Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andraž Hribernik

Napovedovanje optimalnega povezanega spletnega članka z MAB pristopom

magistrsko delo

študijski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2015

Izjava o avtorstvu magistrskega dela

Spodaj podpisani Andraž Hribernik, z vpisno številko **63090059**, sem avtor magistrskega dela z naslovom:

Napovedovanje optimalnega povezanega spletnega članka z MAB pristopom

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika magistrskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, 16. junij 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Dejanu Lavbiču za vodenje in usmerjanje pri izdelavi magistrskega dela. Zahvaljujem se tudi podjetju Zemanta za dostop do podatkov in Lorandu Daliju, uni. dipl. inž. računalništva za vse strokovne nasvete.

Prav tako se zahvaljujem Neži in vsem domačim za podporo in spodbude.

Remember, God does not pay us for results, but for effort.

Don Bosko

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CTR	Click-Through Rate	stopnja odzivnosti
k-NN	k-Nearest Neighbours	k-najbližjih sosedov
MAB	Multi-Armed Bandit	več ročni avtomat
RPM	Randomized Probability Matching	naključno verjetnostno ujemanje
UCB	Upper Confidence Bound	zgornja meja zaupanja

Povzetek

Cilj magistrske naloge je bil zasnovati in preizkusiti inovativen pristop za izbiro optimalne spletne novice glede na trenutno brano novico. Pomemben faktor pri izbiri našega pristopa je bilo dejstvo, da velikokrat v realnem scenariju nimamo podatkov o uporabnikih in njihovih preferencah. Zaradi tega smo za reševanje omenjenega problema uporabili Multi-armed bandit pristop. Primerjavo različnih MAB algoritmov smo naredili na problemu Jackpot, ki je bil zasnovan v sklopu Celtrinega programerskega izziva. Osnovni problem pa smo simulirali in evalvirali na realnih podatkih enega izmed ponudnikov priporočil povezane vsebine. Obstoječ priporočilni sistem smo poskušali izboljšati tako, da smo prerazporedili podobne novice trenutno brani novici. Na ta način so predhodno najboljše priporočene novice bile priporočene najvišje, hkrati pa smo vseskozi raziskovali potencialno še boljše novice. V magistrski nalogi smo preverili, ali je smiselno upoštevati pod katero novico je priporočena novica prikazana, ali je možno aproksimirati stopnjo odzivnosti s pomočjo vsebinsko sorodnih priporočilnih tokov in ali dosežemo boljše rezultate, če z Bayevsovskim pristopom aproksimiramo začetne vrednosti parametrov beta distribucije. Naš najbolj obetaven pristop je povprečno pozicijo priporočila izboljšal za približno 40% glede na naključno razporeditev vsebinsko sorodnih priporočil.

Ključne besede: priporočilni sistemi, Multi-Armed Bandit algoritmi, Randomised Probability Matching, aproksimacija stopnje odzivnosti (CTR)

Abstract

The main purpose of this master thesis was to develop and evaluate innovative approach for selection of optimal related web article. We often do not have access to data about user's profiles and their preferences. Therefore, these settings have an important influence to our research. Consequently, we decided to use Multi-Armed bandit approach to deal with described settings. Comparison of different MAB algorithms has been done on Jackpot problem, which was designed for Celtra programming challenge. The main research problem has been simulated and evaluated on a real data obtained from a provider of related web content. We tried to improve existing recommendation system with reordering similar news to currently read one. Using this approach, the most interesting news have been recommended on top positions and the most promising news have been explored as well. In this master thesis, we tried to answer to the following questions: does it make sense to take into account statistics of recommended news in context of every news stream separately; is it possible to approximate click-through rate using content-related recommendation streams; could we achieve better results, if we approximate initial input parameters of beta distribution using Bayesian approach. Our most promising method has achieved more than 40% average position improvement regarding random selection strategy of content-related recommendations.

Keywords: Recommendation systems, Multi-Armed Bandit algorithms, Randomised Probability Matching, Approximation of Click-Through Rate

Poglavje 1

Uvod

Priporočilni sistemi so pomemben člen informacijskega in e-poslovnega ekosistema. Verjetno najbolj poznana primera uspešnih priporočilni sistemov sta Amazonov [1] in Netflixov [2] priporočilni sistem. Ta dva sistema priporočata na tisoče artiklov milijonom uporabnikov njihovega sistema. Podjetje Amazon je tudi zaradi odličnega priporočilnega sistema povsem spremenil celotno industrijo, ki se ukvarja s prodajo knjig. Priporočilni sistem lahko vsakemu uporabniku personalizirano priporoči knjige glede na njegove želje in preference. To pomeni, da lahko kupcu predstavijo manj znane knjige, ki so všeč samo njemu (oziroma manjši skupini ljudi) in nam ni potrebno predstaviti zgolj najpopularnejših knjig [3]. Klasične knjigarne pa se koncentrirajo na povprečnega kupca, prav tako so prostorsko omejene in niti ne morejo v trgovini predstaviti tolikšnega števila različnih knjig.

V zadnjem času se je aplikacija priporočilnih sistemov zelo razširila in ni več vezana zgolj na področje e-poslovanja. Eno izmed zelo razširjenih področji je tudi priporočanje povezanega spletnega članka. Spletni članek je opredeljen kot spletna novica, spletni blog in podobne spletne vsebine iz različnih spletnih virov. O povezanem članku govorimo zato, ker je priporočen članek prikazan ob trenutnem članku, ki ga uporabnik pregleduje. Pogosto so takšni povezani spletni članki na spletnih virih prikazani v rubrikah z naslovi, kot so: "Preberi več", "Priporočamo", "Drugi bralci so pre-

brali", itd. Cilj je, da je priporočilni sistem tako kakovosten, da so priporočeni spletni članki zanimivi uporabnikom. Boljši kot je priporočilni sistem, večje število uporabnikov bo kliknilo na priporočene spletne članke in višja bo stopnja odzivnosti (ang. CTR). Običajno prebiranje spletnih člankov ne zahteva prijave uporabnikov in je zaradi tega identifikacija uporabnikov težja. Poleg tega je priporočanje spletnih povezanih vsebin velikokrat prepuščeno zunanjemu akterju, ki ni neposredno povezan z izdelavo spletne strani in nima možnosti dostopa do spletnih piškotkov, spletnega brskalnika in drugih podobnih tehnologij za identifikacijo uporabnika. Zaradi tega bomo v sklopu te raziskave upoštevali omejitev, da identifikacija uporabnika ni možna. V tej magistrski nalogi se bomo ukvarjali s problemom optimalne izbire spletnega povezanega članka z Multi-Armed Bandit (v nadaljevanju MAB) pristopom. Glavna prednost MAB algoritmov je ta, da v osnovi za svoje delovanje ne potrebujejo podatkov o uporabnikih, ampak izbirajo takšne članke, ki bodo z najvišjo verjetnostjo kliknjeni pri uporabnikih. Za svoje delovanje potrebujejo zgolj statistiko predhodnih priporočil in katera priporočila so bila kliknjena.

