '58.', 'Silný,', '64.', 'Rolinek,', '71.', 'Valachovič,', '90.', 'Veselý', 'Žluté', 'karty:\r\n18.', 'Rudnytskyy,', '84.', 'Šimon', 'Červené', 'karty:\r\n69.']

Po předzpracování dostaneme:

'nahradnik', 'vitek', 'marka', 'stana', 'chwaszcz', 'ulbrich', 'burda', 'zubek', 'nahradnik', 'vanak', 'pajkrt', 'lauko', 'simon', 'velich', 'kocourek', 'kubista', 'zluty', 'karta', 'otrisal', 'jerabek', 'silny', 'rolinek', 'valachovic', 'vesely', 'zluty', 'karta', 'rudnytskyy', 'simon', 'cervene', 'karta']

* + 1. Podobnost slov a dokumentů

V rámci nauky o přirozeném jazyku jsou odvozovány tři základní míry podobnosti: lexikální podobnost, sémantická podobnost a sémantická příbuznost [72].

Lexikální podobnost je lingvistický pojem zabývající se podobností slov různých jazyků (zanedbává se přitom syntaktická podobnosti a další aspekty [73].

Důležitějším pojmem v rámci umělé inteligence a této práce je pojem sémantické podobnosti. Jedná se o matematickou reprezentaci nalezenou na množině termínů nebo dokumentů na základě významu slov pro čtenáře, resp. sémantiky [74].

Zatímco sémantická podobnost zahrnuje pouze vztahy mezi slovy v poměru „je…“ („is a…“, „ISA“) [75], např. „Vítkovice“ a „hokejový klub“, zanedbává ostatní vztahy – jako např., že může být „Vítkovice“ také příbuzné s „Baník“ (ač se jedná o klub fotbalový) nebo například se slovem „atletika“ (je to také sport, a navíc je závod Zlatá tretra pořádán na městském stadionu v ostravských Vítkovicích). Právě ostatními druhy vztahů mezi slovy zahrnuje sémantická příbuznost.

V rámci informatiky existuje několik způsobů, jak přistupovat k této problematice. Jeden ze základních přístupů je využití ontologií, pomocí kterých lze reprezentovat vlastnosti, kategorie nebo vztahy mezi koncepty a entitami. Pro ontologie lze následně využít teorie grafů a hledat například nejkratší vzdálenost mezi jednotlivými uzly reprezentující pojmy nebo koncepty. Odtud ostatně dostaly název *sociální sítě* dnes tolik založené na doporučovacích systémech (běžný člověk si představí pod pojmem graf spíše ).

V rámci statistického přístupu se poté využívá často model vektorového prostoru („vector space model“) představený v roce 1975 Saltonem, Wongem a Yangem [76].

V modelu vektorového prostoru máme množinu dokumentů *D* a dotazovaný (hledaný) dokument *Q* (dokument ke kterému chceme nalézt podobné dokumenty)*.* Těm pak odpovídají vektory dokumentů *dj* a vektor dotazovaného dokumentu *q*. Prvky těchto vektorů jsou váhy *w1,j* pro vektory dokumentů a *w1,q* pro vektor dotazovaného dokumentu. Pro výpočet těchto vah je poté možné využít různé metody. V rámci této práce byla například využita metoda TF‑IDF.

Na tyto vektory lze poté aplikovat různé metody měření podobnosti. V rámci zpracování přirozeného jazyka se často využívá kosinová podobnost, a to především pro svou schopnost vyjadřovat relativní srovnání rozměrů vektorů oproti absolutnímu, kdy je tato míra vhodnější v případě, kdy je naší prioritou četnost hodnot oproti absolutním hodnotám – tedy například právě v případě četností termínů v dokumentech [77].

Výpočet kosinové podobnosti pro sadu dokumentů lze zapsat jako:

Alternativou pro výpočet podobnosti může být např. Euklidovské vzdálenost, Shannon-Jensenova vzdálenost).

Variantou „klasické“ kosinové podobnosti je jemná kosinová míra („Soft Cosine Measure“, SCM) [78] doporučována v oficinálních příkladech knihovny *Gensim* [79] pro výpočet podobnosti dokumentů pomocí Word2Vec (viz v podkapitole 3.3.5). Metoda SCM umožňuje posoudit podobnost dvou dokumentů, i v případě, že dokumenty nemají žádná společná slova [80]. Na rozdíl od tradiční kosinové podobnosti vypočítává i podobnost dvojic slov, která je pak započítána do výpočtu kosinové podobnosti mezi dvěma dokumenty. Je tedy možné odhalit podobnost dvou dokumentů skládajících se ze synonym („ahoj“ a „čau“) nebo souvisejících slov jako „prezident“ a „premiér“ nebo „hokej“ a „puk“.

Například ve výzkumu Charleta a Damnatiho pomocí jejich systému *SysBow* tato metoda vykazovala lepší výsledky než jiné přístupy [80].

Aplikace modelu vektorového prostoru a příklad výpočtu kosinové podobnosti společně s metodou TF‑IDF je představen v podkapitole 3.3.3.

* + 1. Knihovna Scikit-Learn

*Scikit-Learn* je knihovna poskytující podporu pro metody datové analýzy v programovacím jazyku *Python*. Knihovna využívá v implementaci například matematické knihovny *Numpy*, *SciPy* nebo *Matplotlib* [81]. V tomto projektu byly nástroje knihovny *Scikit-Learn* využity zejména při implementaci metody TF-IDF, ale take pro mnoho dalších dílčích úloh jako například pro výpočet Kosinovy podobnosti.

* + 1. Knihovna Gensim

Pro značnou část implementace podobnosti založené na obsahu bylo využito nástrojů a modelů knihovny (resp. projektu) *Gensim* [82] [83]. *Gensim* je knihovna nástrojů a modelů pro zpracování přirozeného jazyka vyvíjená skupinou RARE Technologies pod vedením Radka Řehůřka. Pro část předzpracování textu plně podporuje jazyk anglický a češtinu podporují jen některé jeho části. Jazyk nehraje takovou roli v pozdější fázi trénování modelů, hraje však velkou roli v části předzpracování textu. S vhodnými vlastními zásahy do části předzpracování a s použitím dalších knihoven lze však *Gensim* využít prakticky pro jakýkoliv jazyk.

*Gensim* pro svou implementaci využívá například optimalizovanou matematicko-vědeckou knihovnu *Numpy* využívající předkompilovaný kód jazyka C pro co nejrychlejší matematické operace.

Tato knihovna byla využívána pro mnoho z činností v této práci kvůli své sadě nástrojů, které poskytuje (kromě modelů strojového učení a algoritmů i některé metody pro usnadnění zpracování textu) a kvůli výhodě v možnosti používat model, aniž by bylo nutné nahrávat všechny dokumenty naráz do paměti [84]. Pro používání *Gensim* je nutné stále potřebné poměrně velké množství paměti, pokud se pracuje s velkými datovými sadami, ale možnost využití vhodných programovacích technik jako návrhových vzorů iterátor nebo generátor, využití paměti značně snižuje.

Více o LDA a využívání paměti v části 3.3.10.

* + 1. Doporučení na základě četnosti a inverzní četnosti termínů

Vzhledem k tomu, že metoda TF‑IDF nevyžaduje trénování modelů strojového učení, její použití je z uvedených metod nejvíce přímočaré. Hlavním úkolem TF‑IDF je nalézt číselné vyjádření vyjadřující, jak je slovo důležité v rámci dokumentu pocházejícího z nějaké množiny (korpusu) dokumentů [85]. Jedná se o metodu, která se velice často používá u doporučujících systémů (např. asi 83 % doporučujících systémů digitálních knihoven v roce 2015 nějakým způsobem metodu TF-IDF využívalo). TF-IDF se rovněž často využívá v rámci internetových vyhledávačů [86]. TF-IDF je ale možné využít i v rámci dalších aplikací, jako například pro automatickou identifikaci nadbytečných slov [87].

Část četnosti termínů („term frequency“) má za úkol nalézt počet výskytů termínů v dokumentu. Protože ale dokumenty bývají různé délky, využívá se vážení termínů představené statistikem Hansem Peterem Luhnem již v roce 1957 [88].

Pro hledání důležitých slov v rámci vět a dokumentů a následně podobných vět ale tento přístup nestačí. Podle českého národního korpusu jsou nejčastěji vyskytujícími slovy v češtině slova: „být”, „a”, „se”, „v”, „na”, „ten”, „on”, „že”, „s“, „z“ atd. Níže v tabulce je vidět srovnání národního korpusu s datovými množinami využitými v této práci - tj. v asi 8tisícové množině článků z *Idnes*.cz a v rámci asi 457 000 množině článků z české *Wikipedie*:

|  |  |  |
| --- | --- | --- |
| **Národní korpus** | **iDnes.cz** | **Wikipedia.cz** |
| být | a | na |
| a | v | se |
| se | se | kategorie |
| v | na | je |
| na | je | do |
| ten | že | ve |
| on | to | byl |
| že | s | roce |
| s | i | roku |
| z | z | jako |

Tabulka 4: Srovnání nejčastějších českých slov (dle národního korpusu) a nejčastějších slov datových sad využitých v této práci

Je vidět, že nejčastější slova jsou spojky a předložky, které jsou v přirozeném jazyku a jeho syntaxi velice důležité. Jelikož se však nachází ve většině všech vět různých významů a témat, samostatně neříkají nic o významu slova ve větě, resp. o významu v rámci celého dokumentu. Už jen z tohoto jednoduchého pokusu tedy lze vidět, že nejčastější slova nejsou dobrým měřítkem pro určení relevantních slov v rámci vět a následně podobných vět. Druhá část TF‑IDF, tj. inverzní četnost dokumentů (idf, „inverse document frequency“) tedy slouží jako jakési vybalancování tohoto fenoménu přirozených jazyků, kdy jsou nejčastější slova ve větách zanedbána ve prospěch určitého kompromisu ve formě unikátních slov v rámci dokumentů. Inverzní faktor četnosti dokumentů snižuje váhu až příliš častých termínů ve prospěch méně častých. Tato slova jsou většinou jiných slovních druhů – podstatná jména, popř. zájmena nebo slovesa.

Přesný statistický popis slov ve větách a dokumentech představila v roce 1972 informatička Karen Spärck Jonesová: „*Úplnost popisu dokumentu je počet termínů, které obsahuje, a specifičnost termínu je počet dokumentů, kterých se týká.*“ [89]

V rámci tohoto projektu bylo zvoleno explicitní odstranění některých nadbytečných slov pomocí sestaveného seznamu takovýchto slov. Většina z výše uvedených nejčastějších slov češtiny se tedy odfiltruje z vět ještě před samotným počítáním TF‑IDF. Některá slova však nemusí být v tomto seznamu zahrnuta a je předpokládáno, že i při explicitním určení nadbytečných slov, část inverzní četnosti termínů TF‑IDF stále nabízí podstatnou část určení významu slov ve větách a dokumentech. Ač tedy některé práce využívají k výpočtům pouze část četnosti termínů [90], při výpočtech v této práci byla část „idf“ ponechána. Explicitní určení nadbytečných slov v rámci implementace TF‑IDF umožňuje také knihovna *Scikit‑Learn* (viz níže). Jedná se tedy o legitimní přístup. Stálo by však za posouzení například v rámci jiné akademické práce, jaký má část „idf“ vliv v případě explicitní definice zbytečných slov, která by měla příliš častá významově prázdná slova již značně odfiltrovat a zda pak dochází k horším výsledkům než v případě kompletního TF‑IDF ve variantě s odfiltrováním klíčových slov před využitím metody a pouhým použitím TF‑IDF.

U metody TF-IDF pracujeme s termíny *t* dokumenty *d* v rámci korpusu dokumentů *D.* Část *TF* poté můžeme neformálně vyjádřit jako:

Odstranění příliš častých, ale neužitečných slov, můžeme neformálně definovat s pomocí celkového počtu dokumentů v korpusu *N* jako:

A nakonec jednoduše vynásobíme tyto části:

Použití logaritmu v tomto vzorci nemusí být na první pohled intituivní čtenáři, který běžně nepracuje s matematikou, avšak snadno si význam logaritmu lze představit ve zjednodušeném prípadě, kdy bychom měli nějaké slovo obsažení v úplně všech dokumentech. V takovém případě máme samozřejmě ve zlomku části *IDF* 1 a tím pádem dostáváme *log(1)*, což je 0. Ať je tedy část *TF* jakákoliv a část *TF* „křičí“ o důležitosti tohoto slova sebevíce, část *IDF* je 0 a násobení tedy dá výsledek také 0. Typicky si to lze představit např. u velmi častých slov jako „a“. Naopak méně časté termíny na jednu stranu část *TF* příliš nevizdvihuje, na druhou stranu od *IDF* neobdrží takovou „penalizaci“. Např. bychom zjistili, že TF je pouze 0,01 a poměr v části IDF 300/100 (300 dokumentů a 100 s výskytem, např. slova „sport“). Takový výsledek sice bude malý, ale určitě nebude 0 (bude to zhruba 0,0047). Pokud by ale těch výskytů bylo zase více, např. něco kolem 200, opět by konečný výsledek klesl. Logaritmus zde funguje jako jakási normalizace. Logaritmická funkce má vzhledem ke svému průběhu tendenci brát velký ohled na rozdíly v malých číslech (např. 1 vs 3), ale rozdíly ve velkých číslech již tolik významné nejsou.

Pro realizaci TF‑IDF bylo využito knihoven *Scikit-Learn* a *Numpy*, jenž poskytují vhodnou sadu nástrojů pro převod textu do maticové podoby a následné maticové operace.

Jako alternativní přístup k TF-IDF lze k vektorizaci textu použít metodu hashování. Ta změní kolekci textových dokumentů na řídkou matici obsahující počty výskytů tokenů (nebo binární informace o výskytu), případně normalizované jako frekvence tokenů nebo promítnuté na euklidovskou jednotkovou sféru [94]. Všimněte si, že na rozdíl od TF-IDF zde není zmíněna penalizace za příliš častý výskyt dokumentů v rámci celého korpusu dokumentů, nicméně metoda hashování má efektivní výkon a je paměťově nenáročná. Vzhledem k tomu, že mnoho nepotřebných slov je z textu odstraněno již ve fázi předzpracování, předpokládá se, že je bezpečné tuto metodu použít, aniž by došlo ke ztrátě kvality modelu. V počátečních experimentálních fázích však metoda TF-IDF vykazovala slibnější výsledky, proto jsme pro představitele "odlehčené metody" zvolili spíše metodu TF-IDF.

* + 1. Doporučení na základě podobnosti vektorů slov

Word2Vec je metoda zpracování přirozeného jazyka, jejíž cílem je nalézt optimální reprezentaci slov v podobě vektorů na základě podobnosti mezi vektory. Podobnost mezi vektory má poté odpovídat sémantické podobnosti vstupních slov (více o podobnosti slov v podkapitole 3.3.4).

Pod metodou Word2Vec se skrývají metody strojového učení implementované různými knihovnami.

Metoda Word2Vec byla vyvinuta ve společnosti *Google* týmem vedeným českým informatikem specializujícím se na obor umělé inteligence Tomášem Mikolovem, který představil Word2Vec v rámci článků *Efficient Estimation of Word Representations in Vector Space* („Efektivní odhad reprezentace slov ve vektorovém prostoru“) a *Distributed Representations of Words and Phrases and their Compositionality* („Distribuovaná reprezentace slov a frází a jejich součástí“) [91] [92]. Tým vedený Mikolovem se snažil poukázat na větší přesnost výsledků při použití jejich přístupu Word2Vec oproti tehdejším dosavadním technikám. Do té doby bylo pro stejné případě použití jedním z oblíbených přístupů například využití metody latentní sémantické analýzy, resp. latentního sémantického indexování (LSA/LSI) zaregistrovaná jako patent již v roce 1988 [93].

Word2Vec funguje ve dvou základních režimech (na 2 různých modelech). Prvním je model využívající techniku Cotinuous-Bag-Of-Words (CBOW). Tato reprezentace transformuje slova z vět do celočíselných vektorů, kde prvky vektoru představují počet výskytů slova v dokumentu. CBOW se snaží předpovědět jedno slovo na základě sousedních slov a na základě velikosti okna, které určuje počet sousedních slov. Druhá varianta modelu Word2Vec využívá techniku Skip-Gram, jejíž účelem je naopak předpovědět určitý počet sousedních slov na základě nějakého jednoho daného slova. Takto lze předvídat kontextová slova cílového slova.

I pro lidského experta je nejspíše jednodušší určit jedno slovo z kontextu ostatních sousedních, stejně tak je to většinou u modelů strojového učení, kdy je předpovídání v rámci CBOW jednodušší než v rámci Skip-Gram a tomu odpovídá i délka trénování: zatímco v původním článku o Word2Vec týmu Mikolova je zmíněna v případě CBOW délka trénování několik hodin, pro Skip-Gram to bylo více než 3 dny [91].

Obrázek 32: Obrázek znázorňující rozdíl mezi přístupem CBOW (vlevo) a Skip-Gram (vpravo) [95] [96]

Alternativou je také využití metody *CountVectorizer*, která si uchovává přehled o celém slovníku všech dokumentů. Proto je paměťově náročnější. Na *CountVectorizer* na druhou stranu může být aplikována metoda inverzní transformace a tím pádem je možné lépe prozkoumat model – např. jaké části dat jsou pro model nejdůležitější [97].

Formát *FastText* byl vyvinut společností Facebook v rámci jejich výzkumného týmu *Facebook AI Research Lab* s motivací vyvinout rychlejší i paměťově méně náročnější formát pro reprezentaci a klasifikaci textu. Jejich postup byl shrnut ve článku *Bag of Tricks for Efficient Text Classification* [98]. Na tomto výzkumu se kromě jiných (Joulin, Grave, Bojanowski) rovněž podílel vývojář Word2Vec Tomáš Mikolov. Tým dosáhl optimalizace pomocí několika metod, například pomocí využití hierarchického (stromového) rozdělení kategorií nebo Huffmanova algoritmu pro balancování stromu, či pomocí využívání vektorů nižších dimenzí [99]. V rámci vývoje bylo dbáno také na to, aby fungoval dobře i pro morfologicky bohaté jazyky jako je právě český jazyk [99].

Pro účely této práce bylo důležité především to, že *FastText* nabízí k využití předem trénované modely sestavené nad články z *Wikipedie* [100], pro Word2Vec tedy nebylo nutné trénovat a optimalizovat vlastní model. Dle přísloví „důvěřuj, ale prověřuj“ však byla provedena rychlá metoda pro ujištění se, že Word2Vec funguje lépe než vektorizace pomocí metody hashování.

Srovnání níže ukazuje nejpodobnější slova pro konkrétní slovo “hokej”. U korpusu *Wikipedie* je například vidět, proč některé lekce z této oblasti často opakují mezinárodní lidové moudro “méně je někdy více”. Při použití všech slov z *Wikipedie* se totiž vyskytují jako nejpodobnější slova vzniklá chybami v textech nebo se jedná o texty názvu obrázků jako například „hokej-“ Nebo „hokej0“. Po nastavení prahu pro počet vektorů slov (u knihovny *Gensim* je to parametr *limit*) na 100 000 tato slova vymizela a zůstala pouze slova relevantní. Přišli jsme ale přitom i o některá relevantní slova jako “nároďák”, “hokejisti” nebo “zimák”. Jsou to sice nespisovné tvary slov, avšak pokud nechceme přijít o slovní zásobu modelu, bylo by vhodné zvolit jako kompromis 350 000 vektorů slov.

U modelu *Idnes*.cz je celkově méně slov a dokumentů, takže byl parametr nastaven na 150 000 slov:

|  |  |
| --- | --- |
| **limit=150000** | |
| Gernát | Sim |
| Špaček | 0.759 |
| Vsetínský | 0.694 |
| Straka | 0.686 |
| Will | 0.657 |
| Kapitán | 0.632 |
| Angažmá | 0.615 |

Tabulka 5: Ukázka nalezených podobných slov ke slovu hokej. Obsahuje jména hokejistů nebo hokejových týmů, či výrazů souvisejících s tímto sportem

Níže jsou představeny výsledky 10 nejrelevantnějších článků podle modelů:

* *Idnes*.cz model (vlastní)
* *FastText* (cs.*Wikipedia*)
* *FastText* limitovaný (cs.*Wikipedia*)

Kroky sestavení modelu:

1. Předzpracování textu (Pozn.: V případě doporučení “na vyžádání” s přepočítáváním podobností je již provedeno z důvodu časové náročnosti předem).
2. Nahrání všech článků z databáze do datového rámce příspěvků.
3. Nahrání kategorií článků z databáze do datového rámce kategorií.
4. Spojení datového rámce příspěvků a kategorií.
5. Smazání objektů příspěvků a kategorií pro uvolnění paměti.
6. Spojení kategorie, nadpisu, výtažku článku a kategorií do jednoho řetězce.

*Nebo se přeskočí kroky 1. až 6. a využije se předem předzpracovaného i spojeného textu.*

1. Pro jednoduchost zpracování je do textu tentokrát přidána i koncovka URL oddělená znakem středníku od zbytku textu.
2. Načtení varianty natrénovaného Word2Vec modelu (založen na článcích z *Idnes*.cz nebo *FastText* z *Wikipedie* nebo limitovaný *FastText*).
3. Inicializace indexu podobnosti termínů WordEmbeddingSimilarityIndex, který vypočítá kosinovou podobnost mezi slovy.
4. Načtení slovníku vytvořeného ze slov z *Idnes*.cz (předpokládáno, že je již vytvořený).
5. Načtení korpusu skládajícího se z článků z *Idnes*.cz (předpokládáno, že je již vytvořený).
6. Výpočet řídké matice podobnosti termínů. Vstupem je index podobnosti termínů a slovník.
7. Vytvoření indexu podobností jemné kosinové podobnosti dokumentů (dále jen doc\_sim\_index) na základě korpusu a matice podobnosti.
   1. Seřazení výsledků, výběr 21 největších podobností.
8. Rozdělení textového řetězce na jednotlivá slova.
9. Vytvoření *bag-of-words* struktury: dvojice (id tokenu, počet slov).
10. Extrakce podobností dotazovaného textu na základě „doc\_sim\_index“. Výstupem je dvojice (id dokumentu, kosinová podobnost).
11. Navrácení původního textu z množiny dokumentů dle identifikátoru.
12. Extrakce koncovky URL.
13. Vložení koncovky URL a skóre podobnosti do slovníku výsledků.
14. Převod *Python* slovníku\* výsledků na JSON.

Pozn: \*Myšleno tentokrát jako datový typ *Python*u, nikoliv slovník v rámci zpracování přirozeného jazyka jako v kroku 10 nebo 12.

Před prvním použitím a při případné aktualizaci modelu je nutné vykonat ještě tyto kroky, jestliže jako v našem případě nemáme data ve formátu *FastText*.

1. Načíst předem natrénovaný model formátu „Facebook vektorů“, standardně uložený v binárním zkomprimovaném souboru *\*.bin.gz.*
2. Doplnění norem vektorů (viz níže).
3. Převod na model formátu *FastText*.