V Poglavju 2 bomo predstavili dva pristopa (vsebinsko in skupinsko filtriranje), ki se uporabljata na področju priporočilnih sistemov. Prav tako bomo pojasnili zakaj skupinsko filtriranje ni primerno za reševanje našega problema. Ob koncu tega poglavja bomo predstavili konkreten priporočilni sistem, ki temelji na vsebinskem filtriranju in ga bomo poskušali izboljšati v sklopu te naloge. Poglavje 3 bo razložilo osnovni princip MAB algoritmov, ki je iskanje najboljšega razmerja med raziskovanjem novega znanja in izkoriščanjem trenutnega znanja. Poleg tega bomo v tem poglavju predstavili nekaj osnovnih MAB algoritmov in preverili njihovo delovanje na primeru naloge Jackpot (Poglavje 4), ki je bila zastavljena v okviru Celtrinega programerskega izziva [4]. V Poglavju 5 bomo predstavili na kakšen način bomo nadgradili obstoječ priporočilni sistem z MAB pristopom in opisali različne metode, ki jih bomo implementirali. Postopek evalvacije metod in vrednotenje dobljenih rezultatov bomo opisali v Poglavju 6. Za konec (Poglavje 7)

pa bomo predstavili zaključne ugotovitve naše magistrske naloge.

Poglavje 2

Pregled obstoječih metod priporočilnih sistemov

V tem poglavju bomo predstavili osnovne principe, na katerih temelji večina sodobnih priporočilnih sistemov. Te principe lahko uvrstimo v dve skupini priporočilnih sistemov [5]:

Vsebinski (ang. content based) priporočilni sistemi preučijo lastnosti artiklov. Priporočilo temelji na podobnosti artiklov glede na ključne lastnosti. Na primer, če je nek uporabnik pogledal že veliko akcijskih filmov nekega režiserja, bi mu takšen priporočilni sistem predlagal še neogledane akcijske filme istega režiserja.

Skupinsko filtriranje (ang. collaborative filtering) temelji na podobnosti med uporabniki in/ali artikli. Če so dvema uporabnikoma všeč podobni filmi, bo sistem drugemu uporabniku predlagal tiste filme, ki so bili všeč prvemu uporabniku in si jih drugi uporabnik še ni ogledal.

Ta dva koncepta priporočilnih sistemov bosta podrobneje predstavljena v nadaljevanju. Predstavili bomo prednosti in pomanjkljivosti posameznega koncepta [6] ter le-te povezali z našo omejitvijo o nezmožnosti identifikacije uporabnika. Ob koncu pa bomo predstavili še konkreten vsebinski priporočilni

sistem, ki ga bomo uporabili za evalvacijo naših metod.

2.1 Vsebinski priporočilni sistemi

Pri vsebinskih priporočilnih sistemih moramo za vsak artikel izdelati profil, ki predstavlja pomembne lastnosti tega artikla. Poglejmo nekaj enostavnih lastnosti filmov, ki bi lahko sestavljali profil in so relevantni za priporočilni sistem [7].

Množica igralcev, ki nastopajo v filmu

Nekateri gledalci imajo raje filme s točno določenimi igralskimi zasedbami.

Režiser filma

Nekateri gledalci raje gledajo filme določenih režiserjev.

Leto izdaje filma

Nekateri gledalci raje gledajo starejše filme, druge zanimajo samo novejši filmi.

Filmski žanr oziroma tip filma

Nekateri gledalci raje gledajo akcijske filme, drugi gledajo drame in komedije.

Seveda bi lahko profil sestavili še iz drugih lastnosti. Vsako izmed lastnosti si lahko predstavljamo kot eno dimenzijo v vektorskem prostoru. Na ta način lahko primerjamo sorodnost med filmi. Če želimo primerjati med seboj besedilne dokumente, lahko ustvarimo nabor besed in primerjamo, katere besede se pojavljajo in kolikokrat v vsakem dokumentu. Več o vektorski predstavitvi besedil in kosinusni podobnosti, ki se uporablja kot mera podobnosti, bomo razložili v Poglavju 5.6.1.

Denimo, da imamo na voljo zelo preprost profil, ki je sestavljen iz množice petih filmskih igralcev in povprečne ocene filma. V Tabeli 2.1 imamo primer

Tabela 2.1: Vektorska predstavitev filmov glede na igralsko zasedbo petih filmskih igralcev

	Igralec1	Igralec2	Igralec3	Igralec4	Igralec5	**povp. oc.**
Film1	1	0	1	0	1	0.6
Film2	1	1	0	0	0	0.8
Film3	0	0	1	1	1	1

vektorske predstavitve filmov. Posamezni stolpci imajo vrednost 1 za igralca, ki nastopa v nekem filmu in 0, če ta igralec ne nastopa v tem filmu. Povprečna ocena za film je normirana in je predstavljena na lestvici od 0 do 1.

Če uporabimo kot mero za iskanje sorodnosti med filmi kosinusno podobnost (definirana v enačbi (2.1)) vidimo, da sta si najbolj sorodna Film1 in Film3 (izračun podobnosti v enačbi (2.2)) ter najmanj Film2 in Film3.

$$sim(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\mid \vec{d_1} \mid\mid \vec{d_2} \mid} \tag{2.1}$$

Film1 in Film3 v vektorske prostoru zapišemo kot $\overrightarrow{\text{Film}_1} = [1, 0, 1, 0, 1]$ in $\overrightarrow{\text{Film}_1} = [0, 0, 1, 1, 1]$. Glede na enačbo (2.1) je torej:

$$sim(\overrightarrow{\text{Film}_1}, \overrightarrow{\text{Film}_2}) = \frac{2}{\sqrt{3}\sqrt{3}} = \frac{2}{3} = 67\% \tag{2.2}$$

Najpreprostejša metoda, ki bi jo lahko uporabili kot vsebinski priporočilni sistem bi bila, da bi ob prikazu ali nakupu Filma1 priporočili še Film3, ker je najbolj podoben glede na igralsko zasedbo. Takšen priporočilni sistem bi uporabili, če ne bi vedeli ničesar o uporabniku. V primeru, da lahko identificiramo uporabnika in poznamo njegove predhodne, ocene lahko izračunamo uporabniški profil. Uporabniški profil je enake dimenzije kot profil filma. Denimo, da stolpec igralec2 predstavlja igralko Jennifer Aniston, potem bi na istem mestu v uporabniškem profilu izračunali povprečno uporabnikovo oceno za filme, v katerih igra Jennifer Aniston. Priporočilo pa bi izdelali tako, da bi izračunali kosinusno podobnost med uporabniškim profilom in vsemi filmi in priporočili film, ki je najbolj soroden glede na uporabnikov okus.

Tabela 2.2: Matrika uporabnikov in njihovih ocen

	a	**b**	**c**	**d**	**e**	**f**
A	4			5	1	
B	5	5	4			
C				2	4	5
D		3			4	3

Velika prednost vsebinskega priporočanja je ta, da ne potrebujemo velike količine podatkov o uporabnikih in njihovih odzivih. Zaradi tega ne prihaja do problema hladnega začetka (angl. cold start problem) in lahko brez težav priporoča nove artikle novim uporabnikom. Prav tako nam ta način omogoča priporočanje za zelo specifičen uporabnikov okus in ni skoncentriran zgolj na priporočanje najpopularnejših artiklov. Poleg tega je njegova velika prednost ta, da je intuitiven in z njim brez težav razložimo dano priporočilo. Na drugi strani pa je slabost vsebinskega priporočanja ta, da je težko poiskati primerne atribute, ki dobro opišejo artikle. Prav tako je takšen sistem preveč specializiran in ni sposoben priporočiti artiklov, ki so izven uporabnikovega profila (morda uporabnika, ki bere o programiranju zanima tudi znanstvena fantastika in psihologija).

2.2 Skupinsko filtriranje

Skupinsko filtriranje uporablja povsem drugačen pristop pri priporočanju. Namesto, da se osredotoča na atribute artiklov (kar je ena izmed pomanjkljivosti vsebinskih priporočilnih sistemov), se osredotoča na podobnost med uporabniškimi ocenami artiklov. Osnova za nadaljnjo razlago delovanja skupinskega filtriranja je matrika uporabnikov in njihovih ocen (v nadaljevanju U). Primer takšne matrike U je v Tabeli 2.2, kjer vsaka vrstica predstavlja uporabnika in vsak stolpec ocene za posamezen artikel. Artikli, ki jih uporabnik ni ocenil, so prazna polja v matriki.

Skupinsko filtriranje bi lahko konceptualno razdelili na dve področji. In

sicer, skupinsko filtriranje, ki temelji na podobnosti med uporabniki in tisto, ki temelji na podobnosti med artikli [8]. Če želimo izračunati priporočilo, ki temelji na podobnosti med uporabniki, za uporabnika u in artikel i, to storimo z enačbo (2.3),

$$p_{u,i} = \frac{\sum_{u' \in N} s(u,u') \cdot r_{u',i}}{\sum_{u' \in N} |s(u,u')|} \tag{2.3}$$

kjer je množica N sestavljena iz uporabnikov, ki so najbolj sorodni uporabniku u, $s(u,u')$ je podobnost med uporabnikoma u in u' in $r_{u',i}$ ocena uporabnika u' za i-ti artikel.

Denimo, da želimo oceniti, s kolikšno oceno bo uporabnik B ocenil artikel e ter velikost množice $|N| = 2$. Za podobnost med uporabniki ponovno uporabimo kosinusno mero podobnosti in dobimo naslednje rezultate: $s(B,A) = 0.38, s(B,C) = 0, s(B,D) = 0.26$. To pomeni, da je množica $N \in \{A,D\}$, saj sta uporabnika A in D najbolj sorodna uporabniku B. Iz tega sledi, da je:

$$\begin{aligned} p_{B,e} &= \frac{\sum_{u' \in \{A,D\}} s(B,u') \cdot r_{u',e}}{\sum_{u' \in \{A,D\}} |s(B,u')|} \\ p_{B,e} &= \frac{0.38 \cdot 1 + 0.26 \cdot 4}{0.38 + 0.26} \\ &= \frac{1.42}{0.64} = 2.22 \end{aligned} \tag{2.4}$$

Na ta način lahko za vsakega uporabnika ocenimo manjkajoče ocene in priporočimo tiste artikle, ki imajo najvišje ocenjene vrednosti. Omenjeno enačbo se lahko izboljša tako, da upoštevamo povprečne ocene uporabnikov in na ta način izenačimo uporabnike, ki so nagnjeni k višjemu oziroma nižjemu ocenjevanju.

Na povsem podoben način deluje skupinsko filtriranje, ki temelji na podobnosti med artikli. Če se pri uporabniškem skupinskem filtriranju računa sorodnost med vrsticami matrike uporabnikov in njihovih ocen, se v tem primeru računa sorodnost med stolpci. Priporočilo za uporabnika u in artikel i izračunamo z enačbo (2.5),

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) \cdot r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{2.5}$$

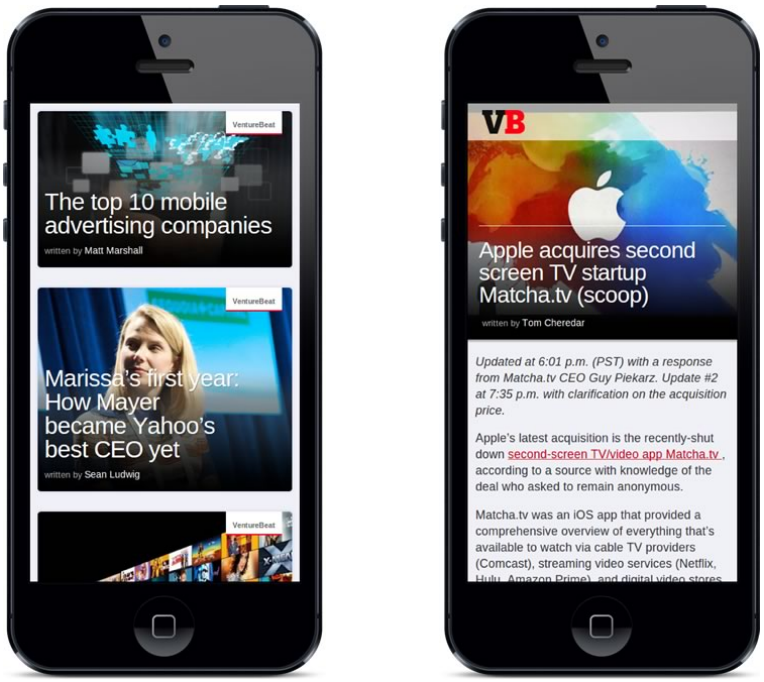
kjer je S množica vseh podobnih artiklov i-temu artiklu (glede na uporabnikove ocene), $s(i, j)$ je podobnost med artikloma i in j in $r_{u,j}$ ocena uporabnika u za artikel j iz množice S.

Zgoraj opisana principa skupinskega filtriranja je možno izboljšati tudi tako, da se kombinirata oba pristopa. Poleg tega se v kombinaciji s skupinskim filtriranjem običajno omenja tudi matrično faktorizacijo [2]. To je množica algoritmov, ki matriko U razcepi na dve manjši matriki katerih produkt je približno enak originalni matriki ($U \approx W \times H$). Za razliko od matrike U, ki je redka (večina uporabnikov oceni zgolj nekaj artiklov), imata matriki W in H same ne ničelne elemente. Na ta način se zmanjša prostorska kompleksnost, hkrati pa se odpravi določen šum. Pokazalo se je, da je na ta način možno občutno izboljšati osnoven princip skupinskega filtriranja [2].