Pokud by se uložil *FastText* do standardního Word2Vec formátu rovnou, v pozdějších částech programu by při výpočtech hrozily chyby. Ještě před uložením do Word2Vec formátu je tedy pro kompatibilitu s knihovnou *Gensim použita* metoda *Gensim* fill\_norms(). Tato metoda pomocí knihovny *Numpy* a její metody *linalg.norm()* zajišťuje, že každý budoucí kód, který bude později modifikovat vektory, zajistí, aby byly buď přepočítány normy vektorů, nebo aby byla nastavena hodnota „None“, která později na vyžádání spustí úplný přepočet norem [101].

Protože knihovna *Gensim* podporuje i kompatibilitu mezi formáty *FastText* a Word2Vec, *FastText* formát byl uložen pro další použití již ve formátu Word2Vec pomocí metody save\_word2vec\_format(), která zredukuje velikost a urychlí načítání souboru uložených vektorů.

* + 1. Doporučení na základě podobnosti vektorů slov a dokumentů

Aby byl brán při výpočtech ohled nejen na jednotlivá slova, byla metoda Word2Vec rozšířena o vektorovou reprezentaci celého dokumentu v rámci práce Leeho a Mikolova: *Distributed Representations of Sentences and Documents*. V rámci knihovny *Gensim* je pojmenována tato metoda jako Doc2Vec [102].

Jedná se opět o způsob reprezentace slov pomocí vektorové reprezentace v rámci tzv. spojitého pytle slov (continuous bag-of-words, CBOW) a Skip-Gram [103]. Tyto myšlenky se využívají k předpovídání podobných významů slov u metody Word2Vec (viz podkapitola 0). V případě Word2Vec představují vektory blízko sebe podobné významy slov na základě kontextu slov ve větách i jinak. Některé z následujících částí kapitoly byly rovněž využity v článku *An approach for recommending relevant articles in news portal based on Doc2Vec* jejímž spoluautorem byl pan RNDr. Bogdan Walek Ph.D.

Jako jakýsi prostřední myšlenkový krok mezi metodami Word2Vec a Doc2Vec je možné vypočítat vektorovou reprezentaci pomocí Word2Vec a následně průměry mezi prvky vektoru. Nad těmito průměry prvků vektorů lze poté použít výpočty podobnosti, jako je kosinová podobnost nebo euklidovská vzdálenost, k nalezení nejpodobnějších vektorů na základě úhlu nebo vzdálenosti, tj. jimi reprezentovaných podobných článků.

V reálných scénářích Word2Vec i Doc2Vec se však k nalezení optimálních vektorů používají modely neuronových sítí. Doc2Vec přidává ke slovním vektorům Word2Vec další vektor, který reprezentuje celý dokument a jeho kontext, aby bylo možné předpovědět slova v dokumentu. Tyto vektory dokumentu jsou kombinovány s vektory slov, aby poskytly lepší výsledek.

Pro implementaci metody Doc2Vec byla opět využita open-source knihovna *Gensim* [82] a pro předzpracování další knihovny a vlastní metody.

Níže jsou uvedeny základní kroky implementace algoritmu. Při nasazení v reálném scénáři jsou tyto kroky odděleny od samotného porovnávání podobnosti, poté uloženy ve vhodném typu souboru, komprimovány a načteny až během porovnávání podobnosti vzhledem k časové náročnosti některých níže popsaných kroků.

Obecně je při znovupoužití tohoto řešení nutné dbát na využití vhodných optimalizačních technik pro konkrétní programovací jazyk použitý pro implementaci algoritmu.

V této implementaci byla použita kombinace klíčových slov, klíčových slov článku, kategorie, názvu a krátkého popisu článku. Tento vykazoval subjektivně lepší výsledky v této kombinaci v našem souboru dat ve srovnání s použitím pouze klíčových slov jako vstupního textu pro metodu Doc2Vec. Pro jiné případy použití nebo datové sady však mohou být užitečné i jiné kombinace vstupů.

Algoritmus pro výpočet nejrelevantnějších (podobných) článků se skládá z následujících kroků:

1. Načtení všech článků.
2. Předběžné zpracování – obsahuje odstranění stop slov a lemmatizaci.
3. Načtení hlavního článku – v tomto kroku je načten hlavní článek. Na základě obsahu jeho článku se vypočítají relevantní články.
4. Extrakce kategorie z hlavního článku.
5. Sloučení klíčových slov článku, kategorie, názvu a perexu článku (krátkého popisu) do jednoho textového řetězce nazvaného MainArticleText – textový řetězec MainArticleText bude dále použit v algoritmu.
6. Vytvoření a trénování modelu Doc2Vec – vytvoření a trénování modelu Doc2Vec se skládá z těchto dílčích kroků:
   1. Načtení všech článků z databáze do modelu Doc2Vec.
   2. Pro každý článek v databázi – sloučení klíčových slov článku, kategorie, názvu a hlavního odstavce článku (krátkého popisu) do jednoho textového řetězce nazvaného ArticleText (text článku).
   3. Tokenizace slov MainArticleText a ArticleText (každého článku v databázi).
   4. Označování dokumentů – každý článek je označen jedinečným identifikátorem.
   5. Nastavení parametrů délky vektoru, nastavení počtu epoch a parametru alfa.
   6. Inicializace modelu Doc2Vec.
   7. Trénování na označeném dokumentu.
   8. Snížení tréninkového parametru alfa. Rychlost učení se mění v čase. Potřebujeme větší změny vah modelu Doc2Vec na začátku procesu učení, abychom dostatečně ovlivnili změny v modelu. A menší změny v době, kdy by model měl mít již určité znalosti o článcích.
   9. Uložení natrénovaného modelu Doc2Vec.  
      Zpracování dat MainArticleText – zpracování dat se skládá z těchto dílčích kroků:
   10. Rozdělení klíčových slov MainArticleText (část MainArticleText, která obsahuje klíčová slova článků) do samostatných frází ve výstupním seznamu.
   11. Rozdělení zbytku MainArticleText (název kategorie, nadpis, perex článku) na jednotlivá slova ve výstupním seznamu.
   12. Spojení výstupního seznamu klíčových slov a zbytku seznamu MainArticleText do výstupního seznamu MainArticleTextList*.*
   13. Převod seznamu MainArticleTextList na vektory.
7. Výpočet podobnosti seznamu MainArticleTextList s ostatními články – obsahuje tyto dílčí kroky:
   1. Normalizace: Přiřazení váhy každému prvku vektoru (číslo z intervalu <1,0 a -1,0>).
   2. Výpočet váženého průměru za všechny prvky vektoru.
   3. Výpočet kosinové podobnosti (vzdálenosti).
   4. Seřazení výsledků na základě podobnosti.
8. Zobrazení setříděného seznamu nejpodobnějších článků

Některé části navrhovaného algoritmu budou demonstrovány v následujících částech.

V tabulce I. je uveden zjednodušený příklad výpočtu vah prvků vektoru MainArticleTextList. Toto slouží jako mezikrok mezi Word2Vec a Doc2Vec. Jednotlivé hodnoty jsou odvozeny z podobnosti slov pomocí modelu Word2Vec v programu *Gensim*. Aritmetický průměr těchto hodnot podobnosti představuje celý dokument. Vysokých hodnot podobnosti se dosahuje při porovnávání slov jako „zemřít“ a „narodit se“ nebo „Kopřivnice“ a „Krkonoše“ (oba zeměpisné názvy, české město a české pohoří). Nízké hodnoty jsou zjištěny u dvojic s nesouvisejícími významy, jako je „představit“ a „Krkonoše“.

|  | **Zemřít** | **Poslední** | **Nosič** | **Krkonoše** | **Helmut** |
| --- | --- | --- | --- | --- | --- |
| Autor | 0.15 | 0.20 | 0.22 | 0.08 | 0.19 |
| Legendární | 0.19 | 0.34 | 0.17 | 0.15 | 0.13 |
| Narozen | 0.55 | 0.11 | 0.13 | 0.12 | -0.002 |
| Rok | 0.24 | 0.32 | 0.13 | 0.08 | 0.17 |
| **Average values of Doc 1** | 0.28 | 0.24 | 0.16 | 0.11 | 0.12 |
| Zemřít | 1 | 0.12 | 0.14 | 0.14 | 0.09 |
| Tvůre | 0.17 | 0.12 | 0.16 | 0.11 | 0.11 |
| Film | 0.29 | 0.20 | 0.26 | 0.15 | 0.13 |
| Národní | 0.05 | 0.27 | 0.11 | 0.21 | 0.10 |
| **Average values of Doc 2** | 0.38 | 0.18 | 0.17 | 0.15 | 0.11 |
| Kopřivnice | 0.10 | 0.05 | 0.17 | 0.41 | 0.21 |
| Limuzína | 0.12 | 0.16 | 0.17 | 0.20 | 0.19 |
| Tatra | 0.09 | 0.16 | 0.18 | 0.40 | 0.21 |
| Představit | 0.29 | 0.17 | 0.08 | 0.02 | 0.07 |
| Veřejnost | 0.14 | 0.19 | 0.09 | 0.19 | 0.08 |
| **Average values of Doc 3** | 0.32 | 0.10 | 0.18 | 0.19 | 0.14 |

Tabulka 6: Zjednodušený příklad principu Doc2Vec jakožto myšlenkový mezikrok mezi Word2Vec a Doc2Vec

Nyní můžeme průměry Word2Vec z porovnání podobnosti slov označit jako vektory reprezentující konkrétní dokument v porovnání s ostatními dokumenty. Po přidání dalších slov v původním článku (např. „ikona“, „Úpa“) lze výsledný vektor identifikovat jako: doc\_1 = (0,28 0,24 0,16 0,11 0,12 0,20 0,02). Při provedení kosinové podobnosti mezi těmito vektory můžeme najít vyšší podobnosti mezi subjektivně příbuznými dokumenty (v tomto velmi omezeném příkladu je dokument 1 podobnější dokumentu 2 než dokumentu 3).

V plné implementaci Doc2Vec se provádějí sofistikovanější operace a váhy se upravují uvnitř neuronové sítě. Níže uvedené výsledky pracují s tímto úplným modelem Doc2Vec.

Příklad konkrétního označeného dokumentu se seznamem klíčových slov je uveden v tabulce níže:

| **Klíčové slovo** | **Klíčové slovo** | **Klíčové slovo** |
| --- | --- | --- |
| kopřivnice | limuzína tatra | limuzína |
| není dnes vyráběno | není vyráběno | raritní |
| vyrobit | auto | tatra |
| představit | veřejnost | osobní |

Tabulka 7: Klíčová slova extrahována ze článku o kopřivnické limuzíně Tatra

Příklad označeného hlavního článku se seznamem klíčových slov je uveden v tabulce níže:

| Klíčové slovo | Klíčové slovo | Klíčové slovo |
| --- | --- | --- |
| krkonoše dávné řemeslo | portýr | živily generace |
| let | poslední žijící nosič | zemřít |
| práce nosiče | ikona | velká úpa |
| krkonoše | řemeslo | sněžka |

Tabulka 8: Klíčová slova pro článek o krkonošském nosiči

Po výpočtu podobnosti seznamu s ostatními články jsou výstupní články seřazeny podle jejich podobnosti. Výsledky článků relevantních k hlavnímu článku jsou uvedeny v tabulce IV.

| URL článku | **Podobnost** |
| --- | --- |
| autor-legendarniho-krtecka-se-narodil-pred-100-lety | 0.548 |
| zemrel-tvurce-filmu-sveraz-narodniho-lovu | 0.538 |
| zemrela-eva-truda-vidlarova-z-brnenskeho-provazku-bylo-ji-73-let | 0.537 |
| zemrela-francoise-cactus-zpevacka-kapely-stereo-total | 0.523 |
| zemrela-prvni-dama-ceskeho-sansonu-hana-hegerova | 0.518 |
| vitorazsko-je-ceske-radovaly-se-ze-scitani-lidu-pred-100-lety-lidove-noviny | 0.511 |
| zemrel-dirigent-levine-legenda-metropolitni-opery | 0.511 |
| hold-nocni-praze-devadesatych-let-vzdava-vystava-alter-ego | 0.509 |
| zemrel-lee-scratch-perry-jamajska-legenda-reggae-a-dubu | 0.507 |
| zemrel-l-g-petrov-z-kapely-entombed | 0.501 |

Tabulka 9: Výsledky nejpodobnějších článků

V této části jsou popsány experimentální výsledky našeho přístupu. Pro experimentální ověření našeho přístupu bylo vybráno 40 náhodných článků ze zpravodajského portálu *Idnes*.cz. Pro každý článek náš přístup doporučil 20 nejpodobnějších (relevantních) článků. Poté byly označeny relevantní a nerelevantní články. A nakonec byl vypočítán poměr navržených relevantních článků pro každý náhodný článek. Celkem jsme tedy analyzovali 800 článků, které doporučil náš přístup. Celý záznam tohoto testování je dostupný v příloze *Podobnost\_clanku\_srovnani.xlsx*. V prvním sloupci je uvedeno pořadí testovaného článku (pouze pro odlišení jednotlivých testovacích pokusů), ve druhém sloupci je uvedena adresa URL článku. Poslední sloupec obsahuje poměr relevantních článků, který vyjadřuje, kolik doporučených článků je pro uživatele skutečně relevantních.

Průměrný poměr relevantních článků byl 77 %. Články z kategorie Sport měly velmi vysoký poměr relevantních článků (100 %). Je to dáno pravděpodobně tím, že články v kategorii Sport mají relativně podobný obsah, opakující se klíčová slova (gól, výhra, prohra, vítězný, poražený), opakující se názvy týmů, sportovců než články v ostatních kategoriích a tato kategorie také obsahuje největší podmnožinu článků ze všech kategorií.

* + 1. Doporučení na základě společných témat

Pro doporučení na základě společných témat byla využita metoda latentní dirichletovy alokace (LDA) [104]. Latentní Dirichletova alokace je v obecném smyslu statistická metoda, která se využívá pro vysvětlení toho, proč jsou si některá pozorování podobná. Metoda LDA původně vznikla v roce 2000 za účelem v oboru populační genetiky, kdy bylo použito pro zjištění přítomnosti strukturované genetické variability ve skupinách jedinců [105]. O tři roky později poté byla představena aplikace LDA pro strojové učení.

LDA je často využíváno pro modelování témat v rámci textu přirozeného jazyka, kdy se předpokládá, že pozorování jsou jednotlivá slova v rámci dokumentů (v případě této práce je dokumentem rozuměna část článku jako titulek a perex, či celý text, tzn. Titulek, perex i tělo článku dohromady) a těmto slovům v rámci dokumentů náleží určitá témata – tzn. dokument se skládá z několika témat, přičemž některá témata v rámci jednoho dokumentu převažují nad jinými. Model si tak „objeví“ v množině dokumentů několik témat a následné nově příchozí dokumenty zařadí do jednoho z těchto témat. Úkolem LDA je shlukovat slovy, termíny, slovní spojení, či celé fráze (záleží na zvolené strategii), které by měly mít společný význam v rámci témat. LDA však již neřeší pojmenování témat – to je úkol, který je většinou ponechán lidskému expertovi.

Tato metoda se nicméně ukázala jako záludná. Její výsledky v experimentální fázi nebyly často uskakující a evaluace byly paměťově velmi náročné, proto je využíváno LDA v tomto projektu pouze v omezené míře a LDA např. chybí pro své neuspokojivé výsledky ve finálním hybridním algoritmu.

* + 1. Skládání výsledků

Pro kombinaci více různých metod bylo využito jednoduchého skládání výsledků, kdy se výsledky omezí na 10 nejlepších výsledků všech 4 představených metod založených na obsahu a následně se dle hodnot 1-10 nastavených v paměti *Redis* vybírají postupně nejlepší výsledky ve zvoleném pořadí metod. Na prvních 4. místech se objevují 1. výsledky všech metod v pořadí TF-IDF, Word2Vec, Doc2Vec, LDA, zatímco na dalších místech už pak některé výsledky některých metod chybí (např. podíl výsledků LDA je malý kvůli nedokončeným záležitostem ohledně této metody). Výsledky tohoto skládání lze poté vidět i v případě nepřihlášeného uživatele po kliku na libovolný článek v sekci pod výtažkem článku.

* 1. Hybridní přístup

Jestliže byly implementovány různé metody založené na obsahu nebo uživatelích, další ambicí bylo kombinovat tyto přístupy dohromady a představit vlastní hybridní algoritmus přizpůsobený použití na zpravodajských článcích. Tento algoritmus je v podstatě řadícím algoritmem pro SVD metodu. Tato metoda je praxí ověřená a využívá jí mnoho zavedených systémů, avšak v této doméně doporučování článků by bylo záhodno, a to i například z důvodů zmíněných v podkapitole 3.3.7 využít obsahových podobností článků z důvodů a články zhlédnuté uživateli. Představený hybridní algoritmus pro zobrazování článků funguje v těchto krocích:

1. Nalezení podobných článcích metodou založenou na uživatelích metodou SVD.
2. Najdou se články, které zhlédnul uživatel (resp. na které prokliknul).
3. Pro každý pár článků doporučený SVD metodou a zhlédnutý uživatelem se vypočítají matice podobností založené na obsahových metodách TF-IDF, Word2Vec a Doc2Vec (LDA bylo v tomto případě vynecháno z důvodu neuspokojivých výsledků na některých typech článků).
4. Normalizace hodnot koeficientů podobností.
5. Každý z těchto algoritmů má navíc nastaven určitý koeficient zesílení vlivu, aby se zhodnotilo, že některá z metod vykazovala v experimentální části lepší výsledky. Tento koeficient lze nastavit v administrátorské sekci aplikace.
6. Osu X matice tvoří nyní články doporučené metodou SVD, osu Y články zhlédnuté. Hodnoty matice jsou podobnosti založené na obsahu. Tato matice existuje ve 3 verzích odpovídající podobnostem TF-IDF, Word2Vec, Doc2Vec zesílené výše zmíněným koeficientem vlivu. Tyto matice se poté sečtou do jedné.
7. Provede se řádkový součet prvků matice a získáme koeficient doporučení pro jeden ze článků doporučený SVD metodou.
8. Pokud navíc článek pochází z kategorie, která patří mezi uživatelovy oblíbené, vynásobí se součet dvěma.
9. Koeficient doporučení je navíc zesílen pro novější články hodnotou, kterou lze nestavit také v administrátorské sekci, stejně jako počet hodin, který má uplynout v rámci 4 různých úrovní, aby bylo na doporučený článek dané nastavené zesílení využito (např. články novější než 1 den, týden, měsíc apod).
10. Seřazení výsledků.

Pro začlenění do hybridního algoritmu se nabízejí také další vlastnosti článků jako například počet zhlédnutí konkrétního článku všemi uživateli. Tento prvek by byl do hybridního algoritmu byl připraven, avšak byl nakonec vynechán kvůli nedostatku testovacích dat, jenž by mohl zkreslovat výsledky.

* + 1. Fuzzy hybridní přístup

Výše popsaný algoritmus obsahuje obsahuje nastavitelné hodnoty zesílení nových článků a koeficient vlivu obsahových metod. Administrátor takto může učinit zásahy do algoritmu. Tato hodnota je však přesně nastavená a chybí zde jakákoliv míra nejistoty, kterou bychom potřebovali pro tento případ modelovat. Tato nejistota se týká především výsledků podobnosti článků a vztahu směrem k čerstvosti článků. Kromě toho bychom chtěli zachytit, jak věříme dané metodě založené na obsahu na základě experimentálních výsledků a zkušeností. Pokud například metoda neprokazovala tak dobré výsledky jako jiná, chtěli bychom ve výsledném řazení například upřednostnit raději novější články. Tento vztah přitom není jednoznačný a lineární.

Pomocí fuzzy přístupu tedy chceme modelovat naši důvěru v danou metodu založenou na obsahu jako takovou. Například v případě metody Word2Vec věříme více samotné kosinové podobnosti článku, nicméně v případě Doc2Vec jsme z výsledků našich experimentů pochopili, že kosinová podobnost nebyla tak přesná v poskytování podobných výsledků ke konkrétnímu článku, a proto chceme spíše využít čerstvosti článku, protože zpravodajský portál, ale i blogy mají obvykle tendenci své nejnovější články posouvat výše než ty staré (viz analýza čerstvosti článků *Idnes* na titulní straně).

|  |  |  |  |
| --- | --- | --- | --- |
| **1 den** | **2 dny** | **3 dny** | **> 3 dny** |
| ≈66 % | ≈23,53 % | ≈5,88 % | ≈3,92 % |

Tabulka 10: Čerstvost článků na titulní straně iDNES.cz

Pro fuzzy inferenci byla využita metoda Mamdani. Výstupem obou inferencí popsaných níže je koeficient, který vynásobí podobnost článků zhlédnutých uživatelem a článků, které doporučuje metoda SVD (viz výše popsaný hybridní algoritmus), resp. průběžný výsledek počítaného koeficientu. Pro návrh fuzzy metody bylo využito programu *LFLC*. Pro implementaci do programovacího jazyku *Python* poté knihovny *Simpful* [106].

Modelování problému čerstvosti článků pomocí fuzzy přístupu:

**Vstupní proměnné:**

similarity(Podobnost): U = [0; 1]; podobnost dvojic článků na základě obsahových algoritmů; Cosine-Similarity.

freshness(Čerstvost článku): V = [0; 5]; počet dní, jak je článek starý.

boosting\_coefficient: W = [0; 1]; poskytuje posílení na základě podobnosti a čerstvosti článku.

Pro fuzzy rozklad byl tentokrát zvolen menší počet proměnných a dny místo hodin, aby byla zachována vyšší úroveň abstrakce, což byl hlavní důvod fuzzy přístupu.

Fuzzy dekompozice univerza pro jednotlivé **výstupní proměnné**:

Similarity: very\_high (velmi vysoká), high (vysoká), medium (střední), low (nízká), very\_low (velmi nízká)

Freshness: very\_fresh (velmi čerstvé), fresh (čerstvé), current (stále aktuální), slightly\_old (spíše staré), old (staré)

Boosting coefficient: very\_high (velmi vysoký), high (vysoký), medium (střední), low (nízký), very\_low (velmi nízký)

Obrázek 33: Návrh fuzzy metody v programu *LFLC*. Na obrázku modelování proměnné „freshness“ představující čerstvost článku

**Fuzzy IF-THEN pravidla:**

R1 = "IF (similarity IS very\_high) AND ((freshness IS very\_fresh) OR (freshness IS fresh)) THEN (boosting IS very\_high)"  
R2 = "IF (similarity IS very\_high) AND ((freshness IS current) OR (freshness IS slightly\_old)) THEN (boosting IS high)"  
R3 = "IF (similarity IS very\_high) AND (freshness IS old) THEN (boosting IS med)"  
R4 = "IF (similarity IS high) AND ((freshness IS very\_fresh) OR (freshness IS fresh)) THEN (boosting IS high)"  
R5 = "IF (similarity IS high) AND (freshness IS slightly\_old) THEN (boosting IS very\_high)"  
R6 = "IF (similarity IS high) AND (freshness IS current) THEN (boosting IS med)"  
R7 = "IF (similarity IS high) AND (freshness IS old) THEN (boosting IS high)"  
R8 = "IF (similarity IS medium\_high) AND ((freshness IS fresh) OR (freshness IS current)) THEN (boosting IS high)"  
R9 = "IF (similarity IS medium\_high) AND ((freshness IS slightly\_old) OR (freshness IS old)) THEN (boosting IS med)"  
R10 = "IF (similarity IS medium\_high) AND (freshness IS very\_fresh) THEN (boosting IS med)"  
R11 = "IF (similarity IS low) AND ((freshness IS old) OR (freshness IS slightly\_old) OR " \  
 "(freshness IS current) OR (freshness IS fresh) OR (freshness IS very\_fresh)) THEN (boosting IS low)"  
R12 = "IF (similarity IS very\_low) AND ((freshness IS old) OR (freshness IS slightly\_old) OR (freshness IS current) " \  
 "OR (freshness IS fresh)) THEN (boosting IS very\_low)"  
R13 = "IF (similarity IS very\_low) AND ((freshness IS very\_fresh) OR (freshness IS fresh)) THEN (boosting IS very\_low)"  
R14 = "IF (similarity IS med) AND (freshness IS very\_fresh) THEN (boosting IS high)"  
R15 = "IF (similarity IS med) AND ((freshness IS old) OR (freshness IS slightly\_old) OR (freshness IS current)) THEN (boosting IS med)"  
R16 = "IF (similarity IS med) AND (freshness IS fresh) THEN (boosting IS med)"

Modelování problému důvěry v přesnost obsahové metody:

**Vstupní proměnné:**

similarity(Podobnost): U = [0; 1]; podobnost dvojic článků na základě obsahových algoritmů; Cosine-Similarity.

freshness(Čerstvost článku): V = [0; 5]; počet dní vyjadřující, jak je článek starý.

confidence\_coefficient: W = [0; 1]; poskytuje posílení na základě podobnosti, čerstvosti a důvěryhodnosti v danou metodu.

Pro fuzzy rozklad byl tentokrát zvolen menší počet proměnných a dny místo hodin, aby byla zachována vyšší úroveň abstrakce, což byl hlavní důvod fuzzy přístupu.

Fuzzy dekompozice univerza pro jednotlivé **výstupní proměnné**:

similarity: hig (vysoká), med (střední), sml (nízká)

freshness: fresh (čerstvé), current (stále aktuální), old (staré)

confidence\_coefficient: hig (vysoký), med (střední), sml (nízký)

**Fuzzy IF-THEN pravidla:**

Níže je uveden příklad pro TF-IDF:

R1 = "IF (Similarity IS small) OR (Freshness IS fresh) THEN (EnsembleRatioTfIdf IS small)"  
R2 = "IF (Similarity IS small) OR (Freshness IS current) THEN (EnsembleRatioTfIdf IS small)"  
R3 = "IF (Similarity IS small) OR (Freshness IS old) THEN (EnsembleRatioTfIdf IS small)"  
R4 = "IF (Similarity IS medium) OR (Freshness IS fresh) THEN (EnsembleRatioTfIdf IS medium)"  
R5 = "IF (Similarity IS medium) OR (Freshness IS current) THEN (EnsembleRatioTfIdf IS medium)"  
R6 = "IF (Similarity IS medium) OR (Freshness IS old) THEN (EnsembleRatioTfIdf IS small)"  
R7 = "IF (Similarity IS high) OR (Freshness IS fresh) THEN (EnsembleRatioTfIdf IS high)"  
R8 = "IF (Similarity IS high) OR (Freshness IS current) THEN (EnsembleRatioTfIdf IS high)"  
R9 = "IF (Similarity IS high) OR (Freshness IS old) THEN (EnsembleRatioTfIdf IS medium)"

V případě ostatních 2 obsahových metod jsou vhodně pozměněny hodnoty konsekventu pro vyjádření nižší nebo vyšší důvěry.

Například pravidlo 6 u Word2Vec varianty IF-THEN pravidel poté obsahuje vyšší důvěru v metodu, jinak řečeno: upřednostníme podobnost nad čerstvostí zprávy:

R6 = "IF (Similarity IS medium) OR (Freshness IS old) THEN (EnsembleRatioWord2Vec IS medium)"

* 1. Klasifikační metody

Poslední ambicí bylo vyzkoušet filtrování nejnovějších příspěvků, které by zavedlo doporučující metody také pro nejnovější články. Není totiž lehké určit, zda uživatel očekává od systému doporučení ze všech, či pouze z aktuálních článků. Zatímco by tak v rámci SVD byly doporučeny články i staršího data, v rámci hybridního algoritmu také, avšak ty novější by byly zvýhodněny, v rámci sekce nejnovějších článků by se provedlo filtrování pouze těch nejčerstvějších článků.

Pro toto filtrování bylo využito metod klasifikačních modelů strojového učení využívající metodu Random Forrest a Support Vector Machine, resp. modifikace Support Vector Classifier (SVC). Toto doporučení se vyhodnocuje na základě:

* Hodnocení uživatelem pomocí palců a hvězdami.
* Převedení textové reprezentace, tentokrát pomocí multi-jazykového modelu BERT.

Model natrénován na trénovací množině a ověřen na validační množině pak srovná uživatelem zvolenou relevanci s touto textovou reprezentací a vyhodnotí, které další, dříve neviděné (resp. v tomto případě nehodnocené) články budou pro uživatele relevantní: v případě binární volby je to zřejmé (předpovídaná hodnota 1), v případě hodnocení Likertovou stupnicí 1 až 5 je nastaven práh relevantních článků na predikované hodnocení >4.

1. PRAKTICKÉ OVĚŘENÍ

První fázi testování pro základní ověření modelů tvořilo testování expertní, tzn. prováděné autorem a vedoucím této práce. V případě, že jich bylo u dané metody možné využít, druhou fázi testování tvořily empirická, automatizovaná ověření bez nutnosti uživatelského testování. Třetí fázi testování pak tvořilo testování uživatelské.

* 1. Ukázka a práce se systémem

Bez registrace a přihlášení se uživateli zobrazují pouze nejnovější články. Může nicméně kliknout na nějaký ze článků a zobrazí se mu doporučení založená na obsahu.

Přihlášený uživatel již může začít s hodnocením pomocí:

* Palců (relevantní/irelevantní).
* Hvězd (Likertova stupnice 1 až 5).

Uživatel dále může jít do sekce uživatelského nastavení, kde si může nastavit oblíbené kategorie a klíčová slova, jenž jej zajímají.

Poté by se mu měly zobrazit další doporučené články na základě těchto voleb.

Pokud vykoná dostatečný počet hodnocení pomocí hvězd, za nějaký čas by se mu po obnovení stránky již měly zobrazit doporučené články pomocí metody SVD, která nabízí články, které se líbily ostatním uživatelům s podobnými preferencemi. Mělo by se také zobrazit hybridní doporučení, které jsou však také ovlivněny historií kliknutí na články, takže i tato spíše „nepřímá“ akce uživatele má svůj vliv na doporučení a nejnovější články filtrované klasifikátorem.

Pokud si uživatel zobrazí článek ze sekce filtrovaných nejnovějších příspěvků, již se mu později nezobrazí znovu. Příspěvky a doporučení jsou také aktualizována periodicky co několik hodin na nových článcích, které se objeví na portále *Idnes*.cz.

Vzhledem k autorským právům a licenčním podmínkám by bylo neetické, pravděpodobně i nelegální na webu zobrazovat obsah celých zpráv z *Idnes*.cz. Dostupný je tedy pouze úryvek, avšak na původní článek je možné projít klikem na tlačítko „Zobrazit původní“ v detailu článku.

Na snímcích níže lze vidět ukázky ze systému.

Obrázek 34: Ukázka hlavní strany webu Moje články

Obrázek 35: Ukázka uživatelské sekce

Administrátor pak může v aplikaci nastavit poměr výsledků a vliv čerstvosti výsledků. Kromě toho je přístupná například také sekce pro evaluaci obsahových algoritmů, kde lze hodnotit relevantnost doporučení na základě jednoho článku a odkud lze exportovat výsledky testování například ve formátu .CSV. Nachází se zde také další experimentální sekce, např. pro předzpracování textu nebo extrakci klíčových slov, některé z těchto dalších sekcí využitých v experimentální části pak však již nebyly dále udržovány a nemusí být tedy plně funkční.

Obrázek 36: Ukázka sekce administrátorského nastavení

Obrázek 37: Ukázka sekce evaluace obsahových metod

* 1. Tok uživatelských událostí a automatizované testování

Kromě unit a integračních testů byly připraveny rovněž prohlížečové testy pomocí rozšíření PHP frameworku *Laravel Dusk*. V automatizovaných testech jsou otestovány také některé níže popsané kroky v systému představující základní práci se systémem.

* + 1. Přihlášení uživatele.
    2. Registrace uživatele.
       - Registrace + přihlášení (současně).
    3. Ohodnocení libovolných článků palci.
    4. Ohodnocení libovolných článků hvězdičkami.

Hodnocení by mělo být 5 a více (jak palci, tak hvězdami). Pro filtrování na základě klasifikátoru ale může být potřeba i více.

* + 1. Libovolné prohlížení článků a hodnocení.

Pro výsledky hybridního algoritmu je potřeba kliknout alespoň na 3 články.

* + 1. Přechod do sekce uživatelské sekce.
* Zadání oblíbených kategorií.
* Zadání klíčových slov.
  + 1. Uživatel si může zobrazit doporučené články na titulní stránce a dále libovolně číst a hodnotit články.

Hybridní doporučení a filtrování nejnovějších příspěvků na základě klasifikátoru v tomto momentě ještě nemusí být dostupné (v závislosti na rychlosti práce uživatele a počtu hodnocení). Uživatel tedy může pokračovat libovolně v prohlížení nebo hodnocení nebo se na stránky vrátit později a po obnovení by se měly již i tyto druhy doporučení náročnější na zpracování zobrazit.

* 1. Validace a optimalizace obsahového přístupu

Tato část práce pojednává jak o optimalizaci výkonu, tak o optimalizaci parametrů modelů strojového učení a o metodách použitých pro validaci modelů a přístupů založených na obsahu.

* + 1. Optimalizace výkonu TF-IDF

U doporučujících systémů je vzhledem k optimalizaci výkonu nutné sledovat, zda existují v kódu části programů, u kterých dochází k opětovnému zpracování rutinních a opakujících se činností, výpočtů, nahrávání.

V rámci TF‑IDF například dochází opětovně k vektorizaci a transformaci vět všech dokumentů z celé datové sady. Při metodě „fit“ dochází k výpočtu průměru a rozptylu jednotlivých sloupců z dat. Metoda transformace pak transformuje všechny prvky pomocí příslušného průměru a rozptylu. Aby nebylo nutné takto neustále transformovat model, může být uložena řídká matice v rámci knihovny *Numpy* pomocí metody *numpy.savez(),* resp. *np.load()*. Je nutné pamatovat na to, že řídká matice („sparse matrix“) v rámci knihovny *Numpy* není pouze matice samotná, ale komplexní datová struktura, která obsahuje více součástí:

* samotná data (číselná reprezentace textů, resp. všechny nenulové prvky řídké matice),
* indexy matic – pole mapující prvky dat na korespondující sloupce v řídké matici,
* index „indptr“ pak spojuje prvky dat a indexů a mapuje je na řádky řídké matice,
* rozměr matice.

Předcházejícím krokem k vytvoření řídké matice je samotný převod kolekce textových dokumentů na maticovou reprezentaci v TF‑IDF podobě. Součástí tohoto kroku je také nahrání nadbytečných slov do objektu knihovny *Scikit-Learn* TfidfVectorizer. Součástí této operace také může být (volitelné) nahrání klíčových slov (např. ze souboru). U tohoto kroku se nabízí optimalizace. Datová sada se mění pouze při webovém scrapování a následném vložení nových článků do databáze. Pro potřeby praktické části této práce byla frekvence operace stahování a vložení nových článků z webu *Idnes*.cz nastavena na dobu co 4 hodiny jakožto kompromis vzhledem k zachování aktuálnosti článků a nevytížení databáze a skriptů zpracování (ačkoliv, pokud by začal být web *Moje články* využíván více, doba by se pochopitelně dala zkrátit na několik minut nebo dokonce sekund). Mezi tím ale nemusí docházet k opětovné inicializaci objektu TfIdfVectorizera nahrávání nadbytečných slov a je možné uložit vytvořenou řídkou matici do souboru. Následně je možné pomocí mechanismu zasílání zpráv (viz kapitola 2.1.6) notifikovat modul (resp. skript) s implementovaným TF‑IDF, aby aktualizoval řídkou matici na základě datové sady doplněné o nové články a uložil jí znovu do souboru.

V tomto projektu optimalizace uložení řídké matice do souboru přinesla časovou optimalizaci a nahrání matice ze souboru na disku bylo rychlejší operací než vytváření celé matice znovu. Při novém sestavování této matice trvá celý proces vyhledávání podobných článků na lokálním stroji při 30 pokusech průměrně 5,372 sekund, při načítání matice ze souboru to bylo 4,482 sekund.

Ještě před uložením však dochází k operacím fit a transform. Statisticky zde v tomto kroku dochází ke standardizaci dat (nulový průměr a směrodatná odchylka =1) a aplikaci transformace na vstupní množinu dat[107] [107]. Výstupem operace je komprimovaná řídká matice obsahující dvojice (id dokumentu [titulku nebo perexu], id tokenu odpovídající slovu) a příslušný koeficient TF‑IDF. Vzhledem k tomu, že v rámci těchto kroků dochází ke komprimaci řídké matice, jedná se o další faktor, proč optimalizace pomohla snížit časovou složitost operace převodu textu na TF‑IDF reprezentaci.

Tyto operace je nutné provádět jak nad daty všech dokumentů, tak nad daty hledaného dokumentu (což je obzvláště důležité zdůraznit, pokud se někdo vydá cestou neaktualizování trénovací množiny a vstupní hledané texty budou dříve neviděné).

Kroky integrace metody TF‑IDF do mé práce jsou následující:

Pozn.: Pro vytvoření datových rámců je použita knihovna pro *Python* *Pandas* implementující tuto datovou strukturu (výhody i nevýhody jsou diskutovány v podkapitole 2.1.3):

1. Nahrání všech článků z databáze (popř. cache databáze) do datového rámce příspěvků
2. Nahrání kategorií z databáze (popř. cache databáze) do datového rámce kategorií.
3. Spojení článků a kategorií do jednoho datového rámce.
4. Nahrání českých a obecných nadbytečných slov z textového souboru.
5. Pokud existuje soubor s již uloženou a vypočítanou komprimovanou řídkou maticí, kroky 6. a 7. se přeskočí.
6. Inicializace objektu vektorizace pro převedení textových dokumentů TfIdfVectorizer. Vstupem je určení datového typu a nadbytečná slova. Pro zrychlení operací byl objekt inicializován na datový typ *numpy.float32* oproti původnímu *numpy.float64*.
7. Aplikace metody fit a transform na celou sadu dokumentů.

Výstupem je *Python* struktura slovníku, kdy je klíčem dvojice (id\_dokumentu, id\_tokenu) a slovem desetinná hodnota TF‑IDF skóre.

* 1. Tyto operace se provádějí na text skládající se z částí:
     1. Název kategorie a nadpis.
     2. Výtažek článku.
     3. Klíčová slova.
     4. Trigramy a bigramy slov aplikované na text celého článku.
  2. Komprimovaná řídká matice je uložená do souboru.

1. Počítání kosinových podobností v rámci řídké matice TF‑IDF koeficientů (které jsou brány jako váhy u výpočtu kosinové podobnosti).
2. Přiřazení podobností původním článkům. Toto je možné díky zachování původních indexů datových rámců. Výsledkem dosavadních kroků je poté čtvercová matice podobnosti s jednotkovými diagonálními prvky (kosinová podobnost článku se sebou samotným je rovna 1):

Obrázek 38: Matice kosinových podobností

1. Vyhledání nejvyšších podobností hledaného článku s ostatními články v matici.
2. Převod dat do JSON (navrácena koncovka URL článku a příslušný koeficient podobnosti).

V konkrétním příkladu využití TF-IDF na datové sadě z *Idnes*.cz je možné vidět slova s nejvyšším koeficientem k danému článku:

Obrázek 39: Heatmapa TF-IDF skóre

**Problém metody TF-IDF**

Důležitější, než samotná hodnota TF IDF jsou však v případě tohoto použití metody navrácené výsledky podobných článků. Ač TF-IDF určuje poměrně přesně doslovnou podobnost s vyskytujícími se slovy v článcích, některé nevýhody metody TF‑IDF v kombinaci s kosinovou podobností lze dobře vidět při detailním pohledu na konkrétní článek z *Idnes*.cz a jeho výsledky. Jako problematický se ukázal být při testování metody například článek:

*„Zemřel poslední krkonošský nosič Helmut Hofer, ikona Velké Úpy“*

Jako podobné články se pomocí metody TF‑IDF a kosinové podobnosti navrátí z datové sady *Idnes*.cz články:

* *„Na Brněnsku auto dostalo smyk, vyletělo ze silnice. Jeho řidič zemřel“*
* *„Mladý řidič auta srazil cyklistu a ujel, senior na místě zemřel“*
* *„Cyklistka zemřela při srážce s autobusem, který jí patrně nedal přednost“*

Mezi desíti nejpodobnějšími články jsou tedy zahrnuty i články, které se někdy s nadsázkou označují jako „černá kronika“. Jedná se o zprávy o nejrůznějších tragédiích. Pokud zanedbáme, že z psychologických důvodů mohou být podobné články mezi čtenáři někdy až překvapivě oblíbené a čtené, chtěli bychom v tomto případě spíše doporučení na články týkající se Krkonoš, tzn. hor jako takových, popř. týkající se výletů. Čtenáře nejspíše nebude zajímat článek, protože „někdo zemřel“, ale protože má určitý vztah ke Krkonoším a horám obecně.

Z tohoto ohledu TF‑IDF také nabídl některé relevantní články. Kompletní výpis 10 nejpodobnějších článků ze serveru *Idnes*.cz k uvedenému článku nejpodobnějších článků je:

1. *„Žena zavalená nedělní lavinou v Krkonoších zemřela, je druhou obětí za zimu“*
2. *„Pec pod Sněžkou je vylidněná. Po obvyklém ruchu není ani památky“*
3. *„Taxisův příkop při Velké pardubické už nebude tak hluboký a půjde natrénovat“*
4. *„Sněžka za pandemie osiřela. Poláci slaví šampaňským, jejich lanovky jezdí“*
5. *„Na Brněnsku auto dostalo smyk, vyletělo ze silnice. Jeho řidič zemřel“*
6. *„Jeden ze tří chlapců, kteří se topili pod ledem v Žamberku, zemřel“*
7. *„Mladý řidič auta srazil cyklistu a ujel, senior na místě zemřel“*
8. *„Ve věku 79 let zemřel duchovní a chartista František Lízna“*
9. *„Žena a muž zemřeli po dvacetimetrovém pádu ze skály na Znojemsku“*
10. *„Skialpinistce, která zemřela, kamarádi nepomohli. Ani nečekali na záchranku“*

Z výsledků je vidět rozporuplnost kvality výsledků TF‑IDF. Zatímco v některých ohledech si vede dokonce lépe než níže zmíněné pokročilejší algoritmy (viz pozdější srovnání algoritmů), v některých případech nedokáže dobře rozpoznat kontext slov ve větě. Jako velmi relevantní lze například označit články 2. a 4. Dále se zde nachází články ohledně nebezpečí na horách mísící tragédie, ale i horská témata, které lze také označit jako poměrně relevantní vzhledem k tématu článku: 9. a 10. článek. Článek 8. v pořadí je relevantní méně, ale stále přijatelný (jakožto článek o významné osobnosti, která zemřela v úctyhodném věku). Poté jsou zde výše zmíněné články „černé kroniky“ informující o tragédiích nesouvisejících s tématem hor. Mezi výsledky je na 3. místě poté také článek naprosto irelevantní, kdy metoda TF‑IDF společně s kosinovou podobností navrátila tento článek tak vysoko patrně pouze na základě falešné asociace mezi názvy „Velká Úpa“ a „Velká pardubická“.

Z výše uvedených důvodů byl věnován důraz také na pokročilejší metody převodu přirozeného jazyka na numerickou podobu využívající strojového učení.

* + 1. Evaluace TF-IDF a dalších metod

V této sekci jsou představeny metody validace použité pro metodu TF-IDF. Tyto metody byly využity i pro první a třetí fázi testování u dalších obsahových metod doporučení. U TF‑IDF však nelze dobře využít fázi testování automatizovanými metodami. Protože nás zajímá navrácení a řazení relevantních výsledků, byly využity metody z příbuzného podoboru doporučujícím systémům z oblasti získávání informací („information retriaval“) a řazení výsledků pomocí strojového učení („learn-to-rank“).

**Precision and Recall**

Jako první byla využita metoda přesnosti a úplnosti (“Precision and recall”, někdy také překládáno jako „přesnost a úplnost“ nebo “přesnost a výtěžnost”). Jedná se o jednu z nejčastějších metod validace využívaných jak v oboru doporučujících systému, tak v příbuzném oboru získávání informací [108], resp. řazení relevantních výsledků (“learn-to-rank“). Přesnost je zde definována jako podíl počtu relevantních navrácených položek a počtu navrácených položek. Úplnost poté jako podíl relevantních navrácených položek a celkových relevantních položek.

Pro výpočet přesnosti byla využita vlastní datová sada zhruba 8 tisíc článků z *Idnes*.cz, ve které byly nalezeny výsledky relevance u XY náhodných článků. Protože není možné ručně počítat relevantnost v celé datové sadě, jsou použity varianty metrik „do k”, kdy je posuzováno *k = 0, 1, … n* výsledků z celkové množiny *N* výsledků (v toto případě *n=20* jakožto 20 výsledků, které byly modelem posouzeno jako ty nejrelevantnější). Na této omezené množině byla spočítána přesnost, úplnost, průměrná přesnost, harmonický průměr (míra F), střední průměrná přesnost a další metriky pro posuzování kvality modelů.

U metody přesnosti byla použita ještě jedna modifikace, kdy namísto počtu relevantních navrácených položek byl brán počet podobných navrácených položek, a to pro rozlišení dvou druhů výsledků: výsledků skutečně relevantních a výsledků podobných slovní zásobou, ale irelevantních vzhledem k významu, kdy je ve výsledcích očividně chybí určitá míra inteligence.