Matrične faktorizacije ne bomo podrobneje predstavljali, saj pri našem konkretnem problemu ne moremo identificirati uporabnika. Zaradi tega ne moremo zgraditi matrike U, ki je osnova vseh metod skupinskega filtriranja. Poleg tega s tem pristopom ne moremo nikoli priporočiti artikla, ki še ni bil ocenjen. Ena izmed prednosti vsebinskega odločanja je tudi ta, da lahko priporoča artikle za uporabnike z zelo specifičnim okusom, česar ta metoda ne more. Prednost skupinskega odločanja je v tem, da nam ni potrebno izbirati atributov in omogoča, da uporabniku priporoči artikle izven njegovega trenutnega okusa. Torej uporabniku, ki prebira knjige o programiranju, lahko priporoči tudi knjige o znanstveni fantastiki, saj je druge uporabnike, ki so brali o programiranju, zanimala tudi znanstvena fantastika.

2.3 Priporočilni sistem Zemanta Streams

Vedno več uporabnikov dostopa do svetovnega spleta preko mobilnih naprav. Zaradi tega večina spletnih strani z novicami uporabniku prikaže drugačen spletni vmesnik, takrat ko do njihove strani dostopa preko mobilne naprave. Zemanta Streams [9] je izdelek mednarodnega podjetja, ki preoblikuje novičarsko stran v neskončen vsebinski tok, ki je prikazan na Sliki 2.1.



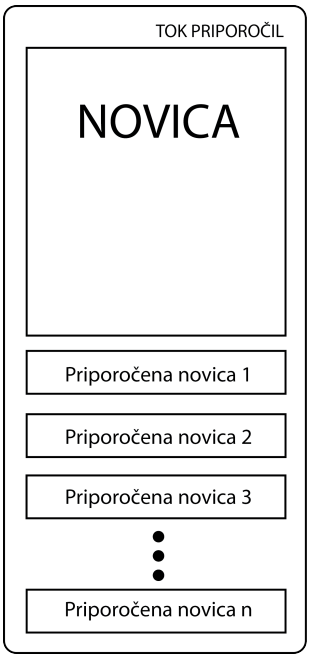
Slika 2.1: Vsebinski priporočilni sistem Zemanta Streams

Vsebinski tok je urejen vertikalno. To pomeni, da je na vrhu prikazana osrednja novica, v nadaljevanju pa sledijo novice, ki so vsebinsko podobne osrednji novici. Ko se uporabnik drsno premika navzdol (glede na zaslon), se priporočene novice dinamično nalagajo. Bolj strukturiran prikaz Zemanta Streams priporočilnega sistema je prikazan na Sliki 2.2. V nadaljevanju se bomo sklicevali na omenjeni priporočilni sistem, zato bi radi poudarili tri pojme:

Novica (v nadaljevanju z veliko začetnico) je besedilo, ki si ga je uporabnik v osnovi želel prebrati. Ko govorimo o novici znotraj toka priporočil, imamo vedno v mislih to novico. Vsak tok priporočil ima zgolj eno Novico in mnogo priporočenih novic.

Tok priporočil je seznam vseh priporočenih novic ob pripadajoči Novici. Priporočene novice, ki tvorijo tok priporočil, so razporejene glede ne vsebinsko podobnost z Novico v padajočem vrstnem redu.

Priporočena novica je oznaka, ki je navadno sestavljena iz fotografije in naslovne novice, ki je po vsebini sorodna Novici z vrha ekrana. Če je uporabnik zainteresiran za priporočeno novico, klikne nanjo in se prikaže celotna novica.



Slika 2.2: Komponente priporočilnega sistema Zemanta Streams

V začetku tega poglavja smo opisali dva koncepta priporočilnih sistemov. Zemanta Streams je vsebinski priporočilni sistem, saj so priporočene novice razporejene glede na vsebinsko podobnost z novico na vrhu. Poleg tega gre za zelo preprost priporočilni sistem, saj je vrstni red priporočil vedno enak, ne glede na okus uporabnika ali pa popularnost posameznih priporočenih novic. Ta sistem nam ne omogoča identifikacije uporabnika, zagotavlja pa nam informacijo o tem, katere priporočene novice so bile kliknjene in katere ne.

2.4 Predlog rešitve

V sklopu te magistrske naloge želimo ta obstoječ sistem izboljšati tako, da bomo vsebinsko sorodne novice prerazporedili v takšnem vrstnem redu, da bodo popularnejše novice priporočene višje. Na ta način bomo povečali zaupanje v priporočilni sistem in bo večje število uporabnikov pregledala vsaj prvih nekaj priporočil. Posledično se bo povečal povprečen čas, ki ga bo uporabnik preživel na spletni strani.

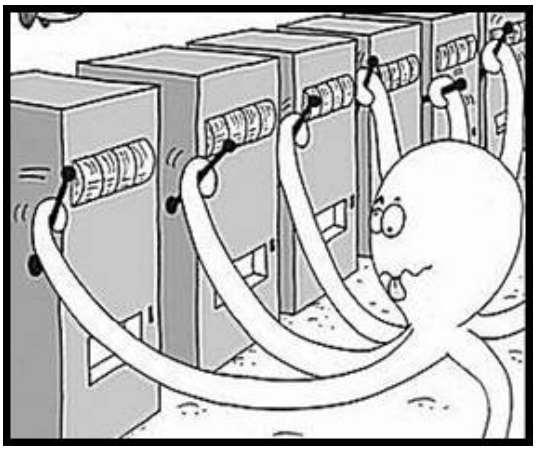
To bomo dosegli s pomočjo MAB algoritmov (več o njih v Poglavju 3), ki bodo najpopularnejše priporočene novice razvrstili na najvišje pozicije. Ti algoritmi bodo znatno povečali dinamičnost obstoječega priporočilnega sistema, saj bodo poleg identifikacije najboljših priporočenih novic, sposobni identificirati tudi nove potencialno še boljše priporočene novice.

Poglavje 3

MAB pristop

Pri produktu Zemanta Streams gre za primer vsebinskega priporočilnega sistema. Zaradi omenjene omejitve (identifikacija uporabnika ni možna) gre pri optimizaciji trenutnega sistema za klasičen problem iskanja kompromisa med priporočanjem popularnih novic in novic, ki bi lahko bile še popularnejše. Takšne probleme rešuje množica algoritmov, ki jih poznamo pod skupnim imenom Multi-Armed Bandit (v nadaljevanju MAB) algoritmi. Ti algoritmi so dobili ime po igralnih avtomatih z več ročicami, saj rešujejo problem, ki ga najlažje opišemo na primeru igralnih avtomatov. Imamo torej igralni avtomat z več ročicami (ali več igralnih avtomatov z eno ročico) in omejeno število potegov ročice. Ob vsakem potegu se moramo odločiti, katero izmed ročic bomo potegnili (kot to prikazuje Slika 3.1) s ciljem, da bo dobiček čim večji. Slika 3.1 zelo slikovito ponazori celoten problem, kjer hobotnica igra vlogo MAB algoritma. Ker ne vemo, s kolikšno verjetnostjo posamezna ročica izplača nagrado, je cilj MAB algoritma, da v vsakem koraku izbere tisto, ki ima največjo verjetnost za izplačilo nagrade.