To znamená že dle výše uvedeného příkladu navrácených podobných článků ke článku o krkonošském nosiči Helmutu Hoferovi by jako relevantní články byly posouzeny články týkající se Sněžky a nebezpečí na horách, ale nebyly by již započítány články pocházející z „černé kroniky“. Do podobných článků by však tyto články započítány byly, ale nebyl do nich započítán irelevantní i nepodobný článek o Velké Pardubické. Dle výše uvedeného příkladu by tedy byly články rozděleny jako (číslo článku je dle pořadí umístění ve výsledcích):

**Relevantní**: Článek 1, Článek 2, Článek 4, Článek 8, Článek 9, Článek 10.

**Podobné**: Článek 1, Článek 2, Článek 4, Článek 5, Článek 6, Článek 7 Článek 8, Článek 9, Článek 10.

**Irelevantní**: Článek 3.

Po vypočítání přesnosti a úplnosti se pro kombinaci těchto dvou metrik často používá míra F (F-skóre, v angl. “F-score” nebo “F-measure”), která je harmonickým průměrem přesnosti a úplnosti [109] [110].

Přesnost a úplnost se dá rozepsat dle falešně pozitivní, falešně negativních a pravdivě pozitivních, pravdivě negativních výsledků také jako:

Míra F se poté dá zapsat jako:

Pro konkrétní příklad výsledků článků tedy dle výše uvedených výsledků:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Navrácených** | **Podobných** | **Relevantních** | **Podíl podobných**  **(Similarity)** | **Podíl relevantních (Precision)** |
| 10 | 9 | 6 | 90 % | 60 % |

Tabulka 11: Podíly výsledků v rámci výpočtu Precison-Recall

Problém výpočtu části “Recall” je ten, že v reálných situacích většinou neznáme celkový počet relevantních článků. U problému řazení výsledků je však možné výpočet úplnosti (“Recall”) zanedbat díky metodám průměrné přesnosti (“Average precision”, někdy označované pouze zkratkou “*AP*”), ve které se počítá pouze s omezeným počtem dokumentů *n* z celkového počtu dokumentů *N* a přesnost je vypočítána do určitého pořadí výsledku *k*. Hodnota “recall” se pak může omezit pouze na 0 nebo 1 [111] v závislosti na tom, zda je dokument relevantní nebo nikoliv, úplnost se tedy poté stává relevancí v pořadí *k*, tedy *rel(k)*. U nerelevantních dokumentů je zanedbána i přesnost, naopak u relevantních dokumentů se hodnoty přesnosti sčítají a poté dělí počtem relevantních dokumentů *REL*.

Více o této metodě a příbuzné metodě střední průměrné přesnosti (MAP) je zmíněno níže v této podkapitole.

Další metodou využívanou pro nemožnost započítání části úplnosti z důvodů uvedených výše je metoda přesnosti v k (“Precision at k”, někdy značeno jako „Precision @k“), kdy se počítá s tím, že úplnost není vyžadována i vzhledem k faktu, že uživatel chce pouze několik nejlepších výsledků, nikoliv procházet úplně všechny dokumenty. Přesnost se tedy používá počítá pouze do určitého pořadí nejlepších *n* výsledků z celkového *N* počtu dokumentů (např. nejlepších 10 nebo 20 výsledků z datové sady zhruba 8100 dokumentů) a do nějaké pozice (umístění v pořadí) *k*. Jestliže tedy nepotřebujeme počítat všechny relevantní dokumenty, ale pouze *n* relevantních dokumentů, můžeme spočítat i úplnost v pořadí *k* (“Recall at k”, někdy značeno jako „Recall @k“).

V tomto případě se jako falešně negativní výsledky myslí všechny ty pravdivě pozitivní výsledky, které se ještě nacházejí ve výsledcích do *n*. Zatímco přesnost („Precision“) tedy bude na nejnižší pozici z *n* (v *k=n*)nižší než na první pozici s tím, jak budou postupně přibývat falešně pozitivní příklady, tzn. dokumenty, které budou navráceny, ale nebudou relevantní a v průběhu zvyšování *k* bude kolísat (pochopitelně kromě příkladu, kdy jsou všechny články relevantní), úplnost („Recall“) bude naopak stoupat s tím, jak bude v pozici *k*ubývat počet článků, které jsou pravdivě pozitivní (relevantní), ale nebyly zatím odhaleny. Úplnost znevýhodňuje první pozice výsledků i pokud jsou relevantní, a to i když není možné, aby je přeskočily v pořadí další relevantní výsledky (logicky nelze, aby první dva relevantní dokumenty na pozici 1. a 2. přeskočilo dalších např. 11 relevantních článků, které se umístily až na dalších pozicích). Jakmile se však pokračuje v pořadí dále (*k* se zvyšuje) a počet relevantních článků na pozicích *k*až *n* klesá, stoupá pak i úplnost.

Výpočet harmonického průměru (F-skóre) poté pouze upravíme pro výpočet pouze *k* výsledků je následující:

Při využití představené metody přesnosti v *k* a úplnosti v *k* můžeme využít ještě alternativní zápis průměrné přesnosti, kde zanedbáme počet relevantních dokumentů:

Ve výše zmíněném případu konkrétně:

Dále můžeme spočítat metodu DCG (“Discounted Cumulative Gain”, „Snížený kumulativní zisk“, „Zlevněný kumulativní zisk“) a NDCG (“Normalized Discounted Cumulative Gain”, „Normalizovaný DCG“).

Myšlenkou metody DCG je změřit užitečnost (“gain”) daného dokumentu v rámci vyhledávacích výsledků na základě jeho pozice ve výsledcích, protože ani vysoce relevantní výsledky nejsou příliš užitečné, pokud jsou umístěny ve výsledcích na příliš nízké pozici. Předchůdcem metody DCG byla metoda CG (“Cumulative Gain”), která ale pouze sčítala výsledky relevance bez započítání pořadí výsledků. Část “Discount” u výpočtu DCG je tu proto zavedena pro penalizaci níže umístěných výsledků, kdy je na základě *i + 1* pozice součet relevance logaritmicky snižován:

Existuje ale také další varianta výpočtu DCG, u které se bere větší důraz na relevantní výsledky [112]. Jedná se tedy o jakýsi kompromis mezi úplným zanedbáním pořadí a kladením velkého důrazu na pořadí.

Obrázek 40: Srovnání 2 variant vzorců DCG. Pro binární relevanci se překrývají, tzn. nezáleží na variantě použitého vzorce

Jak je však vidět i z grafu, tyto metody fungují v případě binární relevance (relevantní, irelevantní) stejně [113] (překrývají se).

DCG má však v některých případech nežádoucí matematickou vlastnost díky kumulativní povaze metody, kdy jsou výsledky s vysokým počtem navrácených prvků stavěny nad výsledky s nižším počtem navrácených dokumentů.

U metody NDCG se tak započítávají pouze relevantní výsledky z *n* výsledků. V rámci NDCG počítáme část IDCG („Ideal discounted cumulative gain“), jenž zastupuje jakousi ideální hodnotu DCG – tedy DCG započítávající pouze ideální (relevantní) výsledky, které mají hodnotu relevance rovnou nebo větší určitému prahu. IDCG se počítá stejně jako DCG, ale nebere se již v potaz všech *n* výsledků z celkového *N* počtu dokumentů v datové sadě, ani samotné *k*, ale místo toho ještě více omezíme počet výsledků na počet pouze relevantních výsledků s hodnotou relevance odpovídají zvolnému prahu *p*. To se značí se většinou jako*|RELp|*. Tzn. u binárního hodnocení relevance (relevantní/nerelevantní) bude práh *p = 1* a počet relevantních výsledků |*RELp|* bude tedy odpovídat počtu hodnot *relevance = 1*. U nebinárního hodnocení relevance hodnocením např. Likertovou stupnicí (1 je nejhorší, 5 je nejlepší), můžeme nastavit práh např. *p = 3* nebo *p = 4* (podle toho, jak usoudíme, jaké hodnocení značí relevanci výsledku).

Jedná se o normovaný výsledek s oborem hodnot 0,0 až 1,0; který je možné využít k porovnání různých dotazů nad jinými dokumenty nebo pro porovnání různých modelů [114].

NDCG se poté tedy vypočítá jako poměr mezi skutečným DCG a ideálním DCG nebo:

Pro výpočet IDCG a NDCG musíme výsledky seřadit dle relevance. Protože v ukázce máme jen binární hodnocení relevance, relevantní články jsou posunuty před nerelevantní a pořadí při shodě relevancí zůstává stejné (resp. je dáno původní kosinovou podobností, podle které byly výsledky seřazeny).

Jelikož je průměrná přesnost metodou, která posuzuje pouze jeden dotaz a jeho příslušný jeden seznam relevantních výsledků. Pokud však chceme posoudit výsledků více, je nutné využít také střední průměrnou přesnost („Mean Average Precision“, „MAP“). Jedná se o prostý výpočet podílu součtu průměrné přesnosti více různých dotazů a počtu dotazů *Q*.

Někdy se také používá průměrná přesnost v *k* („Average Precision at k“, „*AP* @k“)

(Average Precision @k), kdy namísto počtu relevantních dokumentů z *n* (např. nejlepších 10 nebo nejlepších 20) se počítá průměrná přesnost do *k*.

Vidíme, že poslední hodnota v *k = n* odpovídá před tím vypočítané průměrné přesnosti pro všech *n* výsledků.

* + 1. Evaluace Word2Vec

Protože bylo vhodné nějakým způsobem model evaluovat ještě před samotným testováním lidským expertem nebo uživatelem, byla provedena evaluace kvality modelů Word2Vec (vlastního, natrénovaného na *Idnes*.cz a modelu projektu *FastText* natrénovaného na české *Wikipedii*) pomocí srovnání podobnosti dvojic slov [115]. Pro tento účel byla využita datová sada *WordSim353* [116] obsahující expertně sestavený seznam podobnosti 353 dvojic slov (podstatných jmen) a jejich číselné podobnosti. Českou verzi *WordSim353* sestavil *Institut formální a aplikované lingvistiky* z *Fakulty matematiky a fyziky* *Karlovy univerzity* [117] [118] [119]. Druhá datová sada je sada *Questions-Words* obsahující více než 15000 čtveřic analogických slov. V rámci čtveřic jsou přitom vždy 2 slova analogická, např. máme slova:

*Berlín, Německo, Londýn, Anglie*

Model v takovém případě musí odhalit, že slova Berlín a Německo jsou si podobnější než slova Londýn a Anglie (resp. vektory reprezentující tato slova). Datová sada obsahuje jak zeměpisné názvy, tak podstatná nebo přídavná jména.

Tato sada však nebyla pro český jazyk k nalezení. Vzhledem k délce této datové sady bylo tedy využito dávkového překladu pomocí *Python* knihovny *Deep Translator* [120]. Výsledný český strojově přeložený *Questions-Words* data set byl poté veřejně nasdílen na GitHub pro veřejnost:

<https://github.com/patmull/questions-words-cz>.

Výstupy evaluace párů slov jsou:

* Pearsonův korelační koeficient s p-hodnotou oboustranného testu.
* Spearmanův koeficient pořadové korelace s p-hodnotou oboustranného testu.
* Podíl párů s neznámými slovy.

Pro korelaci si stanovíme nulovou a alternativní hypotézu jako:

*H0: „****Neexistuje*** *lineární vztah mezi expertně posouzenými hodnotami podobnosti a podobnostmi změřenými modelem.“*

*H1: „****Existuje*** *lineární vztah mezi expertně posouzenými hodnotami podobnosti a podobnostmi změřenými modelem.“*

V případě využití modelu *FastText* natrénovaného na *Wikipedii* můžeme v obou případech **zamítnout** *H0*. Výsledky počítání podobnosti párů slov v rámci modelu od expertně sestavených podobností párů slov mají statisticky významnou závislost. Pearsonův i Spearmanův korelační koeficient však naznačují vyšší závislost mezi veličinami. Výrazně nižší je podle očekávání podíl vynechaných slov z důvodu chybějících slov ve slovníku (pouze ~5 % oproti ~38 %).

Navíc byly také posouzeny podobnosti párů slov na limitovaném modelu *FastText* obsahujícím 87 000 vektorů slov. Ten byl vytvořen z důvodu výrazně rychlejšího načítání modelu po tomto omezení.

Word pairs evaluation on Limited *FastText* model trained on cs.*Wikipedia*:

Pearson: correlation\_coeff=0.667,   
pvalue =2.730e-38  
Spearman: correlation\_coeff=0.689,

pvalue=2.2483424942569187e-43)  
Out-of-vocab ratio: 15.297

V tomto posledním případě také zamítáme nulovou hypotézu a můžeme výsledky interpretovat jako statisticky významnou souvislost mezi podobností určenou expertně a podobností určenou modelem strojového učení. Zajímavé je, že hodnota korelačního koeficientu se příliš nezměnila po vypuštění vektorů slov. Podíl slov nacházející se mimo slovník podle očekávání významně vzrostl (~15,23 %), pořád se však jedná o hodnotu výrazně menší než ~38 % nenalezených slov v případě *iDNES.cz*.

Evaluace na základě analogických slov byla prováděna pomocí *Gensim* implementace skriptu *Compute-accuracy* z původního Word2Vec zveřejněného *Google*m jenž byl implementován v jazyce C [121] [122]:

Analogies evaluation of *FastText* on *Wikipedia*.cz model:

0.360

Analogies evaluation of *FastText* on *Wikipedia*.cz limited model:

0.350

Srovnání těmito metodami ještě nemusí nutně znamenat, že je daný model lepší nebo horší pro případ užití ke kterému je určen, tzn. k rozpoznávání podobných vět a souvětí článků, resp. doporučení článků, což je úkol dost rozdílný od rozpoznávání podobnosti dvojic slov. Binární posouzení podobností slov je navíc přísné. V ideálním případě by bylo vhodné analogie posuzovat stupnicí a na základě většího počtu synonym.

Toto srovnání však nabízí dobrou základní představu o připravovaných modelech a jejich ztrátách.

Vysvětlující faktory, které mohou ovlivňovat kvalitu modelu založeného na vektorech slov jsou obecně:

* velikost trénovací množiny,
* kvalita dat v trénovací množině,
* velikost vektorů (počet prvků vektorů),
* samotný trénovací algoritmus [123].

Pokud se pak zaměříme konkrétně na parametry trénování Word2Vec a implementaci od *Gensim*, pak máme možnost upravit zejména tyto klíčové parametry:

* Parametr sgurčuje techniku reprezentace slov pomocí vektorů, tzn. Continuous-bag-of-words (CBOW) nebo Skip-Gram.
  + 0 =CBOW nebo 1 =Skip‑Gram

Dle způsobu fungování těchto dvou technik (viz výše) a jejich způsobu předpovídání můžeme předpokládat, že CBOW bude lépe zvládat syntaktické úkoly, zatímco Skip-Gram sémantické úkoly (např. CBOW by mělo teoreticky upřednostňovat například podobnost slov “hokej” a “hokejový”, Skip-Gram může oproti tomu hodnotit jako velmi podobná například i slova “hokej” a “fotbal”). Jelikož je vstupem převodu slovo na vektor u Skip-Gramu jediné slovo, CBOW má tendenci přikládat velkou váhu častým slovům (jedná se o typický problém strojového učení přetrénování, “overfitting”). Jeho vstupem je několik slov ze sousedství, které se v podobném kontextu vyskytují v rámci korpusu mnohokrát. techniky Skip-Gram, který bere jako vstup jediné slovo sice také budou upřednostněny často vyskytující slova, nebudou zde však tolik znevýhodněny slova spíše raritní.

* Parametr hsurčuje funkci trénování. 1 =Softmax. 0 =negativní vzorkování. Negativní vzorkování bylo představeno Mikolovovým týmem v rámci Word2Vec za účelem maximalizace podobnosti slov objevujících se v rámci stejného kontextu a minimalizace podobnosti slov v kontextu rozdílném. Probíhá na základě náhodného výběru několika slov (doporučuje se dle velikosti datové množiny například 5 až 20 [124]). Počet těchto slov lze ovlivnit parametrem *negative*. Pro využití negativního vzorkování je tedy nutné mít nastaveny parametry na hs=1a nenulový *negative* (výchozí nastavení).
* Parametr epochsurčuje počet iterací přes celý korpus, které jsou provedeny při trénování.
* Parametr vector\_sizeurčuje rozměr vektorů reprezentující slova.
* Parametr *sample* určuje práh pro konfiguraci náhodné penalizace slov vysokého počtu výskytů (dochází k převzorkování takovýchto vektorů slov). *Gensim* uvádí jako smysluplný rozsah parametru 0 až 1·10-1.
* Parametr windowurčuje okno (vzdálenost) mezi aktuálním a předpovídaným slovem ve větě, tj. jak široký kontext slova ve větě se bere v potaz, např. hodnota 5 znamená, že se berou v potaz 2 slova před a 2 slova za aktuálním slovem. Parametr musí být alespoň 1 (nemůže být 0).
* Parametr min\_counturčuje práh počtu výskytů slov potřebných pro to, aby se slovo bralo při trénování v potaz.

V případě problémů s výkonem (při trénování na méně výkonném stroji nebo při využití velkého množství slov) lze pak využít parametr:

* max\_vocab\_sizepro limitování počtu slov nahrávaných do RAM. Jestliže bude existovat ve slovníku více slov, než jich je zadaných tímto parametrem, automaticky se vyloučí méně častá slova.

U Word2Vec byly evaluovány parametry negative, vector\_size, window, min\_count, epochs, sample a softmax. Pro každý z nich byl zvolen rozumný rozsah na základě běžných praktik využívaných při trénování Word2Vec modelů. Rozsah parametru sample se pak pohybuje od 0 do 1e-05 na základě doporučení v dokumentaci *Gensim* [124]. Konkrétně: lze vidět rozsah níže v grafech četností náhodně zvolených parametrů.

Obrázek 41: Grafy četností náhodně zvolených parametrů modelu Word2Vec

| **Descriptive Statistics** | | | | | | | | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Model\_Variant** | | **Negative** | | **Vector\_size** | | | **Window** | | | **Min\_count** | | | **Epochs** | | | **Sample** | | | **Softmax** | | |
| Valid |  | 400 |  | 400 | |  | 400 | |  | 400 | |  | 400 | |  | 400 | |  | 400 | |  | 400 | |  |
| Missing |  | 0 |  | 0 | |  | 0 | |  | 0 | |  | 0 | |  | 0 | |  | 0 | |  | 0 | |  |
| Mean |  | 0.477 |  | 10.088 | |  | 214.470 | |  | 8.240 | |  | 4.185 | |  | 24.950 | |  | 0.015 | |  | 0.465 | |  |
| Std. Deviation |  | 0.500 |  | 4.027 | |  | 121.048 | |  | 6.354 | |  | 3.788 | |  | 3.989 | |  | 0.033 | |  | 0.499 | |  |
| Minimum |  | 0.000 |  | 5.000 | |  | 50.000 | |  | 1.000 | |  | 0.000 | |  | 20.000 | |  | 0.000 | |  | 0.000 | |  |
| Maximum |  | 1.000 |  | 15.000 | |  | 450.000 | |  | 20.000 | |  | 12.000 | |  | 30.000 | |  | 0.100 | |  | 1.000 | |  |
|  | | | | | | | | | | | | | | | | | | | | | | |

Tabulka 12: Popisné statistiky srovnávaných parametrů modelů Word2Vec

Níže je vidět mapa korelací statistických testů a nastavených parametrů pro první model. Korelace statistických testů byly pro lepší přehlednost oříznuty (byla zde však patrná očekáváná kladná významná korelace mezi výsledky testu analogií a párů slov).

Obrázek 42: Korelační „teplotní“ mapa hodnocených parametrů a provedených testů

Pro posouzení vlivu parametrů na hodnoty byla využita metoda SelectKBest z knihovny *Scikit-Learn* využívající statistický test ANOVA. Jak SelectKBest, tak ANOVA se používá často pro posuzování vlivu jednotlivých veličin vstupujících do modelu [125] [126], avšak také právě pro testování hyperparametrů u algoritmů strojového učení s učitelem [127].

U učení bez učitele, resp. více přesně u učení s vlastním dohledem („self-supervised learning“) jako v našem případě nám tento postup poskytuje alespoň nějaké indicie pro to, jaké modely vybrat k nasazení, kdy bude možné sledovat relevanci doporučení určenou lidským posouzením z již úzkého výběru modelů.

Evaluace byla provedena na 400 náhodných unikátních kombinací parametrů (inspirováno metodou Random Search [128]).

Evaluace parametrů ukázala, že parameter *min\_count* má oproti osttaním parametrům zásadní vliv na využívané metody testování párů slov a ztátu slov, na test analogií má naopak zásadní vliv zvolený parameter *softmax*.

Na výsledky testu analogií měl naopak nějvětší vliv parameter *softmax*. Na výsledky testu slov nacházejících se mimo slovník měl největší vliv parameter *min\_*count, ostatní parametry měly a oproti tomu zanedbatelný vliv.

Na test párů slov měl při srováná se Spearmanovým koeficientem měl nejzásadnější vliv rovněž paramet min\_count. Stejně tak při srovnání s Pearsnovým koeficientem, se ukázal jako nejvíce vlivný parameter *min\_count*.

Obrázek 43: Vliv parametrů na výsledky testů analogií a dvojic slov

Zde je poté bližší pohledu na méně vlivné parametry:

Obrázek 44: Bližší pohled na méně vlivné parametry na výsledky testů analogií a dvojic slov

Pro první model byla vybrána nejúsoěšnější kombinace parametrů na testu analogií (~9,1 % uhádnutých analogií slov), která byla zároveň úspěšná na testu párů slov s Pearsonovou p hodnotou ~8,26e-28 a Spearmanovou p‑hodnotou ~2,7e-26. Model natrénován na této kombinaci vykazoval však vykazoval větší ztátu slov než většina z 20 nejpřesnějších modelů na testu analogií (~23,79% slov mimo slovník v rámci Out-of-Vocabulary test) vzhledem k vyšší hodnotě parametru minimálního počtu výskytu slov ve slovníku (min\_count = 12).

Model\_variant, Negative,Vector\_size,Window,Min\_count,Epochs,Sample Softmax) = 1,15,200,16,12,30,0.01, 0

Další model zahrnutý do nasazení a finálních evaluací pomocí lidského posuzování relevance doporučení byl:

* Model natrénován na původních parametrech *Gensim* Word2Vec třídy. Tento model však vykazoval pouze ~1,1 % přesnost určení analogií slov a ztrátu ~13,6 % slov.
* Model s nejmenší ztrátou slov podle Out-of-Vocabulary testu (~10,48 %), který zároveň uspěl v testu na WordSim323 datsetu (Pearsonova p hodnota = ~1,73e-22; Spearmanova p hodnota = ~2,16e‑22). Jeho úspěšnost v testu analogií byla však menší (~2,99 %).  
  (Model\_variant, Negative,Vector\_size,Window,Min\_count,Epochs, Sample Softmax) = 0, 15, 158, 5, 3, 20, 0.0, 0
* Kompromis mezi nejlepším výkonem na testech analogií ~5,35 % a s menší ztrátou slov ~13,59 % s rovněž úspěšnými testy na datasetu WordSim323.  
  (Model\_variant, Negative,Vector\_size,Window,Min\_count,Epochs, Sample Softmax) = 1, 15, 250, 20, 5, 25, 0.0, 0

*Pozn.: 0 = CBOW; 1 = Skip-Gram*

Finální vybrané modely tedy měly parametry:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model num.** | **Model variant** | **Negative sampling** | **Vector\_size** | **Window** | **Min\_count** | **Epochs** | **Sample** | **Soft-max** |
| 1 | Skip-Gram | 15 | 200 | 16 | 12 | 12 | 0,01 | 0 |
| 2 | All parameters default | | | | | | | |
| 3 | CBOW | 15 | 158 | 5 | 3 | 20 | 0,0 | 0 |
| 4 | Skip-Gram | 15 | 250 | 20 | 20 | 5 | 0,0 | 0 |

Tabulka 13: Parametry variant modelů iDNES.cz

Tyto variant modelů byly poté využity pro testování lidsky posouzené relevance.