V literaturi [10, 11] se običajno govori o optimalnem razmerju med izkoriščanjem in učenjem (ang. exploration vs. exploitation). Na primer, pri izboru najbolj dobičkonosne ročice igralnega avtomata je potrebno preizkusiti vse ročice (učenje), da pridobimo neko informacijo o verjetnosti za izplačilo nagrade za posamezno ročico. Ko imamo dovolj znanja, pa lahko



Slika 3.1: Slikovit prikaz MAB eksperimenta [12].

izkoriščamo znanje o ročicah in v nadaljevanju vlečemo tisto, ki nam prinaša največji dobiček. Seveda je omenjeni opis poenostavljen, saj večina MAB algoritmov ne ločuje med fazo učenja in raziskovanja tako eksplicitno. Prav tako se lahko verjetnost za izplačilo nagrade pri igralnih ročicah spremeni, zaradi tega se morajo MAB algoritmi prilagoditi tudi temu.

MAB princip je lahko uporabljen na veliko različnih področjih in aplikacijah. Ta princip je uporabljen pri kliničnih poskusih, kjer se preizkuša različne metode zdravljenja, ob tem pa se minimizira število stranskih učinkov [13]. Zelo znan produkt, ki uporablja MAB princip, je Googlov spletni optimizator [14]. Ta omogoča, da lahko oblikovalec spletne strani oblikuje več verzij spletne strani, ki se razlikujejo v barvni shemi, pisavi, itd. To orodje na podlagi eksperimentiranja ugotovi, katera verzija je najboljša. V našem primeru bomo MAB pristop uporabili v povezavi s priporočilnimi sistemi. Kontekstualni MAB algoritem je uporabljen na YahooNews spletni strani. Le ta priporoči uporabniku tisto novico, ki je najbolj verjetna, da jo bo kliknil glede na njegove predhodne preference [15]. V našem primeru identifikacija uporabnika ni možna, zato bo naš pristop nekoliko drugačen.

Glavna razlika med klasičnim eksperimentiranjem in eksperimentiranjem, ki temelji na MAB, je v uravnoteženosti eksperimenta. Pri klasičnem ekspe-

rimentu bi enakomerno preizkušali vse ročice in ob koncu poskusa ugotovili, katera izmed ročic je najboljša. Tak poskus je seveda statistično bolj zanesljiv, vendar v praksi s takšnim eksperimentom izgubimo veliko sredstev. Če si ročice predstavljamo kot oglase, kjer zaslužimo ob vsakem kliku oglasa, to pomeni, da bomo uravnoteženo prikazovali tako dobre kot slabe oglase. S pomočjo MAB algoritmov se sprotno učimo, katere ročice so boljše od drugih in tiste boljše izkoriščamo na ta način, da jih večkrat potegnemo. V primeru oglasov, bi to pomenilo, da bi večkrat prikazali oglase, ki so dobri. S tem bi ob omejenem številu potegov ročic povečali dobiček.

3.1 MAB model

Matematično je v reviji [10] MAB model definiran z naključno spremenljivko $X_{i,n}$ za $1 \leq i \leq K$ in $n \geq 1$, kjer i predstavlja indeks igralnega avtomata in K številov vseh igralnih avtomatov. S spremenljivko n pa definiramo zaporedno številko potega. Zaporedni potegi igralnih avtomatov i imajo nagrade $X_{i,1}, X_{i,2}, ...$ Igralni avtomati so med seboj neodvisni in imajo nam neznano verjetnost izplačila nagrade μ_i.

Algoritem A v vsakem trenutku izbere en avtomat iz množice K avtomatov. Algoritmu A pravimo dodeljevalna strategija (ang. allocation strategy) in njegovo uspešnost merimo z izgubo (ang. regret), ki jo algoritem povzroči v n potegih. Izguba ρ je definirana z enačbo (3.1),

$$\rho = \mu_n^* - \sum_{t=1}^{n} \mu_t \tag{3.1}$$

kjer je μ_n^* maksimalen možen dobiček po n potegih in $\sum_{t=1}^{n} \mu_t$ dobiček, ki ga je algoritem A priigral. Poznamo več različnih dodeljevalnih strategij A, ki minimizirajo izgubo in jih bomo nekaj izmed njih predstavili v naslednjem poglavju.

3.2 MAB algoritmi

Algoritmov ϵ-Greedy in UCB nismo implementirali sami, ampak smo uporabili implementacijo iz [16].

3.2.1 ϵ-Greedy

ϵ-Greedy strategija je najpreprostejši algoritem za optimizacijo problema raziskovanja in izkoriščanja znanja. Tako kot smo problem raziskovanja in izkoriščanja dosedanjega znanja razdelili v dve ločeni fazi; tako deluje tudi ta algoritem. Z verjetnostjo ϵ je v fazi raziskovanja in izbere naključno ročico, medtem ko z verjetnostjo $1-\epsilon$ izkorišča trenutno znanje in potegne najboljšo ročico. Psevdokoda ϵ-Greedy je zapisana v algoritmu 1, kjer funkcija be-

Data: arms, epsilon
if *randomFloatBetween(0,1) > epsilon* **then**
| **return** *bestArm(arms)*
else
| **return** *randomArm(arms)*
end

Algoritem 1: ϵ-Greedy algoritem

stArm vrne tisto ročico iz nabora vseh ročic, ki ima najvišjo verjetnost klika do trenutka izbire in funkcija randomArm vrača naključno ročico iz seznama le teh.

3.2.2 Upper Confidence Bound

Značilnost ϵ-Greedy strategije je ta, da privzeto vedno izbere ročico, ki ima v danem trenutku najboljšo ocenjeno verjetnost nagrade ter raziskuje popolnoma naključno. Avtor knjige [16] je ocenil ta algoritem kot naiven, saj je lahko ocenjena vrednost zavajajoča v primeru nekaj negativnih poizkusov. Glavni pomanjkljivosti ϵ-Greedy algoritma sta, da je uspešnost preveč od-

visna od naključja in da upošteva zgolj ocenjeno verjetnost nagrade in ne zaupanja v to oceno (kolikokrat smo potegnili posamezno ročico).