Další variantou byl vlastní model natrénován na *Wikipedii*. Na rozdíl od modelu *FastText* byl zde text předzpracován včetně vytvoření trigramů v textu. Vzhledem k náročnosti trénování Word2Vec modelu nebylo prověřeno 400 variant náhodně vybraných parametrů jako v případě kratšího datasetu *Idnes*.cz, ale i tak se podařilo ve variantách parametrů najít model, který prokazoval na analogiích slov vyšší přesnost než model *FastText*: ~37,16 % oproti ~35,84 %. Jeho parametry byly:

(Model\_variant,Negative,Vector\_size,Window,Min\_count,Epochs,Sample, Softmax) = 1, 10, 200, 16, 3, 25, 0.0, 0

Nejdůležitější zjištěné poznatky po evaluaci Word2Vec byly následující:

* Modely *Idnes* vykazovaly nízkou úspěšnost při určování slovních analogií. Model s nejúspěšnější kombinací parametrů dokázal odhadnout trojice slovních analogií přibližně v 9,1 % případů.
* Vyhodnocení analogií předtrénovaného modelu *FastText* na *Wikipedii*.cz: ~35,8 % - Vyhodnocení analogií předtrénovaného modelu *FastText* na omezeném modelu *Wikipedia*.cz bylo jen o něco nižší: ~35 %.
* Nejlepší varianta vlastního modelu Word2Vec natrénovaná na *Wikipedii* vykazovala přesnost při určování analogií slov 37,16 %. - Všechny vybrané varianty modelu vybrané podle výsledků testování analogií musely projít testem dvojic slov (na datové sadě *WordSim353*), aby mohly být vybrány pro závěrečné lidské hodnocení na úloze podobnosti dokumentů.

Naše experimentální zjištění byla následující:

* Některé dobře fungující modely v testování analogií vykazovaly při závěrečném hodnocení špatné nebo relativně špatné výsledky (varianty „Word2Vec *FastText*“ a „Word2Vec *Wikipedia*“).
* Dvě z variant modelu Word2Vec s parametry vybranými podle výsledků testů analogií vykazovaly o něco horší výsledky než model Word2Vec natrénovaný na výchozích parametrech.
* Kompromis mezi testy analogií a poměrem mimo slovní zásobu stejně jako nejvyšší výkon v testu analogií nepřinesl zlepšení oproti výchozí variantě parametrů Word2Vec.
* Naopak model Word2Vec s nejnižší ztrátou slovní zásoby si vedl výrazně lépe než varianty Word2Vec, Doc2Vec natrénované na výchozích variantách a v závěrečné úloze poskytování podobných článků překonal i metodu TF‑IDF.

Protože výše představená metoda může být nápomocná pro evaluaci této metody a protože neexistuje příliš informací a metod pro evaluaci Word2Vec ještě před ověřením na finální úloze (např. právě doporučování), byla tato metoda představena také v rámci článku *Using Word2Vec for news articles recommendations: Considering evaluation options for hyperparameter optimization and different input option*s.

Po testování uživatelů se ukázalo, že model natrénován dle výše zmíněných testů si vedl výrazně lépe než jiné metody (podkapitola 4.4.1).

Pokud by se i dále prokázalo, že existuje určitý vztah mezi úspěchem ve výše zmíněných testech a úspěchem ve finálním použití metody Word2Vec, jednalo by se o značné ulehčení práce s touto metodou. Tuto záležitost by tedy stálo za to prozkoumat dále.

* + 1. Evaluace Doc2Vec

Evaluace metody Doc2Vec probíhala obdobně k evaluaci Word2Vec.

Evaluovány parametry pro Doc2Vec byly obdobné k Word2Vec, jediný rozdílný parametr byl:

* dm {0;1}: definice tréninkového algoritmu. 1 = distribuovaná paměť (obdoba CBOW pro Doc2Vec); 0 = distributed bag-of-words (obdoba Skip‑gram pro Doc2Vec).

Zdaleka nejvýraznější vliv u Doc2Vec měl parametr min\_count.

Obrázek 45: Vliv parametrů u modelů metody Doc2Vec

Evaluaci Doc2Vec obdobnými metodami jako u Word2Vec nicméně překáželo, že výsledkem testu analogií byl v případě všech prohledávaných parametrů nulový výsledek. Nepodařilo se přitom zjistit, kde se nachází chyba. Rozhodl jsem se proto využít raději původní parametry modelu Doc2Vec, které vykazovaly slušné výsledky s relevancí okolo 74 %. Následně se však také přitom přišlo i na to, že pokud se využily parametry zjištěné ve srovnání Word2Vec pro Doc2Vec, relevance stoupla na 83 %. Přehledně lze vidět testované varianty a porovnání s variantami Word2Vec v následující tabulce.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model variant** | **(Mean) Average Precision** | **Precision**  **score** | **Balanced**  **Accuracies** | **DCG** | **F1-SCORE** | **NDCG** | **NDCG @5** |
| Word2Vec #3 | .86 | .54 | .66 | 5.32 | .69 | .88 | .76 |
| Doc2Vec #3 (vec\_size according to Word2Vec #3) | .83 | .56 | .72 | 4.82 | .6 | .84 | .69 |
| TF-IDF | .80 | .5 | .63 | 5.19 | .66 | .88 | .74 |
| Word2Vec #2 (default parameters) | .79 | .49 | .65 | 4.83 | .6 | .84 | .69 |
| Word2Vec #1 | .77 | .41 | .61 | 4.18 | .5 | .78 | .61 |
| Word2Vec #4 | .76 | .38 | .61 | 3.78 | .44 | .75 | .55 |
| Doc2Vec #2 (full text) | .74 | .3 | .5 | 4.17 | .46 | .82 | .59 |
| Word2Vec *Wikipedia* | .73 | .41 | .61 | 4.21 | .49 | .8 | .6 |
| Doc2Vec #1 (short text) | .63 | .25 | .5 | 3.25 | .34 | .71 | .47 |
| Word2Vec *FastText* | .32 | .1 | .4 | 1.37 | .12 | .43 | .24 |

Tabulka 14: Srovnání evaluací metod a variant modelů

* + 1. Evaluace a optimalizace LDA

V rámci metody LDA každému tématu náleží více různých termínů, které zde mají určité pravděpodobnostní rozdělení. LDA počítá s tím, že stejné termíny mohou náležet více různým tématům, avšak s jinou pravděpodobností. Např. slovo „míč“ se může objevit jak v tématu shlukující sportovní články o profesionálních sportech, tak v tématu shlukující články o zdravém životním stylu – např. míč pro cvičení; v takovém případě by ale nejspíše měl vzhledem k počtu výskytů větší pravděpodobnost termín míč u tématu shlukující sportovní články, kde se tento termín vyskytuje ve větším počtu článků.

Při vykonávání LDA nad články z *Idnes*.cz je například patrná podobnost termínů v rámci shluků č. 2 a 7, které shlukují sportovní témata:

Obrázek 46: Znázornění vzdáleností témat

Obrázek 47: Nejrelevantnější slova tématu č. 2

Obrázek 48: Nejrelevantnější slova tématu č. 7

Shluky číslo 10 a 13 jsou naopak od těchto shluků vzdálené (nenachází se zde příliš mnoho podobných slov). Jedná se o shluky s tématy technologickými (např. mobily, počítače):

U LDA může být problematické, že některé shluky nemusí být snadno pojmenovatelné. Například tento shluk č. 3 se zdá být záludný. Obsahuje termíny, které by se daly tématem pojmenovat jako „rodina“, popř. „pro dámy“. Převažují zde patrně termíny, resp. články, které v rámci *Idnes*.cz náleží kategorii „Ona” (LDA přitom o kategoriích explicitně povědomí nemá, resp. kategorie tvoří pouze jedno ze slov v celém řetězci). V této sekci se ale často vyskytují také témata celebrit („herečka“, „hrát“, „film“…).

Obrázek 49: Nejrelevantnější termíny pro téma 3

Podobně každému dokumentu náleží více různých témat, přičemž témata mají určité pravděpodobnostní rozdělení – některá témata náleží dokumentu více než jiná.

Při trénování modelu je LDA model aktualizován na základě EM algoritmu („Expectation maximization algorithm“). Jedná se o iterativní algoritmus využívající učení bez učitele. Vstupem je množina pozorování a výstupem jsou parametry statistického modelu. EM algoritmus střídavě vykonává učení a rozpoznávání. Na rozdíl od algoritmu k-means, u kterého se určuje příslušnost daného pozorování ke shluku binárně (patří / nepatří), EM algoritmus využívá aposteriorní pravděpodobnost, která udává pravděpodobnost klasifikace pozorování do shluku. Parametry jsou tedy EM algoritmem odhadovány tak, ať je maximalizována aposteriorní pravděpodobnost [129].

EM algoritmus se skládá z dvou hlavních kroků: krok E a krok M. Krok E má za úkol odhadnout chybějící pozorování na základě skutečných pozorování a aktuálního odhadů parametrů strojového učení pro trénování modelu pomocí podmíněné střední hodnoty [130] [131]. V rámci kroku M je maximalizována pravděpodobnostní funkce, přičemž se používá odhad chybějících údajů z kroku E namísto skutečných chybějících údajů [131].

Implementace LDA od *Gensim* je založena na práci *Online Learning for Latent Dirichlet Allocation* [132]. Umožňuje vytvoření vlastního LDA modelu na základě strojového učení bez učitele, kdy hlavní vstupní objekty tvoří korpus a slovník.

Dokument je v rámci *Gensim* prostý řetězec tvořen objektem typu textového řetězce (resp. textové posloupnosti objekt str).

Slovník shromažďuje všechna unikátní slova vyskytující se v rámci dokumentů. Každé slovo obsahuje unikátní identifikátor.

Korpusem se v rámci LDA rozumí množina dokumentů skládající se ze slov. V rámci *Gensim*, resp. *Python*u je korpus tvořen původní datovou strukturou seznamu (list), kdy jednotlivé prvky seznamu jsou další seznamy (dokumenty korpusu) shromažďující slova dokumentů. Korpus se v rámci *Gensim* ukládá serializací do souborového formátu *Matrix Market* (MmCorpus), kdy je text zabalen do matice termínů dokumentů (matice dokument-termín)**.** Tato matice popisuje termíny (slova), které se vyskytují v dokumentech. Řádky jsou dokumenty, sloupce slova. Po nahrání z disku je v rámci knihovny *Gensim* reprezentován korpus jako objekt, kterým je možné iterovat po jednotlivých dokumentech (nedochází tak k takovému zatížení paměti jako kdyby byl do ní nahrán celý korpus na začátku běhu programu).

Kompletní slovník sestavený z předzpracovaných výstupů české *Wikipedie*. Součástí slovníku je 2 029 741 unikátních slov (a vzhledem ke předzpracování někdy i dvojic slov oddělených znakem podtržítka).

Dictionary(2029741 unique tokens: ['aktualita', 'clanek', 'dobre', 'jazyk', 'jine']...)

Jak je již uvedeno v podkapitole 3.3.1, korpus z *Wikipedie* obsahuje 457 108 dokumentů (odpovídá počtu článků na české *Wikipedii* odstraněných od příliš krátkých článků) a 2 029 741 „features“ odpovídá počtu slov ve článcích na české *Wikipedii*.

Poslední hodnota („non-zero entries“) pak vyjadřuje počet nenulových prvků řídké matice TF-IDF (řídká matice, angl. „sparse matrix“ je matice, jejíž většina prvků je nulových).

Korpus sestavený z české *Wikipedie* je například:

MmCorpus(457108 documents, 2029741 features, 35024816 non-zero entries)

Vytvoření korpusu a slovníků je možné v rámci *Gensim* vytvořit v případě výstupů komprimovaného XML z *Wikipedie* ve formátu XML BZ2 pomocí příkazu make\_wiki. Stejně jako u jiných metod založených na podobnosti obsahu je ale i u LDA důležité správné předzpracování dat (v tomto případě textu). V tomto případě, kdy máme v plánu texty upravovat, pokud jsou používány vlastní texty, je nutné vytvořit jak vlastní korpus, tak slovník. Korpus je vytvořen iterací nad textem, kdy se text jednotlivých dokumentů převádí pomocí výše popsaného slovníku všech slov na strukturu „bag-of-words֧“ ve formátu dvojic (id slova, počet výskytů slova v dokumentu).

Např. slova z dokumentu číslo 1 500 jsou reprezentována takto (zkráceno):

[(28, 1.0), (32, 2.0), (39, 4.0), (62, 5.0), (64, 1.0), (117, 1.0), (174, 1.0), …]

Iterováním nad konkrétním dokumentem z korpusu pak lze vrátit původní slova a jejich výskyty:

Slovo 28 ("astrometrie") se nachazi v dokumentu 1.0krát.

Slovo 32 ("balon") se nachazi v dokumentu 2.0krát.

Slovo 39 ("celkem") se nachazi v dokumentu 4.0krát.

Slovo 62 ("eklipticky") se nachazi v dokumentu 5.0krát.

Slovo 64 ("exoticky") se nachazi v dokumentu 1.0krát.

Slovo 117 ("nahled") se nachazi v dokumentu 1.0krát.

Slovo 174 ("prevazujici") se nachazi v dokumentu 1.0krát.

Slovo 218 ("souhvezdi") se nachazi v dokumentu 2.0krát.

Slovo 228 ("statistik") se nachazi v dokumentu 1.0krát.

Slovo 235 ("svet") se nachazi v dokumentu 1.0krát.

Slovo 317 ("biogeografie") se nachazi v dokumentu 1.0krát.

Slovo 326 ("botanika") se nachazi v dokumentu 1.0krát.

Slovo 366 ("fytologie") se nachazi v dokumentu 1.0krát.

Slovo 398 ("lekarstvi") se nachazi v dokumentu 3.0krát.

Některé důležité parametry LDA modelu od *Gensim* jsou například:

* num\_topics – počet témat vytvořených z korpusu dokumentů
* alpha– určuje přirazení téma k dokumentu; vyšší hodnota parametru alfa obvykle znamená více jednotné kombinace témat pro dokumenty [133] [134]
* eta(někdy označován jako *beta* parametr)– určuje přiřazení slov k tématům; vysoká hodnota parametru alfa obvykle znamená více jednotné kombinace slov pro dokumenty [133] [134]
* chunksize (integer) – počet dokumentů v jednom kusu dokumentů; počet dokumentů, který je nahrán do paměti a pro výpočty než se model aktualizuje. Odpovídá počtu dokumentů, který bere v potaz EM algoritmus při kroku E. Parametr tedy přímo souvisí s přesností daného modelu (čím větší je tento parametr, tedy čím více dokumentů je v jeden moment bráno v potaz, tím jsou výsledky přesnější). Nevýhodou využívání většího počtu dokumentů je ale možné přetížení paměti.
* iteration(integer) – počet procházení celou množinou dokumentů (korpusu) algoritmem; konkrétně je to kolikrát je proveden krok expectation step („E step“) v rámci EM algoritmu.
* passes(integer) *–* počet samotných trénování v rámci jednoho trénovacího pokusu.
* update\_every(integer)– počet kusů dokumentů (počet dokumentu v jednomu kusu je dán předchozím parametrem *chunksize*); určuje kolik kusů dokumentů bude bráno v potaz v kroku M EM algoritmu.
* random\_state (integer) – vstupní inicializace náhodného prvku algoritmu („random seed“). Při srovnávání výsledků více různých modelů by měl zůstat parametr na stejné hodnotě.
* keep\_n(integer) – parametr udávající kolik termínů se bere v potaz; *Gensim* obsahuje metodu *filter\_extremes()*, která kromě vynechání extrémů v podobě příliš málo a příliš mnoho se vyskytujících slov, zredukuje slovník na 100 000 termínů. Pokud není metoda *filter\_extremes()* využívána (programátor např. chce zachovat i extrémy), tento parametr je možné nastavit na 100 000, popř. jinou hodnotu, zredukovat počet dokumentů a tím pádem ušetřit paměť
* per\_word\_topics(boolean) – určuje zde se bude vypočítávat pravděpodobnost příslušnosti k tématům pro každé slovo v dokumentu anebo pouze pro významná slova v rámci téma (nevýznamná budou z výpočtů úplně vynechána)
  + phi\_value(double) slouží poté jako práh pro zahrnutí (resp. přijetí) slova do výpočtů

Praktický dopad má kombinace parametrů chunksizea update\_everynásledující: pokud je chunksizenastaven například na hodnotu 100 000 a update\_everyna 1, je to ekvivalentní ke kombinaci parametr chunksize nastavený na 50 000 a update\_everyna 2. Druhá volba zabere více času, ale je výhodnější vzhledem ke konzumaci paměti.

V rámci LDA existuje na rozdíl od předchozích metod představených v této kapitole objektivní způsob měření kvality modelu pomocí koherence témat.

Pro evaluaci LDA se využívají většinou 2 způsoby měření koherence: metodou UMASSa CV [135].

V rámci této práce byl trénován LDA model nad různými počty témat a bylo sledováno, jak se mění hodnoty koherence. V případě CV se snažíme najít globální maximum funkce v intervalu <0;1>, stejně tak v případě UMASS hledáme nejvyšší hodnotu v intervalu <-14;14>. Pro metodu CV je nutné přidat „tokenizované“ texty (rozdělené na slova) a tím se běh algoritmu a jeho paměťová náročnost zpomalí. Toto srovnání tedy bylo provedeno pro menší datovou sadu *Idnes*.cz, zatímco pro *Wikipedii* byla zvolena metoda UMASS.

Vzhledem k časové náročnosti výpočtů nemělo smysl prohledávat všechny kombinace parametrů. Koherence byla tedy posuzována vzhledem k rozdílnému počtu témat jakožto parametru, který byl identifikován jako nejdůležitější a vzhledem k binárnímu parametru per\_word\_topics. Parametry alphaa etabyly nastaveny na hodnotu „auto“ pro automatický odhad parametru modelem z dodaného korpusu dokumentů. Počet kol trénování (parametr passes) byl intuitivně nastaven na hodnotu 20 jakožto kompromis mezi přesností a délkou výpočtu. Spolu s počtem iterací (parametr iterations) však byly tyto hodnoty ověřeny také pomocí údaje o konvergenci dokumentů v rámci textových hlášení o trénování modelu LDA od *Gensim* v režimu „DEBUG“ ukládaných pro zpětné dohledání do souboru. Ostatní parametry byly ponechány původní (viz dokumentace modelu LDA od *Gensim* [133]). Ideálně by v rámci jiné práce mohlo být prováděno prohledávání výše uvedených parametrů provedeno sofistikovanější metodou jako např. metodou *Grid Search*. To by však mohlo zabrat i několik dnů, či měsíců výpočtů s nejistými výsledky, obzvláště vzhledem k tomu, že koherence pořád ještě nezaručuje dobré výsledky při použití LDA na konečném úkolu v doporučujícím systému.

Obrázek 50: Hodnoty koherence pro model založený na data setu iDNES.cz

Z hodnot koherence pro model LDA založeném na data setů *Idnes*.cz lze vidět, že hodnoty koherence se výrazně liší v případě vyšších počtů témat. Koherence je výrazně vyšší pro model, u kterého se tvoří téma pro každé slovo než u modelu, kde se hodnoty pro každé slovo netvoří, ale u nižších počtu témat je naopak.

|  |  |  |
| --- | --- | --- |
| **num\_topics** | **Coherence (u\_mass)** | |
| **Number of topics** | **Model 1 (Per Word Topics = True)** | **Model 2 (Per Word Topics = False)** |
| 2 | -2,304 | -2,354 |
| 8 | -2,109 | -2,126 |
| 14 | -2,066 | -2,236 |
| 20 | -2,211 | -2,223 |
| 40 | -2,319 | -2,272 |
| 60 | -2,566 | -2,346 |
| 80 | -2,546 | -2,606 |
| 100 | -2,627 | -2,827 |
| 200 | -3,857 | -3,853 |

Tabulka 15: Hodnoty koherence pro model Wikipedie

Obrázek 51: Hodnoty koherence pro LDA model založen na datasetu Wikipedie

Tabulka ukazuje příklady kompletních výpisů některých ze shluků a jejich expertní pojmenování na základě uvedených převažujících slov a dvojic slov (bigramů). Některé shluky však není možné snadno pojmenovat díky postrádání jasného pojítka mezi slovy:

|  |  |  |
| --- | --- | --- |
| **Číslo téma** | **Slova a jejich váha v rámci téma** | **Vlastní pojmenování téma** |
| 1 | 0.079\*"tým" + 0.058\*"výkon" + 0.049\*"sezona" + 0.022\*"tlak" + 0.021\*"vítězství" + 0.021\*"soupeř" + 0.019\*"délka" + 0.017\*"tomáš" + 0.014\*"obrana" + 0.014\*"útok" | Sport |
| 2 | 0.083\*"pacient" + 0.078\*"nemocnice" + 0.060\*"nemoc" + 0.055\*"lékař" + 0.032\*"protilátka" + 0.029\*"buňka" + 0.029\*"léčba" + 0.024\*"onemocnění" + 0.019\*"účinnost" + 0.017\*"používaný" | Zdravotnictví / Nemoci |
| 3 | 0.035\*"automatický" + 0.024\*"hluboko" + 0.022\*"bodový" + 0.010\*"stůl" + 0.003\*"ráj" + 0.000\*"sankce" + 0.000\*"silniční\_provoz" + 0.000\*"nestihnout" + 0.000\*"akumulátor" + 0.000\*"dopravní\_přestupek" | Dopravní pokuty |
| 4 | 0.051\*"dítě" + 0.028\*"rodina" + 0.028\*"generace" + 0.020\*"doma" + 0.017\*"start" + 0.017\*"technologie" + 0.017\*"hra" + 0.016\*"škola" + 0.015\*"skvělý" + 0.013\*"rodič" | Moderní školství |
| 9 | 0.055\*"zima" + 0.040\*"teplota" + 0.034\*"tuna" + 0.028\*"sníh" + 0.025\*"vrchol" + 0.023\*"mráz" + 0.023\*"rekord" + 0.021\*"stanic" + 0.019\*"suchý" + 0.018\*"minimum" | Počasí / Extrémy v počasí |
| 11 | 0.040\*"výroba" + 0.035\*"nabídka" + 0.026\*"jenže" + 0.025\*"představit" + 0.019\*"vybrat" + 0.014\*"továrna" + 0.013\*"německý" + 0.012\*"obrovský" + 0.012\*"přední" + 0.012\*"boj" | Výroba / Průmysl |
| 14 | 0.045\*"rok" + 0.027\*"let" + 0.015\*"český" + 0.011\*"společnost" + 0.010\*"cena" + 0.009\*"měsíc" + 0.008\*"verze" + 0.008\*"procent" + 0.007\*"svět" + 0.007\*"veliký" | České firmy |