Omenjeni dve pomanjkljivosti rešuje Upper Confidence Bound (v nadaljevanju UCB) algoritem. Le ta ne uporablja nobene naključnosti ter za vsako ročico izračuna vrednost glede na enačbo (3.2),

$$ucb_{a,n} = value_a + \frac{\sqrt{2 \log n}}{n_a} \tag{3.2}$$

kjer je $value_a$ povprečna nagrada za ročico a, n število vseh potegov, ki smo jih naredili do tega trenutka in n_a število potegov za izbrano ročico. UCB algoritem izbere tisto ročico, ki ima največjo vrednost glede na enačbo (3.2). Psevdokoda za UCB je navedena v algoritmu 2. Funkcija *count*(*arm*)

```
Data: arms
foreach arm ∈ arms do
    if count(arm) == 0 then
        return arm
    end
end
ucbValues = [0 for arm ∈ arms ]
totalCounts = sum([count(arm) for arm ∈ arms])
foreach arm ∈ arms do
    bonus = sqrt(2 log totalCounts / count(arm))
    ucbValues[arm] = values[arm] + bonus
end
return bestArm(ucbValues)
```

Algoritem 2: UCB algoritem

vrača število dosedanjih prikazov za izbran avtomat in funkcija *values*(*arm*) vrača povprečen zaslužek avtomata.

3.2.3 Randomized Probability Matching

Za Randomized Probability Matching (v nadaljevanju RPM) strategijo bi lahko dejali, da združuje lastnosti ϵ-Greedy in UCB strategije. RPM se

pri svojem delovanju do določene mere zanaša na naključje, na drugi strani strani pa upošteva dejstvo, da so ocene verjetnosti nagrade bolj zanesljive pri ročicah, ki smo jih večkrat potegnili.

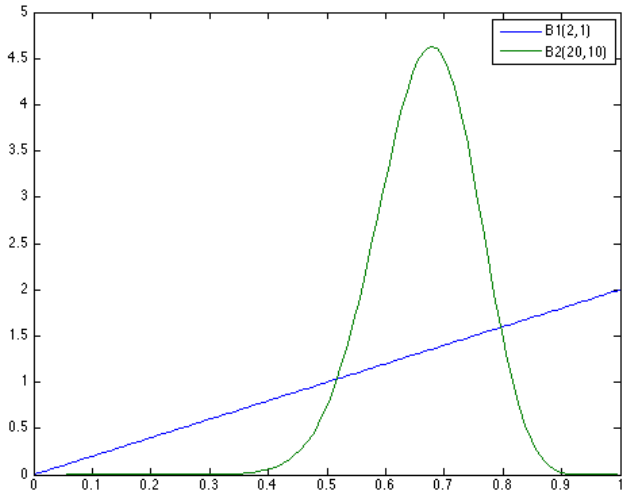
Formalno je ta metoda izpeljana in definirana v [11]. Bistvo te metode je, da je ocena verjetnosti nagrade ročice naključno porazdeljena glede na beta distribucijo. Beta distribucija je družina zveznih verjetnostnih funkcij definiranih na intervalu $[0, 1]$ z dvema vhodnima parametroma α in β. Verjetnostna porazdelitvena funkcija naključne spremenljivke x ob danih vhodnih parametrih α in β je definirana v enačbi (3.3).

$$f_{\alpha,\beta}(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}} \tag{3.3}$$

Srednja vrednost in standardni odklon beta distribucije pa sta definirani v enačbi (3.4).

$$\begin{aligned} \mu &= \frac{\alpha}{\alpha+\beta} \\ \sigma^2 &= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \end{aligned} \tag{3.4}$$

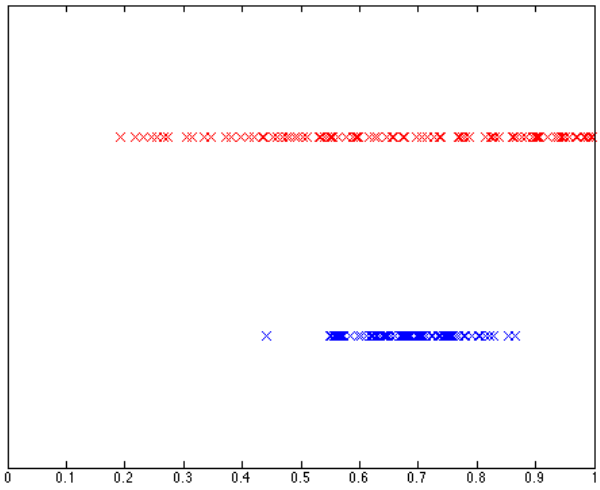
Denimo, da imamo dve beta distribuciji $B_1(2, 1)$ in $B_2(20, 10)$. Obe distribuciji imata enako srednjo vrednost ($\mu = \frac{2}{3}$), a različna standardna odklona ($\sigma^2_{B_1} = 0.056$ in $\sigma^2_{B_2} = 0.007$). Slika 3.2 prav tako kaže razliko v verjetnostni porazdelitvi naključne spremenljivke x. Vidimo, da bi naključna spremenljivka x, ki je porazdeljena glede na B_2 najverjetneje zavzela vrednost, ki je blizu povprečne vrednosti, medtem ko bi x glede na B_1 z največjo verjetnostjo imel vrednost 1. Za izbrani beta distribuciji B_1 in B_2 smo naredili eksperiment, kjer smo 100-krat generirali naključno vrednost po obeh distribucijah. Porazdelitev naključnih spremenljivk smo prikazali na Sliki 3.3. Vidimo, da so vrednosti naključne spremenljivke pri distribuciji B_1 porazdeljene na celotnem intervalu [0,1]. Največja gostota je v okolici vrednosti 1, kar ustreza Sliki 3.2. Prav tako je porazdelitev druge naključne spremenljivke (glede na distribucijo B_2) pričakovana. Vrednosti so zelo zgoščene okrog povprečja in splošna razpršenost je veliko manjša. Pri tem eksperimentu smo ob vsakem



Slika 3.2: Porazdelitev naključne spremenljivke x glede na B_1 in B_2

žrebu primerjali vrednosti obeh naključnih spremenljivk. Ugotovili smo, da je v 47% višjo vrednost zavzela naključna spremenljivka, ki je porazdeljena po B_1 in v 53% naključna spremenljivka, ki je porazdeljena po B_2.

MAB algoritem deluje torej tako, da oceni verjetnost nagrade ročice naključno glede na beta distribucijo, ki je določena s parametroma α in β. Vrednost teh dveh parametrov pa je določena s predhodnimi poizkusi. Vrednost parametra α je enaka številu uspešnih potegov ročice izbranega avtomata, vrednost parametra β pa številu neuspešnih potegov. V algoritmu 3 je zapisana pseudokoda strategije RPM, kjer funkcija $betaVariate(\alpha, \beta)$ vrača naključno vrednost distribuirano glede na beta distribucijo, ki jo določata parametra α in β. Funkcija success(arm) pa vrača število uspešnih predhodnih priporočil izbrane ročice (kolikokrat je izbrana ročica vrnila dobitek).