Tabulka 16: Převažující slova v rámci témat identifikovaných metodou LDA

* 1. Uživatelské testování

Uživatelské testování probíhalo při zúčastnění studentů katedry informatiky a počítačů, kteří byli instruování následujícím kroky:

1. Přechod do sekce uživatelské sekce (klik na uživ. jméno vpravo nahoře).
2. Zadání oblíbených kategorií. Pro jednotnost 5 kategorií.
3. Zadání klíčových slov, která by měla vyjadřovat, co uživatele zajímá. Pro jednotnost 5 kategorií.
4. Na základě předchozích akcí je uživateli vygenerováno 40 článků k hodnocení. Obsahuje 10 článků zvolených na základě zadaných klíčových slov, 10 na základě kategorií. 20 náhodně vybraných.
5. Uživatel přejde na odkaz, který je mu nabídnut. Odkaz vede na speciální testovací sekci (běžně nepřístupné, ale vychází z rozvržení přední strany).
6. Uživatel hodnotí 40 zobrazených článků. Každý článek hvězdami i palci. Pokud se mezi 40 články již u některých nemůžete rozhodnout, obnovte stránku a přibydou vám některé další.
7. Uživatel klikne na články, kterého nejvíce zajímají (měly by to být ty články, u kterých byste chtěli přečíst celý text článku). Počet je na uživateli, ale minimum je 5 kliků na článek.
8. Kontrola tlačítkem úplně dole.
9. (Přidání chybějících hodnocení nebo zhlédnutí).
10. Měl by se zobrazit zelený rámec, můžete přejít zpět do uživatelské sekce klikem na tlačítko profilu budete přesměrování do sekce Playground 🡪 Uživatelské testování (<https://moje-clanky.herokuapp.com/administration-section/user-playground>).
11. Posouzení relevance algoritmů založených na uživatelích: uživateli se nabídnou testovací výsledky založené na metodách založených na uživateli.
12. SVD
13. Na základě klíčových slov.
14. Na základě kategorií.
15. Posouzení relevance algoritmů založených na obsahu.
16. Náhodné vygenerování koncovky článku.
17. Uživatel posoudí relevanci 20 článku vzhledem k náhodně vygenerovanému článku.
18. TF-IDF
19. TF-IDF Full Text
20. Word2Vec Eval #3
21. Doc2Vec
22. LDA
23. Uživatel přijde později / na druhý den a ohodnotí:
24. Hybridní.
25. Hybridní, fuzzy.
    * 1. Výsledky uživatelského testování a srovnání obsahových metod

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **variant** | **AP** | **balanced** | **DCG** | **DCG\_AT\_5** | **F1** | **NDCG** | **NDCG\_AT\_5** |
| doc2vec | 0.51 | 0.5 | 1.92 | 0.78 | 0.14 | 0.58 | 0.3 |
| tfidf | 0.47 | 0.44 | 2.38 | 0.97 | 0.21 | 0.59 | 0.34 |
| tfidf\_full\_text | 0.52 | 0.5 | 1.84 | 0.75 | 0.12 | 0.58 | 0.28 |
| w2v\_eval | 0.76 | 0.5 | 2.84 | 1.15 | 0.28 | 0.65 | 0.42 |
| lda\_full | 0.32 | 0.5 | 1.12 | 0.46 | 0.06 | 0.48 | 0.21 |

Tabulka 17: Výsledky uživatelského testování a obsahové metody+

První část finálního uživatelského srovnání se týkala posouzení 20 doporučených článků k jednomu, náhodně vygenerovanému článku. Zajímavým zjištěním z uživatelského testování je preference Word2Vec modelu založeném na vlastní metodě evaluace hyper parametrů, která dosáhla lepších výsledků než metoda Doc2Vec, která byla využita jako lepší nástupce Word2Vec, avšak srovnávaný model Doc2Vec v rámci této práce byl natrénován pouze na heuristicky určených parametrech. Velmi jednoduchá představená metoda využívající datové sady otázek-odpovědí a analogických slov tedy dokázala podle uživatelů přinést daleko lepší výsledky podobnosti 1 článku k ostatním než na stejných datech natrénovaný Doc2Vec model, který tuto metody nevyužíval.

Nižší výsledky mohou být způsobeny také způsobem volby relevance – články byly v původním stavu brány jako nerelevantní a uživatelé je teprve museli zaškrtnout jako relevantní. Museli si být tedy jisti, že je článek relevantní. To mohlo způsobit přísnější volbu oproti testování, kdy jsou uživatelé např. postaveni před tlačítka palec nahoru a dolů.

* + 1. Výsledky uživatelského testování a srovnání uživatelských metod

Zatímco v první části posouzení relevance doporučených článků byli uživatelé představeni před spíše objektivní posouzení podobnosti ke konkrétnímu článku, v druhé fázi testování se jednalo výhradně o subjektivní volbu. Ač je SVD metodou velmi využívanou a dobře ověřenou (viz sekce SVD), ukázal se u ní problém studeného startu při malém množství uživatelských hodnocení. Ač jsme se pokusili tento problém řešit pomocí vygenerovaných hodnocení na základě kategorií článků, ukázalo se, že SVD pouze částečně překonalo primitivní metodu založenou na nejlépe hodnocených článcích z daných kategorií. Hybridní metody, ač z části založeny na SVD poté prokázaly lepší výsledky i přes problém studeného startu. Při začátcích na malém množství uživatelů se jako překvapivě dobrá projevila metoda srovnávání zadaných klíčových slov uživatelem a článků pomocí metody TF-IDF. Pozitivní hodnocení této metody lze interpretovat i tak, že zatímco u metody SVD uživatelé nevidí takovou spojitost mezi jejich hodnocením a ostatními uživateli, u klíčových slov ví okamžitě na základě čeho, jim byly články doporučeny.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **model\_variant** | **AP** | **balanced** | **DCG** | **DCG\_AT\_5** | **F1-SCORE** | **NDCG** | **NDCG\_AT\_5** |
| categories | 0.49 | 0.67 | 3.46 | 1.45 | 0.4 | 0.72 | 0.5 |
| hybrid | 0.62 | 0.5 | 3.17 | 1.33 | 0.31 | 0.72 | 0.46 |
| hybrid\_fuzzy | 0.56 | 0.5 | 3.7 | 1.55 | 0.39 | 0.77 | 0.52 |
| keywords | 0.67 | 0.56 | 4.15 | 1.74 | 0.49 | 0.79 | 0.6 |
| svd | 0.55 | 0.5 | 3.64 | 1.52 | 0.36 | 0.78 | 0.52 |

Tabulka 18: Uživatelské testování a uživatelské metody

* 1. Evaluace klasifikátoru

Po trénování modelů klasifikátorů bylo dosaženo například těchto výsledků. Modely jsou nicméně průběžně znovu trénovány, aby doporučení reagovala na nejnovější uživatelská hodnocení.

Tato metoda byla vyvíjena pouze jako experimentální, nebyla tedy součástí uživatelského testování předepsaným scénářem, a to především z důvodu neuspokojivých předběžných výsledků při použití článků v českém jazyce a zvýšení nároků na zpracování dat a robustnost systému a možných technických problémů, které mohou při použití této metody vzniknout (každý uživatel např. potřebuje svůj vlastní natrénovaný model). Výsledky níže mohou být tedy zavádějící. Jsou založeny pouze na hodnocení náhodně příchozích uživateli, tvůrcem systému a vedoucím práce.

V experimentálním módu vykazovala metoda Random Forrest tyto výsledky:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **accuracy\_**  **score** | **precision\_**  **score\_**  **weighted** | **balanced\_**  **accuracy\_**  **score** |
| **Mean** | 0.532 | 0.457 | 0.463 |
| **Std. Deviation** | 0.279 | 0.339 | 0.253 |
| **Minimum** | 0.000 | 0.000 | 0.000 |
| **Maximum** | 1.000 | 1.000 | 1.000 |

Tabulka 19: Evaluace filtrování zpráv při experimentování s klasifikační metodou Random Forrest

Experimentální model založený na metodě SVC vykazoval tyto výsledky:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **accuracy\_**  **score** | **precision\_**  **score\_**  **weighted** | **balanced\_**  **accuracy\_**  **score** |
| **Mean** | 0.545 | 0.400 | 0.461 |
| **Std. Deviation** | 0.323 | 0.339 | 0.300 |
| **Minimum** | 0.000 | 0.000 | 0.000 |
| **Maximum** | 1.000 | 1.000 | 1.000 |

Tabulka 20: Evaluace filtrování zpráv při experimentování s klasifikační metodou SVC

Nízké skóre přesnosti tedy ukazovalo, že implementace těchto metod je složitější. Pokud by se nicméně podařilo tento druh doporučení dotáhnout do konce, nabízí pak elegantní způsob pro filtrování zpráv na základě voleb relevance uživatele, touto metodou by tedy bylo zajímavé se zabývat v rámci jiné práce.

* 1. Možné problémy doporučujících metod

Ač tato poznámka sahá daleko za rámec této práce, na výše uvedeném jednoduchém příkladu článku a jeho podobnosti v podkapitole 4.3.1 je možné vidět, jak nesprávné využití inteligentních metod a algoritmů může vést k nežádoucím jevům. Z hlediska ekonomického a marketingového může dojít ke ztrátě zájmu zákazníka na základě mimořádně nežádoucích, irelevantních doporučení. Mohli bychom ale spekulovat i o psychologickém a sociologickém rozměru doporučujících systémů v případě, že uživatel naopak bude pokračovat ve čtení. Představme si, že si uživatel chce přečíst články týkající se Krkonoš, následně narazí na článek pojednávající o krkonošském nosiči a jeho úmrtí. Z příjemného počtení se najednou stává brouzdání černou kronikou a jsou mu servírovány články o tragických nehodách a úmrtích. Na psychicky slabší osoby mohou v dlouhodobém horizontu podobné zážitky nepochybně působit negativně. Metody založené na obsahy přitom lze vnímat jako ty spíše více transparentní směrem k uživateli. Ještě negativnější vliv mohou mít kolaborativní metody shlukující podobné uživatele, které mohou zapříčinit uživatelovo uzavírání se do informačních bublin [136].

Efekt doporučujících systémů jako takových na lidskou psychiku však není zatím tolik prozkoumán, protože se většinou zkoumá již výsledná platforma (např. sociální sítě), jejíž jsou doporučující systémy součástí. U nich se však v posledních letech zjistily negativní vlivy na lidskou psychiku, a to zejména u dospívajících [137] [138].

Tyto problémy, které šlo pozorovat již z prvních výsledků tohoto doporučujícího systému pro zobrazování článků tedy poukazují na to, proč je nutné v rychle vyvíjejícím se technologickém světě k doporučujícím systémům přistupovat zodpovědně a chápat i jejich širší kontext a dopad jak na jednotlivce, tak na společnost. I na základě těchto poznatků byly navrhnuty další níže zmíněné metody spolu s vlastním hybridním přístupem.

1. ZHODNOCENÍ

V této práci byla představena kombinace přístupů pro problematiku doporučujících systémů v oblasti zpravodajských článků. Při tvorbě práce se ukázaly některé podstatné rozdíly mezi náročností implementace kolaborativních metoda a metoda založených na obsahu. Zatímco implementace kolaborativní metody SVD byla přímočará a vykazovala již v první fázi testování dobré výsledky, metody založené na obsahu se ukázaly jako více problematické a jejich implementace zabrala mnohonásobně více času, stejně jako jejich ověření. Důvody náročnosti spočívaly především v náchylnosti modelů na správné vstupy a zvolené parametry strojového učení pro trénování modelů a rovněž v jejich náročnosti na výkon. Náročnost metod založených na obsahu také spočívá ve složitosti zpracování přirozeného jazyka a v nutnosti přípravy textu před vložením do modelů strojového učení.

Na vlastní datové sadě sestavené z článků zpravodajského serveru *Idnes*.cz vykazovala implementačně jednodušší a zastaralejší metoda TF‑IDF založená na „tradičních“ metodách lineární algebra a statistiky, překvapivě dobré výsledky ve srovnání se sofistikovanějším metodám strojového učení využívajícím neuronovou síť (Word2Vec, Doc2Vec), resp. pokročilejším statistickém modelu LDA. Poměr výkonu a náročnosti implementace (a optimalizace) byl v případě datové sady *Idnes*.cz tedy na straně TF‑IDF a pro menší projekty se tento algoritmus jeví jako vhodný. Po hledání vhodných parametrů Word2Vec na základě testů párů slov a analogií však Word2Vec prokázalo lepší výsledky než TF-IDF. Hybridní algoritmus splnil očekávání v tom, že kombinace různých přístupů by měla vykazovat vyšší relevanci výsledků než metody založené pouze na obsahu nebo uživatelích.

* 1. Otevřené záležitosti

Tvorba shluků v rámci LDA modelu vykazovala poměrně smysluplné výsledky, kdy většina shluků slov byla smysluplná a shluky šlo pojmenovat společným tématem. Problematické se ovšem ukázalo použití pro výběr nejpodobnějších článků k jednomu konkrétnímu. Toto zklamání z výsledků metody LDA modelu založeném na článcích z *Idnes*.cz přiměly k využití textů z *Wikipedie*, ty pomohly relevanci této metody vylepšit, ale nikoliv tolik, jak od ní bylo slibováno.

U metod klasifikátorů by se poté dalo provést i pokročilejší ladění parametrů modelů. Vzhledem k tomu, že podstatná část času však byla věnována doporučením založeným na obsahu, které byly v případě novinových článků identifikovány jako nejdůležitější, neměl tento typ doporučení takovou prioritu a jedná se stále o otevřenou, spíše experimentální záležitost.

Zajímavé by bylo ještě blíže prozkoumat jednoduchou metodu srovnávání analogií slov, která se ukázala jako výhodná pro stanovení parametrů modelu Word2Vec. To je přitom někdy mylně představováno jako čistě heuristická úloha a jako výpočetně a zdrojově náročná úloha.

Pro kompletní srovnání by bylo podrobit uživatelskému testování také metodu založenou na klasifikátoru relevance (na základě metod Random Forrest a SVC).

Pro lepší ověření by bylo vhodné otestovat systém na větším množství uživatelů.

ZÁVĚR

Cílem této diplomové práce bylo navrhnout a implementovat doporučující systém pro doporučování aktuálních článků. V rámci této práce byly popsány, naprogramovány, evaluovány a optimalizovány doporučující metody. Jejich využití bylo implementováno v rámci webové aplikace Moje články a některé poznatky z této práce byly nebo budou využity k napsání samostatných článků.

Byl prozkoumán současný stav v rámci významných aplikací pracujících s novinovými články. Jako další zdroje informací posloužily jiné akademické práce nebo knihy zabývající se touto tématikou. Na základě těchto zdrojů byl navrhnut a implementován vlastní doporučující systém.

Vytvořený doporučující systém založen na několika přístupech v oblasti doporučujících systémů může uživateli nabídnout jemu přizpůsobený obsah na základě jeho akcí a preferencí, stejně tak na základě akcí ostatních uživatelů. Při proklicích na články se zobrazují články podobné danému článku. Uživatel obdrží články na základě jeho hodnocení a na základě hodnocení jinými uživateli. Uživatel si může zvolit klíčová slova témat, které jej zajímají a na základě této akce jsou mu zobrazeny další doporučené články. V rámci hybridního algoritmu se pak kombinují výše zmíněné přístupu a v potaz jsou také brány uživatelovy zvolené kategorie a rovněž historie zhlédnutí článků uživatelem.

Při menších úpravách některých částí je možné tento doporučující systém využít i pro jinou doménu než pro zobrazování článků, avšak v rámci této práce byl kladen zvláštní důraz kromě na uživatelích založených metodách také na metody založené na obsahu, tzn. využívající pouze přirozený jazyk.

RESUMÉ

Výsledkem této práce je webová aplikace *Moje články*, která uživatelům doporučuje články na základě jejich preferencí, činností a obsahu, který je zajímá.

V částech věnovaných optimalizaci a vyhodnocování byly nabídnuty přístupy uznávané v rámci oborů, jako je datová analytika a strojové učení, ale byly představeny i některé méně používané metody a modifikace metod vhodné pro účely této práce.

V rámci webových výukových materiálů je tato důležitá část optimalizace a vyhodnocování často opomíjena a zanedbávána a některé metody jsou náročné na zdroje. Tato práce by proto měla poskytnout jasnější pohled na tuto problematiku a ukázala výhodnost využití jednoduchých testů analogií pro zvolení hyper parametrů, které je obtížné zvolit u Word2Vec metody, které je založena na učení bez učitele nebo pod vlastním dohledem. Hybridní algoritmus představený v rámci této práce je inspirován jinými hybridními algoritmy, ale je upraven a rozšířen pro oblast doporučování novinových článků.

V neposlední řadě může tato práce sloužit jako vhled do doporučujících systémů pro studenty napříč obory (např. včetně humanitních), které by zajímaly detaily implementace takových systémů v této citlivé oblasti, která má a v budoucnu pravděpodobně bude mít velký vliv na vyhledávání informací v rámci společnosti.

BIBLIOGRAFIE

|  |  |
| --- | --- |
| [1] | CHRISTIAN, Jon. We still don’t know how Google News works. In: *The Outline* [online]. [cit. 2017-11-22]. Dostupné z: https://theoutline.com/post/2512/we-still-don-t-know-how-google-news-works |
| [2] | A look at how news at Google works. In: *Google: The Keyword* [online]. [cit. 2021-04-06]. Dostupné z: https://blog.google/products/news/look-how-news-google-works/ |
| [3] | SULLIVAN, Danny. Debunking The Italian Newspapers’ Antitrust Allegations Against Google. In: *Search Engine Land* [online]. Third Door Media, Inc., 2021 [cit. 2021-05-23]. Dostupné z: https://searchengineland.com/deunking-the-italian-newspapers-antitrust-allegations-against-google-24698 |
| [4] | The journalists making the calls behind the Apple News app. In: *The Irish Times* [online]. Dublin, 2021 [cit. 2021-05-06]. Dostupné z: https://www.irishtimes.com/business/technology/the-journalists-making-the-calls-behind-the-apple-news-app-1.3680529 |
| [5] | *Flipboard* [online]. 2021 [cit. 2020-04-20]. Dostupné z: https://flipboard.com/ |
| [6] | *Feedly* [online]. [cit. 2021-04-06]. Dostupné z: https://feedly.com/ |
| [7] | *Pocket* [online]. Read It Later, Inc., 2021 [cit. 2021-04-06]. Dostupné z: https://getpocket.com/my-list |
| [8] | Read It Later Raises $2.5 Million, Wants To Become The Dropbox Of Content. In: *Tech Crunch* [online]. TechCrunch, 2021 [cit. 2021-05-06]. Dostupné z: https://techcrunch.com/ |
| [9] | Očekávaný projekt českého Livesportu je tady. FlashNews jsou centrem veškerého dění, které vás zajímá. In: *Czech Crunch* [online]. CzechCrunch s.r.o., 2021 [cit. 2021-10-29]. Dostupné z: https://www.czechcrunch.cz/ |
| [10] | FORTUNA, Blaž, Carolina FORTUNA a Dunja MLADENIć. Real-Time News Recommender System. *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 583-586. Lecture Notes in Computer Science. ISBN 978-3-642-15938-1. Dostupné z: doi:10.1007/978-3-642-15939-8\_38 |
| [11] | BAGHER, Rahimpour, Hamid HASSANPOUR a Hoda MASHAYEKHI. User trends modeling for a content-based recommender system. *Expert Systems with Applications*. 2017, **87**, 209-219. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2017.06.020 |
| [12] | BLEI, David. COS 597C: Bayesian nonparametrics. In: *Department of Computer Science: Princenton University* [online]. The Trustees of Princeton University., 2022 [cit. 2022-04-01]. Dostupné z: https://www.cs.princeton.edu/courses/archive/fall07/cos597C/scribe/20070921.pdf |
| [13] | HARPER, F. a Joseph KONSTAN. The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems*. 2016, **5**(4), 1-19. ISSN 2160-6455. Dostupné z: doi:10.1145/2827872 |
| [14] | JAKIMOV, Matěj. *Multimodální doporučovací systém*. Brno, 2016. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Doc. RNDr. Vlastislav Dohnal, Ph.D. |
| [15] | LINDEN, G., B. SMITH a J. YORK. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*. 2003, **7**(1), 76-80. ISSN 1089-7801. Dostupné z: doi:10.1109/MIC.2003.1167344 |
| [16] | FALK, Kim. *Practical recommender systems*. 2. Shelter Island, NY: Manning, 2019. ISBN 978-161-7292-705. |
| [17] | MARMANIS, Haralambos a Dmitry BABENKO. *Algorithms of the intelligent web*. Greenwich, CT: Manning, 2009. ISBN 19-339-8866-5. |
| [18] | *Unauthorized Use of Copyrighted Works in Digital Form: Report on Piloting in Finland*. Kulttuuripoliittisen tutkimuksen edistämissäätiö, 2016. ISBN 978-952-7200-08-7. |
| [19] | GILES, John. The rights to a database | Who owns the copyright in data?. In: *Michalsons* [online]. Kapské město: Michalsons Lexing Inc, c2002-2021 [cit. 2020-11-23]. Dostupné z: https://www.michalsons.com/blog/the-rights-to-a-database/2937 |
| [20] | *The Netflix Tech Blog* [online]. [cit. 2022-04-14]. Dostupné z: https://netflixtechblog.com/ |
| [21] | AMATRIAIN, Xavier a Justin BASILICO. System Architectures for Personalization and Recommendation. In: *The Netflix Tech Blog* [online]. [cit. 2022-04-14]. Dostupné z: https://netflixtechblog.com/system-architectures-for-personalization-and-recommendation-e081aa94b5d8 |
| [22] | WONG, Jeffrey a Colin MCFARLAND. Design Principles for Mathematical Engineering in Experimentation Platform at Netflix. In: *The Netflix Tech Blog* [online]. [cit. 2022-04-14]. Dostupné z: https://netflixtechblog.com/design-principles-for-mathematical-engineering-in-experimentation-platform-15b3ea143b1f |
| [23] | Dynomite with Redis on AWS — Benchmarks. In: *The Netflix Tech Blog* [online]. [cit. 2022-03-12]. Dostupné z: https://netflixtechblog.com/dynomite-with-redis-on-aws-benchmarks-5c942fc7ca38 |
| [24] | How Long Before You Run Out of Cash? Your Capital Runway is the Answer. In: *Nordea* [online]. Nordea Bank, 2022 [cit. 2022-06-14]. Dostupné z: https://www.nordea.com/en/news-topics/corporate-insights/business/how-long-before-you-run-out-of-cash-your-capital-runway-is-the-answer/# |
| [25] | ADOMAVICIUS, G. a A. TUZHILIN. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*. 2005, **17**(6), 734-749. ISSN 1041-4347. Dostupné z: doi:10.1109/TKDE.2005.99 |
| [26] | GOMEZ-URIBE, Carlos a Neil HUNT. The Netflix Recommender System. *ACM Transactions on Management Information Systems*. 2016, **6**(4), 1-19. ISSN 2158-656X. Dostupné z: doi:10.1145/2843948 |
| [27] | HOEKSTRA, Rinke. The Knowledge Reengineering Bottleneck. *Semant. Web*. NLD: IOS Press, 2010, **1**(12), 111–115. ISSN 1570-0844. |
| [28] | PROCHÁZKA, Jaroslav a Jaroslav ŽÁČEK. INFORMAČNÍ SYSTÉMY. *Elektronick`y učební text. Ostravská univerzita*. 2012. |
| [29] | Native methods and the Java Native Interface. In: *IBM Documentation* [online]. c2002–2013 [cit. 2022-04-14]. Dostupné z: https://www.ibm.com/docs/en/i/7.2?topic=languages-native-methods-java-native-interface |
| [30] | KERVIZIC, Julien. Become a Pro at Pandas, Python’s Data Manipulation Library. In: *KD Nuggets* [online]. 2022 [cit. 2022-06-14]. Dostupné z: https://www.kdnuggets.com/2019/06/pro-pandas-python-library.html |
| [31] | OGENLUKE, Samuel. Why Is Python Popular for Data Science?. In: *MakeUseOf: Technology, Simplified* [online]. 2022 [cit. 2021-03-03]. Dostupné z: https://www.makeuseof.com/why-is-python-popular-for-data-science/ |
| [32] | Pandas on Ray. In: *RiseLab: UCBerkeley* [online]. The UCBerkeley [cit. 2022-06-14]. Dostupné z: https://rise.cs.berkeley.edu/blog/pandas-on-ray/ |
| [33] | Python Kafka Client Benchmarking. In: *Activision: Game Science* [online]. Activision Game Science, 2017 [cit. 2022-06-14]. Dostupné z: https://activisiongamescience.github.io/2016/06/15/Kafka-Client-Benchmarking/ |
| [34] | Kafka vs RabbitMQ. In: *Project Pro* [online]. Iconiq Inc., 2022 [cit. 2022-06-14]. Dostupné z: https://www.projectpro.io/article/kafka-vs-rabbitmq/451 |
| [35] | ROBINSON, Daniel. A comprehensive guide to HPC in the data center: Compare Hadoop vs. Spark vs. Kafka for your big data strategy. In: *TechTarget: Search Data Center* [online]. TechTarget, c2000–2022 [cit. 2022-06-14]. Dostupné z: https://www.techtarget.com/searchdatacenter/feature/Compare-Hadoop-vs-Spark-vs-Kafka-for-your-big-data-strategy |
| [36] | *Redis: The open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker.* [online]. [cit. 2022-03-12]. Dostupné z: https://redis.io/ |
| [37] | MAIER, David. Building a Recommendation Engine with Redis. In: *No SQL Geek: SQL Not Only* [online]. [cit. 2022-06-14]. Dostupné z: http://nosqlgeek.blogspot.com/2019/02/building-recommendation-engine-with.html |
| [38] | Applications Message Delivery Reliability with AMQP. In: *IBM Documentation* [online]. IBM Corporation, c1993–2022 [cit. 2022-03-04]. Dostupné z: https://www.ibm.com/docs/en/ibm-mq/9.1?topic=applications-message-delivery-reliability-amqp |
| [39] | AMQP 0-9-1 Model Explained. In: *RabbitMQ* [online]. c2007–2022 [cit. 2022-03-11]. Dostupné z: https://www.rabbitmq.com/tutorials/amqp-concepts.html |
| [40] | Possible future directions for data on the Web. In: *Https://www.w3.org* [online]. W3C, 2021 [cit. 2021-03-04]. Dostupné z: https://www.w3.org/blog/2017/06/possible-future-directions-for-data-on-the-web/ |
| [41] | GODSKIND, Ken. 5, 10, 15 seconds? How Long Will You Wait For a Web Page to Load?. In: *Smart Bear* [online]. 2022 [cit. 2022-02-04]. Dostupné z: https://smartbear.com/blog/5-10-15-seconds-how-long-will-you-wait-for-a-web-p/ |
| [42] | WALL, Matthew. How long will you wait for a shopping website to load?. In: *BBC* [online]. BBC [cit. 2022-06-14]. Dostupné z: https://www.bbc.com/news/business-37100091 |
| [43] | Largest newspaper websites losing users due to slow load times on mobile. In: *Device Atlas* [online]. DeviceAtlas Limited 2022, 2022 [cit. 2021-03-08]. Dostupné z: https://deviceatlas.com/blog/news-websites-load-times-on-mobile-statistics |
| [44] | DOMES, Martin. *Tvorba internetových stránek pomocí HTML, CSS a JavaScriptu*. Kralice na Hané: Computer Media, 2005. ISBN 80-866-8639-6. |
| [45] | MÜLLER, Patrik. *Návrh a implementace varovného systému proti přírodním katastrofám*. Ostrava, 2020. Bakalářská. Ostravská univerzita. |
| [46] | DEBARROS, Anthony. *Practical SQL: a beginner's guide to storytelling with data*. 2nd edition. San Francisco: No Starch Press, 2022. ISBN 978-171-8501-065. |
| [47] | PostgreSQL vs. MySQL: What’s the Difference?. In: *IBM: IBM Cloud* [online]. IBM Corporation [cit. 2022-04-11]. Dostupné z: https://www.ibm.com/cloud/blog/postgresql-vs-mysql-whats-the-difference |
| [48] | PostgreSQL: A powerful, open source object-relational database system. In: *Tech Stack Intelligence: https://stackshare.io/* [online]. StackShare, Inc., 2022 [cit. 2022-06-14]. Dostupné z: https://stackshare.io/postgresql |
| [49] | Beautiful Soup Documentation. In: *Crummy: The Site: News You Can Bruise* [online]. [cit. 2020-09-16]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ |
| [50] | *File archive of Umeå University* [online]. In: . Umeå University: Academic Computer Club [cit. 2022-02-15]. Dostupné z: https://ftp.acc.umu.se/mirror/wikimedia.org/ |
| [51] | COVINGTON, Paul, Jay ADAMS a Emre SARGIN. *Deep Neural Networks for YouTube Recommendations*. 2016. |
| [52] | GROWER, Stephen. Netflix Prize and SVD. In: *University of Puget Sound* [online]. 2014 [cit. 2022-06-14]. Dostupné z: http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf |
| [53] | *THE PERSONAL NEWS CYCLE* [online]. Media Insight Project, AP, 2013 [cit. 2022-12-04]. Dostupné z: https://www.americanpressinstitute.org/wp-content/uploads/2014/03/The\_Media\_Insight\_Project\_The\_Personal\_News\_Cycle\_Final.pdf |
| [54] | *The Reuters institute digital news report 2013*. Oxford, England: Reuters Institute for the Study of Journalism, 2013, 108 s. ISBN 9781907384103. |
| [55] | LANGUAGES, Oxford. *Oxford Dictionary of English*. 3rd Edition. Oxford: Oxford University Press, 2010. ISBN 978-0199571123. |
| [56] | HU, Jian, Lujun FANG, Yang CAO, Hua-Jun ZENG, Hua LI, Qiang YANG a Zheng CHEN. Enhancing text clustering by leveraging Wikipedia semantics. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*. New York, New York, USA: ACM Press, 2008, 179-. ISBN 9781605581644. Dostupné z: doi:10.1145/1390334.1390367 |
| [57] | WANG, Pu, Jian HU, Hua-Jun ZENG a Zheng CHEN. Using Wikipedia knowledge to improve text classification. *Knowledge and Information Systems*. 2009, **19**(3), 265-281. ISSN 0219-1377. Dostupné z: doi:10.1007/s10115-008-0152-4 |
| [58] | Experiments on the English Wikipedia. In: *Gensim: Topic Modeling for Humans* [online]. [cit. 2022-06-16]. Dostupné z: https://radimrehurek.com/gensim/wiki.html |
| [59] | WOłK, Krzysztof. *Machine Learning in Translation Corpora Processing*. 1st Edition. CRC Press, 2019. ISBN 978-0367186739. |
| [60] | How to download Wikipedia. In: *Data Science, Data Analytics and Machine Learning Consulting in Koblenz Germany* [online]. [cit. 2022-04-16]. Dostupné z: https://www.rene-pickhardt.de/index.html%3Fp=249.html |
| [61] | Largest encyclopedia online. In: *Guinness World Records* [online]. [cit. 2022-04-03]. Dostupné z: https://www.guinnessworldrecords.com/world-records/85651-largest-encyclopedia-online |
| [62] | DAVIS, J, P COMPAGNONI a P NANNINGA. Roles for knowledge-based systems in environmental planning. *Environment and Planning B: Planning and Design*. 1987, **14**(3), 239-254. ISSN 0265-8135. Dostupné z: doi:10.1068/b140239 |
| [63] | DAVIS, Randall. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill College, 1981, 490 s. ISBN 978-0070155572. |
| [64] | GABRILOVICH, Evgeniy a Shaul MARKOVITCH. *Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge: Enhancing Text Categorization with Encyclopedic Knowledge*. AAAI Press, 2006, 1301–1306 s. |
| [65] | MCCARTHY, J. *Programs with Common Sense*. 1958, 77-84 s. |
| [66] | KUHN, Max a Kjell JOHNSON. *Applied Predictive Modeling*. In: . s. 27. Dostupné z: doi:10.1007/978-1-4614-6849-3 |
| [67] | Stopwords Czech (CS). In: *GitHub* [online]. [cit. 2021-03-05]. Dostupné z: https://github.com/stopwords-iso/stopwords-cs |
| [68] | MIHALCEA, Rada a Paul TARAU. *TextRank: Bringing Order into Text: Bringing Order into Text*. Association for Computational Linguistics, 2004, 404-411 s. Dostupné také z: https://aclanthology.org/W04-3252 |
| [69] | GOMES, Luís. A Stemmer for Czech. In: *Luís Gomes' page* [online]. [cit. 2022-06-15]. Dostupné z: https://research.variancia.com/czech\_stemmer/ |
| [70] | KETTUNEN, Kimmo, Markus SADENIEMI, Tiina LINDH-KNUUTILA a Timo HONKELA. Analysis of EU Languages Through Text Compression. *Advances in Natural Language Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 99-109. Lecture Notes in Computer Science. ISBN 978-3-540-37334-6. Dostupné z: doi:10.1007/11816508\_12 |
| [71] | *Free natural language morphology* [online]. In: . Faculty of Informatics, Masaryk University: Natural Language Processing Centre [cit. 2021-05-29]. Dostupné z: https://nlp.fi.muni.cz/polish-morphology-analyser/ |
| [72] | GEERAERTS, Dirk. *Teorie lexikální sémantiky*. Praha: Univerzita Karlova, nakladatelství Karolinum, 2019. Lingvistika (Karolinum). ISBN 978-80-246-4194-2. |
| [73] | *Metody měření podobnosti jazyků*. Liberec, 2014. Diplomová práce. Technická univerzita v Liberci. Vedoucí práce Prof. Ing. Jan Nouza, CSc. |
| [74] | FENG, Yue, Ebrahim BAGHERI, Faezeh ENSAN a Jelena JOVANOVIC. The state of the art in semantic relatedness: a framework for comparison. *The Knowledge Engineering Review*. 2017, **32**. ISSN 0269-8889. Dostupné z: doi:10.1017/S0269888917000029 |
| [75] | BALLATORE, Andrea, Michela BERTOLOTTO a David WILSON. An evaluative baseline for geo-semantic relatedness and similarity. *GeoInformatica* [online]. 2014, **18**(4), 747-767 [cit. 2022-06-15]. ISSN 1384-6175. Dostupné z: doi:10.1007/s10707-013-0197-8 |
| [76] | SALTON, G., A. WONG a C. YANG. A vector space model for automatic indexing. *Communications of the ACM*. 1975, **18**(11), 613-620. ISSN 0001-0782. Dostupné z: doi:10.1145/361219.361220 |
| [77] | CONNOR, Richard. A Tale of Four Metrics. *Similarity Search and Applications*. Cham: Springer International Publishing, 2016, 210-217. Lecture Notes in Computer Science. ISBN 978-3-319-46758-0. Dostupné z: doi:10.1007/978-3-319-46759-7\_16 |
| [78] | SIDOROV, Grigori, Alexander GELBUKH, Helena GÓMEZ-ADORNO a David PINTO. Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computación y Sistemas*. 2014, **18**(3). ISSN 1405-5546. Dostupné z: doi:10.13053/cys-18-3-2043 |
| [79] | ŘEHŮŘEK, Radim. Soft Cosine Measure. In: *Gensim: Topic Modeling for Humans* [online]. [cit. 2021-03-02]. Dostupné z: https://radimrehurek.com/gensim//auto\_examples/tutorials/run\_scm.html |
| [80] | CHARLET, Delphine a Geraldine DAMNATI. SimBow at SemEval-2017 Task 3: Soft-Cosine Semantic Similarity between Questions for Community Question Answering. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2017, 315-319. Dostupné z: doi:10.18653/v1/S17-2051 |
| [81] | Scikit-learn. In: *GitHub* [online]. [cit. 2022-04-15]. Dostupné z: https://github.com/scikit-learn/scikit-learn/blob/6c3e17989a7d80c34f124365f2c436a3fdcb1497/sklearn/base.py#L659-L690 |
| [82] | *Gensim: Topic Modeling for Humans* [online]. 2009 [cit. 2021-03-02]. Dostupné z: https://radimrehurek.com/gensim/ |
| [83] | ŘEHŮŘEK, Radim a Petr SOJKA. *Software Framework for Topic Modelling with Large Corpora*. ELRA, 2010, 45-50 s. |
| [84] | Gensim: Topic Modelling in Python. In: *GitHub* [online]. [cit. 2022-04-15]. Dostupné z: https://github.com/b5510546671/gensim |
| [85] | LESKOVEC, Jurij, Anand RAJARAMAN a Jeffrey ULLMAN. *Mining of massive datasets*. Second edition. Cambridge: Cambridge University Press, 2014. ISBN 978-131-6147-313. |
| [86] | BIN, Li a Guoyong YUAN. Improvement of TF-IDF Algorithm Based on Hadoop Framework. *Proceedings of the 2nd International Conference on Computer Application and System Modeling*. Paris, France: Atlantis Press, 2012, -. ISBN 978-94-91216-00-8. Dostupné z: doi:10.2991/iccasm.2012.98 |
| [87] | GERLACH, Martin, Hanyu SHI a Luís AMARAL. A universal information theoretic approach to the identification of stopwords. *Nature Machine Intelligence*. 2019, **1**(12), 606-612. ISSN 2522-5839. Dostupné z: doi:10.1038/s42256-019-0112-6 |
| [88] | LUHN, H. P. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*. 1957, **1**(4), 309-317. ISSN 0018-8646. Dostupné z: doi:10.1147/rd.14.0309 |
| [89] | SPARCK JONES, KAREN. A Statistical Interpretation Of Term Specificity And Its Application In Retrieval. *Journal of Documentation*. 1972, **28**(1), 11-21. ISSN 0022-0418. Dostupné z: doi:10.9776/17210 |
| [90] | BEEL, Joeran, Corinna BREITINGER a Stefan LANGER. *Evaluating the CC-IDF citation-weighting scheme: How effectively can 'Inverse Document Frequency' (IDF) be applied to references?: How effectively can 'Inverse Document Frequency' (IDF) be applied to references?*. 2017. |
| [91] | MIKOLOV, Tomas, G.s CORRADO, Kai CHEN a Jeffrey DEAN. *Efficient Estimation of Word Representations in Vector Space*. 2013, 1 s. |
| [92] | MIKOLOV, Tomas, Ilya SUTSKEVER, Kai CHEN, Greg CORRADO a Jeffrey DEAN. *Distributed Representations of Words and Phrases and their Compositionality*. arXiv, 2013. Dostupné z: doi:10.48550/ARXIV.1310.4546 |
| [93] | *Computer information retrieval using latent semantic structure*. 1988. United States. US4839853A. Uděleno 1988-09-15. Zapsáno 1988-09-15. Scott C. Deerwester Susan T. Dumais George W. Furnas Richard A. Harshman Thomas K. Landauer Karen E. Lochbaum Lynn A. Streeter. |
| [94] | HashingVectorizer. In: *Scikit-learn* [online]. c2007-2022 [cit. 2022-03-05]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.HashingVectorizer.html |
| [95] | COMMONS, Wikimedia. *File:Skipgram.png --- Wikimedia Commons, the free media repository: Skipgram.png --- Wikimedia Commons, the free media repository*. 2020. Dostupné také z: https://commons.wikimedia.org/w/index.php?title=File:Skipgram.png&oldid=506212658 |
| [96] | COMMONS, Wikimedia. *File:CBOW.png --- Wikimedia Commons, the free media repository: CBOW.png --- Wikimedia Commons, the free media repository*. 2020. Dostupné také z: https://commons.wikimedia.org/w/index.php?title=File:CBOW.png&oldid=505345946 |
| [97] | CountVectorizer. In: *Scikit-learn* [online]. [cit. 2021-11-04]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.CountVectorizer.html#sklearn.feature\_extraction.text.CountVectorizer |
| [98] | JOULIN, Armand, Edouard GRAVE, Piotr BOJANOWSKI a Tomas MIKOLOV. *Bag of Tricks for Efficient Text Classification*. arXiv, 2016. Dostupné z: doi:10.48550/ARXIV.1607.01759 |
| [99] | BOJANOWSKI, Piotr, Edouard GRAVE, Armand JOULIN a Tomas MIKOLOV. FastText. In: *Meta Research: Giving people the power to build community through research and innovation* [online]. [cit. 2022-03-15]. Dostupné z: https://research.facebook.com/blog/2016/8/fasttext/ |
| [100] | Word vectors for 157 languages. In: *FastText* [online]. Facebook Inc., 2022 [cit. 2022-06-15]. Dostupné z: https://fasttext.cc/docs/en/crawl-vectors.html |
| [101] | FastText model. In: *Gensim: Topic Modeling for Humans* [online]. 2009 [cit. 2022-02-18]. Dostupné z: https://radimrehurek.com/gensim/models/fasttext.html |
| [102] | ŘEHŮŘEK, Radim. Doc2vec paragraph embeddings. In: *Gensim* [online]. 2009 [cit. 2021-04-16]. Dostupné z: https://radimrehurek.com/gensim/models/doc2vec.html#module-gensim.models.doc2vec |
| [103] | FAN, Shuzhan. Understanding Word2Vec and Doc2Vec. In: *Shuzhan Fan: Sharing is caring* [online]. [cit. 2021-01-16]. Dostupné z: https://shuzhanfan.github.io/2018/08/understanding-word2vec-and-doc2vec/ |
| [104] | BLEI, David, Andrew NG a Michael JORDAN. Latent Dirichlet Allocation. *J. Mach. Learn. Res*. JMLR.org, 2003, **3**(), 993–1022. ISSN 1532-4435. |
| [105] | NOVEMBRE, John. Pritchard, Stephens, and Donnelly on Population Structure. *Genetics*. 2016, **204**(2), 391-393. ISSN 1943-2631. Dostupné z: doi:10.1534/genetics.116.195164 |
| [106] | Simpful: A friendly python library for fuzzy logic reasoning. In: *GitHub* [online]. [cit. 2023-04-05]. Dostupné z: https://github.com/aresio/simpful |
| [107] | 6.3.1. Standardization, or mean removal and variance scaling. In: *Scikit-learn: Machine Learning in Python* [online]. [cit. 2022-03-15]. Dostupné z: https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling |
| [108] | Lecture 5: Evaluation. In: *Department of Computer Science and Technology* [online]. Cambridge: University of Cambridge, 2022 [cit. 2022-04-15]. Dostupné z: https://www.cl.cam.ac.uk/teaching/1314/InfoRtrv/lecture5.pdf |
| [109] | ZEMAN, Daniel. Vyhodnocování úspěšnosti. In: *Institute of Formal and Applied Linguistics* [online]. Praha, 2022 [cit. 2022-06-15]. Dostupné z: https://ufal.mff.cuni.cz/~zeman/vyuka/podklady/pzpj03-vyhodnocovani.pdf |
| [110] | VERNEROVÁ, Anna. Strojové učení: klasifikace. In: *Institute of Formal and Applied Linguistics* [online]. Praha, 2022 [cit. 2022-06-15]. Dostupné z: https://ufal.mff.cuni.cz/~vernerova/2016/docs/prednaska\_06\_klasifikace.pro\_tisk.pdf |
| [111] | HORAK, Karel. MAP for Object Detection. In: *Machine Vision Group* [online]. Brno: Brno University of Technology, 2020 [cit. 2022-06-15]. Dostupné z: http://vision.uamt.feec.vutbr.cz/ROZ/lectures/MachineLearning\_mAP.pdf |
| [112] | BURGES, Chris, Tal SHAKED, Erin RENSHAW, Ari LAZIER, Matt DEEDS, Nicole HAMILTON a Greg HULLENDER. Learning to rank using gradient descent. *Proceedings of the 22nd international conference on Machine learning - ICML '05*. New York, New York, USA: ACM Press, 2005, 89-96. ISBN 1595931805. Dostupné z: doi:10.1145/1102351.1102363 |
| [113] | CROFT, Bruce, Trevor STROHMAN a Donald METZLER. *Search Engines: Information Retrieval in Practice*. 1st Edition. Pearson, 2009. ISBN 978-0136072249. |
| [114] | HUSTON, Samuel a W. CROFT. Evaluating verbose query processing techniques. *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*. New York, New York, USA: ACM Press, 2010, 291-. ISBN 9781450301534. Dostupné z: doi:10.1145/1835449.1835499 |
| [115] | Store and query word vectors. In: *Gensim: Topic Modeling for Humans* [online]. 2009 [cit. 2021-03-06]. Dostupné z: https://radimrehurek.com/gensim/models/keyedvectors.html#gensim.models.keyedvectors.KeyedVectors.evaluate\_word\_pairs |
| [116] | ALFONSECA, Enrique. WordSim353 - Similarity and Relatedness. In: *Enrique Alfonseca's home page* [online]. [cit. 2021-04-14]. Dostupné z: http://alfonseca.org/eng/research/wordsim353.html |
| [117] | CINKOVÁ, Silvie. WordSim353 for Czech: Lecture Notes in Artificial Intelligence. In: *Text, Speech and Dialogue, Proceedings of the 19th International*. Berlin-Heidelberg, Germany: Springer, 2016. |
| [118] | Placing Search in Context: The Concept Revisited: The Concept Revisited. *ACM Trans. Inf. Syst*. New York, NY, USA: ACM, 2002, **20**(1), 116-131. ISSN 1046-8188. Dostupné z: doi:10.1145/503104.503110 |
| [119] | AGIRRE, Eneko, Enrique ALFONSECA, Keith HALL, Jana KRAVALOVA, Marius PAşCA a Aitor SOROA. *A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches*. Association for Computational Linguistics, 2009, 19-27 s. Dostupné také z: http://dl.acm.org/citation.cfm?id=1620754.1620758 |
| [120] | BACCOURI, Nidhal. Deep-translator 1.8.3. In: *PyPI: Python Package Indexer* [online]. [cit. 2022-03-20]. Dostupné z: https://pypi.org/project/deep-translator/ |
| [121] | Models.keyedvectors: Store and query word vectors. In: ŘEHŮŘEK, Radim. *Gensim: Topic Modeling for Humans* [online]. 2009 [cit. 2022-03-02]. Dostupné z: https://radimrehurek.com/gensim\_3.8.3/models/keyedvectors.html |
| [122] | *Analogy (State of the art)*. |
| [123] | Word2vec. In: *Google Code Archive* [online]. [cit. 2022-02-15]. Dostupné z: https://code.google.com/archive/p/word2vec/ |
| [124] | Word2vec embeddings. In: *Gensim: Topic Modeling for Humans* [online]. 2009 [cit. 2022-06-15]. Dostupné z: https://radimrehurek.com/gensim/models/word2vec.html |
| [125] | Feature Engineering and Selection. In: KUHN, Max a Kjell JOHNSON. *Feature engineering and selection: a practical approach for predictive models*. Boca Raton, 2021, s. 242. ISBN 978-1138079229. |
| [126] | BROWNLEE, Jason. How to Perform Feature Selection With Numerical Input Data. In: *Machine Learning Mastery* [online]. 2022 [cit. 2022-07-27]. Dostupné z: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search |
| [127] | HUTTER, Frank, Holger HOOS a Kevin LEYTON-BROWN. *An Efficient Approach for Assessing Hyperparameter Importance*. 32. PMLR, 2014, , 754 s. Dostupné také z: https://proceedings.mlr.press/v32/hutter14.html |
| [128] | BERGSTRA, James a Yoshua BENGIO. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res*. JMLR.org, 2012, **13**(), 281–305. ISSN 1532-4435. |
| [129] | Cvičení z RPZ: Expectation-Maximization Algorithm. In: *Center for Machine Perception @ CTU in Prague* [online]. Prague: Czech Technical University in Prague [cit. 2022-03-07]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/recognition/Lab\_archive/RPZ\_05-06s/em/em.pdf |
| [130] | BISHOP, Christopher M. *Pattern recognition and machine learning*. [New York]: Springer, 2006. Information science and statistics. ISBN 978-0387310732. |
| [131] | BORMAN, Sean. The Expectation Maximization Algorithm: A short tutorial. In: *Laboratoire de Recherche en Informatique* [online]. [cit. 2022-03-06]. Dostupné z: https://www.lri.fr/~sebag/COURS/EM\_algorithm.pdf |
| [132] | HOFFMAN, Matthew, David BLEI a Francis BACH. *Online Learning for Latent Dirichlet Allocation*. Curran Associates Inc, 2010, 856–864 s. |
| [133] | Latent Dirichlet Allocation. In: *Gensim: Topic Modeling for Humans* [online]. [cit. 2022-04-14]. Dostupné z: https://radimrehurek.com/gensim/models/ldamodel.html |
| [134] | FAIRLESS, Andrew. Parameter Testing. In: *Andrew Fairless, Ph.D.: Data, Science, and Tinkering* [online]. 2022 [cit. 2022-04-14]. Dostupné z: https://afairless.com/the-peanuts-project/topic-modeling/parameter-testing/ |
| [135] | VANG, Jee. Topic Modeling with Gensim. In: *Data Science Topics* [online]. [cit. 2022-04-06]. Dostupné z: https://datascience.oneoffcoder.com/topic-modeling-gensim.html |
| [136] | GE, Yingqiang, Shuya ZHAO, Honglu ZHOU, Changhua PEI, Fei SUN, Wenwu OU a Yongfeng ZHANG. Understanding Echo Chambers in E-commerce Recommender Systems. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2020, 2261-2270. ISBN 9781450380164. Dostupné z: doi:10.1145/3397271.3401431 |
| [137] | FOKES, Elizabeth a Lei LI. A survey of security vulnerabilities in social networking media. *Proceedings of the 3rd annual conference on Research in information technology - RIIT '14* [online]. New York, New York, USA: ACM Press, 2014, 57-62 [cit. 2020-12-01]. ISBN 9781450327114. Dostupné z: doi:10.1145/2656434.2656444 |
| [138] | ELSAYED, Walaa. The negative effects of social media on the social identity of adolescents from the perspective of social work. *Heliyon*. 2021, **7**(2). ISSN 24058440. Dostupné z: doi:10.1016/j.heliyon.2021.e06327 |
| [139] | Chee, S.H.S., Han, J., Wang, K. (2001). RecTree: An Efficient Collaborative Filtering Method. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds) Data Warehousing and Knowledge Discovery. DaWaK 2001. Lecture Notes in Computer Science, vol 2114. Springer, Berlin, Heidelberg. Dostupné z: https://doi.org/10.1007/3-540-44801-2\_15 |
| [140] | PARTHASARATHY, S. a C.C. AGGARWAL. On the use of conceptual reconstruction for mining massively incomplete data sets. In: *IEEE Transactions on Knowledge and Data Engineering*. 2003, s. 1512-1521. ISSN 1041-4347. Dostupné z: doi:10.1109/TKDE.2003.1245289 |
| [141] | Wang, Shoujin, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. "Graph learning based recommender systems: A review." Dostupné z: https://arxiv.org/abs/2105.06339 |
| [142] | OGHENEOVO, Edward E. a Promise A. NLERUM. Iterative Dichotomizer 3 (ID3) Decision Tree: A Machine Learning Algorithm for Data Classification and Predictive Analysis. In: *International Journal of Advanced Engineering Research and Science*. 2020, s. 514-521. ISSN 23496495. Dostupné z: doi:10.22161/ijaers.74.60 |
| [143] | BERTOLOTTI, Jacopo. Gradient descent. *Wikimedia Commons* [online]. 2019 [cit. 2024-08-31]. Dostupné z: https://commons.wikimedia.org/wiki/File:Gradient\_descent.gif |
| [144] | Module 3: Examining Relationships: Quantitative Data. *Course sidekick* [online]. [cit. 2024-08-31]. Dostupné z: https://www.coursesidekick.com/statistics/study-guides/wmopen-concepts-statistics/linear-regression-2-of-4/ |
| [145] | AAAI-17 Invited Panel on AI History: Expert Systems. In: COMPUTER HISTORY MUSEUM. *YouTube* [online]. 2018 [cit. 2024-08-31]. Dostupné z: https://www.youtube.com/watch?v=Kd2vS4250Jg |
| [146] | *GOOGLE LLC. Google News [online]. 2024. [cit. 2022-09-12]. Dostupné z: https://news.google.com/home* |