Slika 3.3: Simulacija naključne spremenljivke x glede na B_1(rdeča) in B_2(modra)

Data: arms
rpmValues = [0 for arm ∈ arms]
foreach *arm* ∈ *arms* **do**
    failures = count(arm) - sucess(arm)
    rpmValues[arm] = betaVariate(success(arm), failiures)
end
return *bestArm(rpmValues)*

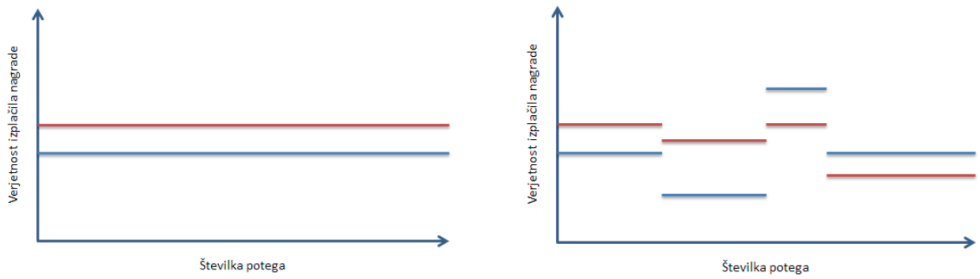
Algoritem 3: RPM algoritem

Poglavje 4

Uporaba MAB pristopa na problemu Jackpot

4.1 Opis problema

Podjetje Celtra je v sklopu študentskega programerskega izziva [4] zasnovala problem Jackpot. Cilj problema je bi izdelava algoritma, ki je sposoben priigrati čim večji zaslužek na množici igralnih avtomatov. Šlo je torej za klasičen MAB problem, ki nam bo služil za primerjavo algoritmov. MAB algoritme bomo primerjali na 10 primerih. Primeri se razlikujejo po številu ročic, številu potegov in verjetnosti zaslužka, ki ga ima posamezna ročica. V splošnem bi lahko primere razdelili v dve množici, in sicer takšne, kjer imajo ročice konstantno verjetnost izplačila (Slika 4.1a) in druge, kjer se verjetnost izplačila posamezne ročice spreminja skozi čas (Slika 4.1b). Pri primerih, kjer je verjetnost izplačila nagrade konstanta, je cilj algoritma ta, da čim hitreje ugotovi, katera ročica vrača najvišji zaslužek in potem do konca simulacije algoritem izbira zgolj to ročico. Pri spreminjajočih primerih pa mora biti algoritem takšen, da se je sposoben čim hitreje prilagajati spremembi. Seveda pa mora oba tipa problemov reševati isti algoritem, ker ne vemo v naprej, za kakšen primer gre. Izvorna koda rešitve je na voljo v repozitoriju programske kode [17].



(a) Konstantna verjetnost izplačila (b) Spreminjajoča se verjetnost izplačila

Slika 4.1: Prikaz statičnih in dinamičnih verjetnosti za izplačilo nagrad [4]

4.2 Primerjava uspešnosti MAB algoritmov na problemu Jackpot

Povprečne izgube, ki so jih na teh desetih primerih dosegli MAB algoritmi, so prikazane v Tabeli 4.1. Potrebno je poudariti, da gre za povprečne rezultate, ki smo jih dobili tako, da smo vsak primer ponoviti 20-krat. Vsak primer smo večkrat ponovili, saj nismo želeli, da bi se na rezultatih poznal vpliv naključja. Prav tako so rezultati agregirani. Stolpec $\hat{\rho}$ konstantni je povprečna izguba za vseh 5 primerov, kjer so verjetnosti dobička za posamezno ročico konstante skozi celoten eksperiment. V stolpcu $\hat{\rho}$ dinamični je povprečna izguba za 5 različnih primerov, kjer se verjetnosti za dobiček spreminjajo. Zadnji stolpec $\hat{\rho}$ pa predstavlja skupno povprečno izgubo za konstantne in dinamične verjetnosti dobičkov. Ker je cilj minimizacija izgube, so zaradi tega nižje vrednosti boljše.

algoritem	$\hat{\rho}$ **konstantni**	$\hat{\rho}$ **dinamični**	$\hat{\rho}$
ϵ-Greedy (ϵ=0.2)	5.2%	49.4%	39.6%
UCB	14%	22.6%	22.7%
RPM	6,9%	22.6%	20.6%

Tabela 4.1: Primerjava uspešnosti MAB metod pri Jackpot problemu

Iz rezultatov je razvidno, da je izguba bistveno manjša pri primerih, ki

algoritem	$\hat{\rho}$ **konstantni**	$\hat{\rho}$ **dinamični**	$\hat{\rho}$
RPM	6,9%	22.6%	20.6%
RPM z drsečim oknom	10,6%	15,8%	14.3%

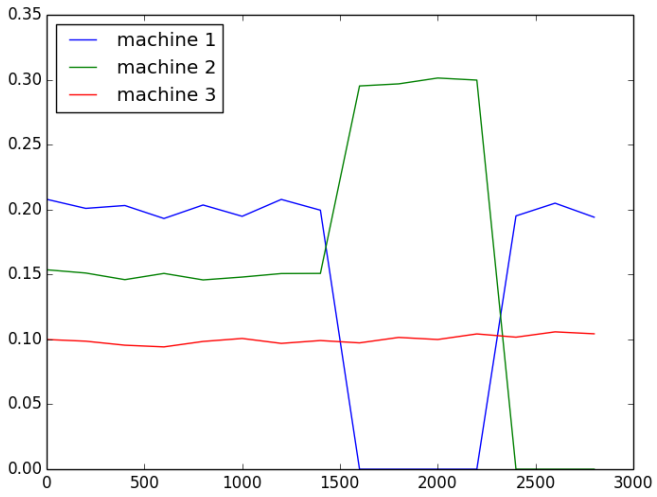
Tabela 4.2: Primerjava uspešnosti metode RPM in RPM z drsečim oknom.

imajo konstantno verjetnost dobička. To je seveda pričakovano, saj gre za preprostejši problem. Prav tako vidimo, da na konstantnih primerih najboljše rezultate dosežemo z metodo ϵ-Greedy in RPM, medtem ko na dinamičnih primerih razlike med RPM in UCB ni. Razlog, da skupna povprečna izguba ($\hat{\rho}$) ni aritmetična sredina med konstantno in dinamično povprečno izgubo je ta, da se je na dinamičnih primerih dalo zaslužiti (in izgubiti) večji dobiček. Vse vrednosti $\hat{\rho}$ konstantni, $\hat{\rho}$ dinamični in $\hat{\rho}$ so bile izračunane po enačbi 3.1 tako, da smo dobiček, ki smo ga izmerili eksperimentalno, primerjali z matematično najvišjim možnim dobičkom.