SEZNAM POUŽITÝCH zkratek

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AJAX |  |  | Asynchronous JavaScript and XML | |
| API | |  |  | Application programming interface | |
| API | |  |  | Application Programming Interface | |
| CMS |  |  | Content Management System | |
| DCG@k |  |  | Discounted Cumulative Gain at k | |
| DVD | |  |  | Digital Video Disc | |
| HTTP  DCG  NDCG | |  |  | Hypertext Transfer Protocol  Discounted Cumulative Gain  Normalized Discounted Cumulative Gain | |
| JSON | |  |  | JavaScript Object Notation | |
| LDA | |  |  | Latentní Dirichletova alokace | |
| MAP |  |  | Mean Average Precision | |
| *MySQL* |  |  | Systém zprávy relační databáze; „My“ se používá pro odlišení od SQL („Structured Query Language“) jakožto dotazovacího jazyka | |
| NDCG@k |  |  | Normalized Discounted Cumulative Gain at k | |
| REST | |  |  | Representational state transfer; | |
| RMSE | |  |  | Root-mean-square error | |
| RSS | |  |  | RDF Site Summary / Really Simple Syndication | |
| RUP |  |  | Rational Unified Process | |
| SCM |  |  | Soft Cosine Measure | |
| SVD | |  |  | Singular value decomposition | |
| TF‑IDF | |  |  | Term frequency–inverse document frequency | |
| UML |  |  | Unified Modeling Language | |
| URL | |  |  | Uniform Resource Locator | |
| XML |  |  | Extensible Markup Languag | |

SEZNAM OBRÁZKŮ

[Obrázek 1: Načrtnutí principu Turingova testu. Turing chápal strojovou inteligenci jako doba uhádnutí toho, zda komunikujete přes počítač s člověkem s počítačem nebo pouze s počítačem. Čím déle toto trvá uhádnout, tím větší měla být inteligence počítače. 14](#_Toc177071028)

[Obrázek 2: Umělá inteligence z matematického pohledu. Vnímání umělé inteligence jako aproximace funkce je možná značné zjednodušení problematiky. Zejména u učení s učitelem je však tento pohled na problematiku výstižný. 18](#_Toc177071029)

[Obrázek 3: Naznačení gradientního sestupu do minima trojrozměrné funkce tvořící složitější povrch s více minimy a maximy [143]. 23](#_Toc177071030)

[Obrázek 4: Naznačení problému lokálních a globálních minim a maxim. Při optimalizaci často dojde k uváznutí v lokálním minimu. 24](#_Toc177071031)

[Obrázek 5: Znázornění měření chyby u lineární regrese pomocí metody nejmenších čtverců pomocí internetového nástroje pro vizualizaci [144]. Přímka tvoří odhadované hodnoty a datové body jsou skutečnými naměřenými body. Rozdíl nám dává stranu čtverce. Součet těchto čtverců nám dává chybu modelu, kterou se snažíme minimalizovat (optimalizovat). 25](#_Toc177071032)

[Obrázek 6: Znázornění principu Bayesovské statistiky. 30](#_Toc177071033)

[Obrázek 7: Znázornění principu redukce dimenzí. *Var 1* a *2* znázorňuje proměnné, *PC1* hlavní komponentu (v tomto velmi zjednodušeném případě) je pouze jedna. *x̄* je průměr (výběru/vzorku dat) a *s2* („s na druhou“) rozptyl (výběru dat). 31](#_Toc177071034)

[Obrázek 8: Znázornění problému měření vzdáleností u vysokých dimenzí. 31](#_Toc177071035)

[Obrázek 9: Rozhodovací strom sestavený na základě dat výše. 32](#_Toc177071036)

[Obrázek 10: Náčrtek postupu shlukovací metody. Iterativně dochází k přepočtu těžiště s dodržením zásad shlukovací metody. 33](#_Toc177071037)

[Obrázek 11: Nahoře znázornění Euklidovy a Manhattonské vzdálenosti, dole cosinovy podobnosti 34](#_Toc177071038)

[Obrázek 12: Znázornění pokročilé metody ověření modelu, tzv. *křížové validace*. 35](#_Toc177071039)

[Obrázek 13: Znázornění příkladu vizualizace lineárně oddělitelných dat a lineárně neoddělitelných. 35](#_Toc177071040)

[Obrázek 14: Parametry a vstupy v neuronové síti. 36](#_Toc177071041)

[Obrázek 15: První část výpočtu. 36](#_Toc177071042)

[Obrázek 16: Příklad klasifikace dat pomocí perceptronu. 36](#_Toc177071043)

[Obrázek 17: Ilustrace sigmoidní křivky. 37](#_Toc177071044)

[Obrázek 18: Vizualizace průběhu logaritmické funkce log(x). Zatímco je velký rozdíl mezi hodnotou na ose *y* u hodnoty na ose *x* 2 a 4, u vyšších čísel rozdíly nejsou tak významné. 37](#_Toc177071045)

[Obrázek 19: Znázornění pravděpodobností a šancí u logistické regrese. 38](#_Toc177071046)

[Obrázek 20: Znázornění vztahu mezi funkcí logit a pravděpodobností. 38](#_Toc177071047)

[Obrázek 21: Rozdíl a podobnosti perceptronu a logistické regrese. 38](#_Toc177071048)

[Obrázek 22: Logistická regrese se dá chápat jako jednoduchá neuronová síť. 38](#_Toc177071049)

[Obrázek 23: Bias ovlivňuje tvar sigmoidní funkce tvořící práh neuronové sítě. 39](#_Toc177071050)

[Obrázek 24: Náčrt principu neuronové sítě. Pozorujeme rozdíl mezi skutečnou (správnou) hodnotou v tréninkové sadě (pozorováním) a výsledkem modelu spočítaném neuronovém sítí. Na základě této chyby aktualizujeme váhy neuronové sítě. 40](#_Toc177071051)

[Obrázek 25: Likertova škála (v uživatelském rozhraní nejčastěji zobrazeno jako hvězdičky) vs unární hodnocení (nejčastěji palce nahoru nebo symboly srdce, apod.). 44](#_Toc177071052)

[Obrázek 26: Ilustrace rozdílu v datech, které jsou vstupem metod založených na klasifikaci vs metod založených na kolaborativním přístupu. 45](#_Toc177071053)

[Obrázek 27: Shlukování zpráv na Google News [3]. Sekce zobrazuje články, které se týkají amerických voleb [146]. 52](#_Toc177071054)

[Obrázek 28: Use case diagram tohoto projektu 73](#_Toc177071055)

[Obrázek 29: Neformální diagram systému (v plném rozlišení v příloze) 74](#_Toc177071056)

[Obrázek 30: RSS kanál zpravodajského serveru iDNES.cz 75](#_Toc177071057)

[Obrázek 31: Naznačení geometrického významu SVD a dvou rotací. 78](#_Toc177071058)

[Obrázek 32: Obrázek znázorňující rozdíl mezi přístupem CBOW (vlevo) a Skip-Gram (vpravo) [95] [96] 96](#_Toc177071059)

[Obrázek 33: Návrh fuzzy metody v programu *LFLC*. Na obrázku modelování proměnné „freshness“ představující čerstvost článku 111](#_Toc177071060)

[Obrázek 34: Ukázka hlavní strany webu Moje články 116](#_Toc177071061)

[Obrázek 35: Ukázka uživatelské sekce 116](#_Toc177071062)

[Obrázek 36: Ukázka sekce administrátorského nastavení 116](#_Toc177071063)

[Obrázek 37: Ukázka sekce evaluace obsahových metod 116](#_Toc177071064)

[Obrázek 38: Matice kosinových podobností 120](#_Toc177071065)

[Obrázek 39: Heatmapa TF-IDF skóre 121](#_Toc177071066)

[Obrázek 40: Srovnání 2 variant vzorců DCG. Pro binární relevanci se překrývají, tzn. nezáleží na variantě použitého vzorce 128](#_Toc177071067)

[Obrázek 41: Grafy četností náhodně zvolených parametrů modelu Word2Vec 134](#_Toc177071068)

[Obrázek 42: Korelační „teplotní“ mapa hodnocených parametrů a provedených testů 136](#_Toc177071069)

[Obrázek 43: Vliv parametrů na výsledky testů analogií a dvojic slov 136](#_Toc177071070)

[Obrázek 44: Bližší pohled na méně vlivné parametry na výsledky testů analogií a dvojic slov 137](#_Toc177071071)

[Obrázek 45: Vliv parametrů u modelů metody Doc2Vec 140](#_Toc177071072)

[Obrázek 46: Znázornění vzdáleností témat 143](#_Toc177071073)

[Obrázek 47: Nejrelevantnější slova tématu č. 2 143](#_Toc177071074)

[Obrázek 48: Nejrelevantnější slova tématu č. 7 143](#_Toc177071075)

[Obrázek 49: Nejrelevantnější termíny pro téma 3 144](#_Toc177071076)

[Obrázek 50: Hodnoty koherence pro model založený na data setu iDNES.cz 148](#_Toc177071077)

[Obrázek 51: Hodnoty koherence pro LDA model založen na datasetu Wikipedie 149](#_Toc177071078)

SEZNAM TABULEK

[Tabulka 1: Data o počasí pro rozhodovací strom. 32](#_Toc177071079)

[Tabulka 2: Srovnání kategorií na článcích převzatých z iDNES.cz 82](#_Toc177071080)

[Tabulka 3: Přehled využitých datových sad 83](#_Toc177071081)

[Tabulka 4: Srovnání nejčastějších českých slov (dle národního korpusu) a nejčastějších slov datových sad využitých v této práci 93](#_Toc177071082)

[Tabulka 5: Ukázka nalezených podobných slov ke slovu hokej. Obsahuje jména hokejistů nebo hokejových týmů, či výrazů souvisejících s tímto sportem 98](#_Toc177071083)

[Tabulka 6: Zjednodušený příklad principu Doc2Vec jakožto myšlenkový mezikrok mezi Word2Vec a Doc2Vec 104](#_Toc177071084)

[Tabulka 7: Klíčová slova extrahována ze článku o kopřivnické limuzíně Tatra 105](#_Toc177071085)

[Tabulka 8: Klíčová slova pro článek o krkonošském nosiči 106](#_Toc177071086)

[Tabulka 9: Výsledky nejpodobnějších článků 107](#_Toc177071087)

[Tabulka 10: Čerstvost článků na titulní straně iDNES.cz 110](#_Toc177071088)

[Tabulka 11: Podíly výsledků v rámci výpočtu Precison-Recall 126](#_Toc177071089)

[Tabulka 12: Popisné statistiky srovnávaných parametrů modelů Word2Vec 135](#_Toc177071090)

[Tabulka 13: Parametry variant modelů iDNES.cz 138](#_Toc177071091)

[Tabulka 14: Srovnání evaluací metod a variant modelů 142](#_Toc177071092)

[Tabulka 15: Hodnoty koherence pro model Wikipedie 149](#_Toc177071093)

[Tabulka 16: Převažující slova v rámci témat identifikovaných metodou LDA 151](#_Toc177071094)

[Tabulka 17: Výsledky uživatelského testování a obsahové metody+ 153](#_Toc177071095)

[Tabulka 18: Uživatelské testování a uživatelské metody 155](#_Toc177071096)

[Tabulka 19: Evaluace filtrování zpráv při experimentování s klasifikační metodou Random Forrest 156](#_Toc177071097)

[Tabulka 20: Evaluace filtrování zpráv při experimentování s klasifikační metodou SVC 156](#_Toc177071098)

HESLÁŘ

|  |  |
| --- | --- |
| **agregátory zpráv** | mobilní aplikace a webové služy, které uživateli sbírají na základě různých metod obsah zpravodajských webů, o který má zájem. |
| **cache** | obecně je to rychlé uložiště; v rámci doporuč. systémů často je často využívané jako doplněk k relačními databázím, či API pro co nabídnutí co nejrychlejších výsledků (které jsou např. předem vypočteny, aby na ně uživatel nemusel dlouze čekat) |
| **creative commons** | Americká nezisková vzdělávací organizace, která se zaměřuje na sběr tvůrčích děl, které mohou ostatní legálně využívat a sdílet je. |
| **dataset** | datová množina; datová sada; množina všech dat využívána pro danou metodu nebo model strojového učení |
| **distribuované zpracování** | výpočetní model, ve kterém jsou součásti softwarového systému sdíleny mezi více počítači |
| **Doc2Vec** | Rozšíření Word2Vec využívající kromě samostatný slov také celý dokument |
| **fair-use** | právní doktrína, která říká, že za určitých okolností můžete znovu použít materiál chráněný autorskými právy, aniž byste museli získat souhlas vlastníka autorských práv. |
| **FastText** | optimalizovaná knihovna pro úlohy strojového učení v rámci přirozeného jazyka vyvinutá společností Facebook |
| **Gensim** | knihovna pro programovací jazyk *Python* určená obsahující nejrůznější metody a objekty pro strojové učení v rámci přirozeného jazyka a zpracování přirozeného jazyka |
| **informační bublina** | opakovaná konzumace informací jednoho názorového spektra nebo jednoho úhlu pohledu na věc |
| **kolaborativní filtrování** | metoda doporučujících systémů založená na myšlence, že uživatelé s podobnými preferencemi nabízí dobrou příležitost pro předpovídání relevantního obsahu jiným uživatelům s podobnými preferencemi |
| **kosinová podobnost** | míra podobnosti vektorů získaná výpočtem kosínu uhlů vektorů; po vhodném převodu textu na vektory je možné jí využít pro přirozený jazyk |
| **LightGBM** | framework vyvinutý společnosti Microsoft, který se používá v oblasti strojového učení jak pro řazení výsledků, tak např. pro klasifikaci |
| **model** | v rámci této práce míněno jako model strojovéh učení; objekt, algoritmus, soubor nebo samostatný program, který dokáže vykonávat úlohy strojového učení jako rozpoznávání vzorů, shluků, klasifikace apod. |
| **nearline vrstva** | mezi vrstva doporučujících systému sloužící např. pro ukládání do paměti a načítání z paměti |
| **offline výpočety** | výpočty prováděné v rámci doporučujících systémů prováděné “bokem”; většinou za účelem časově a paměťově náročnějším trénováním modelů strojového učení nad většími datovými sadami nebo a účelem experimentů, analýz a evaluací |
| **one hot encoding** | způsob kódování, kdy jsou kategorické hodnoty rozděleny do sloupců a příslušnost položek ke kategorii je vyjádřena binárně |
| **online výpočty** | výpočty prováděné v rámci doporučujících systémů rovnou (“za běhu”) za účelem flexibility doporučujícího systému a dodáni schopnosti se mu přizpůobit co nejrychleji; jsou zde využívány ty části doporučovacého systému, které nejsou tolik náročné na výpočet |
| **Pandas** | knihovna pro programovací jazyk *Python* poskytujcííc datovou strukturu datových rámců a nejrůznější metody pro rutinní operace s daty |
| **parsování** | procházení webových stránek, nejčastějii za účelem stahování textových informací nebo obrázků a multimédií a jejich ásledné zpracování |
| **Pojem** | Text |
| **PostgreSQL** | Open-source systém pro správu relační databáze |
| **Precision and Recal** | jedna z nejčastějších metod validace využívaných jak v oboru doporučujících systému, tak v příbuzném oboru získávání informací |
| **redakční systém** | webový správce sloužící pro administraci příspěvků |
| **refaktoring** | proces vylepšení kódu bez vytváření nové funkcionality |
| **řídká matice** | matice, kteá má většinu prvků nulových; v informatice je využívána pro své výhodné vlastnosti vzhledem k šetření místa po kompresi a rychlosti výpočtů |
| **Scikit-Learn** | optimalizovaná knihovna pro programovací jazyk *Python* podporující operace lineární algebry, úlohy strojového učení, statistiky apod. |
| **TensorFlow** | open-source platforma pro strojové učení obsahující různé nástroje a knihovny podporující úlohy strojového učení |
| **use case diagram** | diagram značící případy užití systému uživateli |
| **user flow diagram** | diagram vyjadřující uživatelské akce v rámci software |
| **vektorizace** | v rámci této práce (pokud není uvedeno jinak) je míněno jako vektorizace textu  – tzn. převod slov, vět, či dokumentů přirozeného jazyka na vektorovou reprezentaci pomocí různých technik |
| **Word2Vec** | Metoda pro zpracování přirozeného jazyka, která za použití neuronové sítě natrénované na velkém množství textu umí odvozovat asociace mezi slovy |