Zaradi tega smo metodo RPM prilagodili s pomočjo drsečega okna. To pomeni, da pri izbiri optimalne ročice nismo upoštevali celotne zgodovine predhodnih potegov. Velikost drsečega okna smo določili eksperimentalno. Najboljši rezultat smo dosegli pri velikosti okna, ki je imel dolžino 20% celotnega števila potegov. Takšno dolžino smo si lahko privoščili, saj smo že vnaprej vedeli, koliko potegov imamo na voljo za posamezen primer. Iz Tabele 4.2 je razvidno, da smo pri metodi RPM z drsečim oknom nekoliko poslabšali rezultate pri konstantnih primerih, vendar smo občutno izboljšali rezultate pri primerih, kjer se verjetnost za dobiček dinamično spreminja. Kot smo opazili že prej, pa le-ti bolj vplivajo na skupni zaslužek in so zaradi tega pomembnejši, tako da smo skupno izgubo zmanjšali za več kot 6%. S to metodo smo torej osvojili več kot 85% od matematično izračunanega maksimalnega možnega dobička. Prav tako je bila ta metoda izbrana za najboljšo študentsko rešitev opisanega Jackpot problema.

4.3 Razlaga algoritma RPM na konkretnem primeru

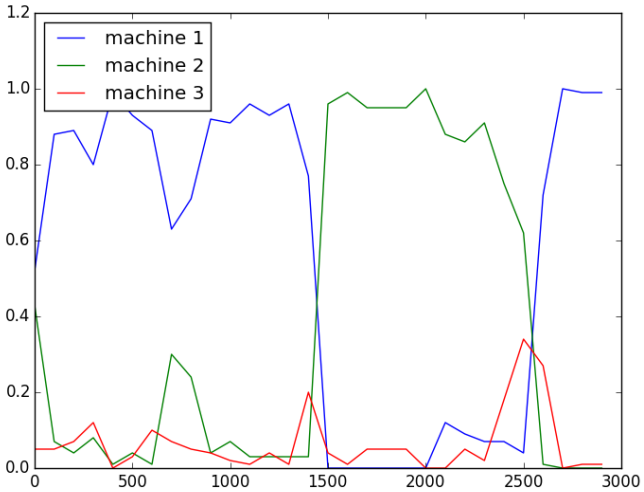
Oglejmo si še delovanje našega algoritma, ki se je izkazal kot najboljši na konkretnem primeru, ki je prikazan na Sliki 4.2. Na tej sliki je prikazana verjetnost posamezne ročice za izplačilo nagrade pri posameznem potegu (na voljo imamo 3000 potegov). Gre za primer, kjer se verjetnost izplačila nagrade spreminja skozi čas in v vsakem koraku algoritem potegne eno izmed treh ročic. Na začetku ima najvišjo verjetnost izplačila ročica1(20 %). Med 1400-tim in 2400-tim potegom ima najvišjo verjetnost izplačila ročica 2 in ob koncu ponovno ročica 1. Ročica 3 ima vseskozi konstantno vrednost izplačila nagrade (10%). Idealen algoritem bi tako na začetku vedno potegnil prvo ročico, v vmesnem obdobju drugo in na koncu ponovno prvo.



Slika 4.2: Konkreten primer, kjer se verjetnosti ročic za dobitek spreminjajo skozi čas

Na Sliki 4.3 vidimo, kolikšna je verjetnost, da bo naš algoritem izbral posamezno ročico od vsakem potegu. Vidimo lahko, da so te verjetnosti močno korelirane z verjetnostmi nagrad posameznih ročic. Na začetku je

največja verjetnost, da bo algoritem izbral prvo ročico, potem drugo in ob koncu ponovno prvo ročico. Vidimo lahko, da se verjetnost za poteg tretje ročice dvigne zgolj okrog 1400 in 2400 potega. To je takrat, ko pride do spremembe verjetnosti dobitka pri prvi in drugi ročici.



Slika 4.3: Spreminjanje verjetnosti za izbiro ročice glede na RPM algoritem

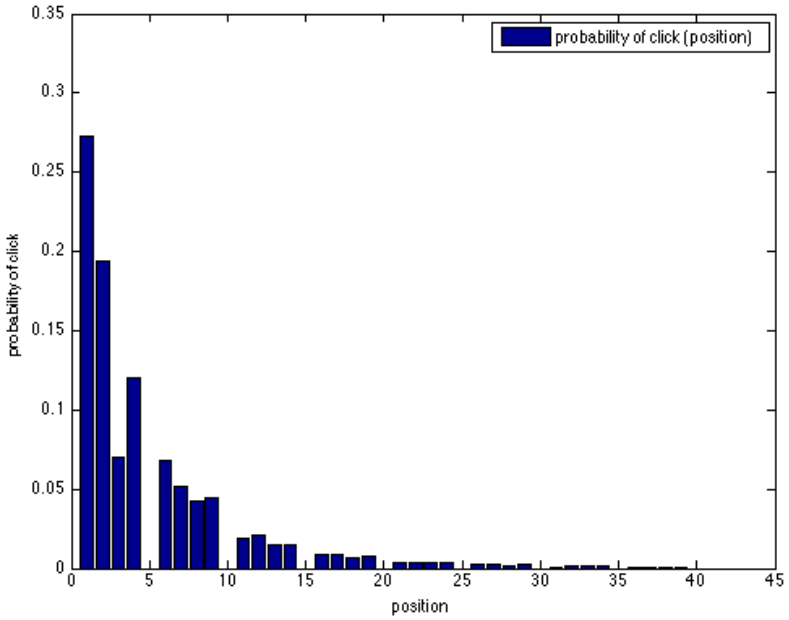
Poglavje 5

Uporaba MAB pristopa na Zemanta Streams priporočilnem sistemu

5.1 Motivacija

Slika 5.1 prikazuje porazdelitev uporabniških klikov glede na položaj priporočene novice. Kot lahko vidimo se v primeru klika na priporočilo, le-ta zgodi na prvem položaju v več kot četrtini primerov. Kar 60% vseh klikov pa se zgodi na prvih štirih pozicijah. To pomeni, da uporabniki zaupajo priporočilom sistema, ki temelji na podobnosti. Zaradi tega je zelo pomembno, da so novice, ki so uporabnikom zanimive in imajo najvišjo verjetnost klika, pozicionirane čim višje. V idealnem primeru, bi novice, ki so sorodne in imajo najvišjo stopnjo odzivnosti, vedno priporočali na najboljšem položaju.

Poglejmo si konkreten primer toka priporočil. Novica z naslovom *"Yahoo's mobile video streaming app is now on Android too"* (ker so bila priporočila aplicirana na spletni strani v angleškem jeziku, bomo naslove novic ohranili v izvirniku), ki je bila objavljena 30.4.2014 na spletni strani [18] in je bila prikazana 545-krat (tolikokrat je bila prikazana na mobilni spletni strani do 5.5.2014). Ker so bile priporočene novice vedno razvrščene glede na



Slika 5.1: Porazdelitev uporabniških klikov glede na položaj priporočene novice

podobnost, je bil vrstni red priporočil vseskozi enak. Tako je bila na prvem mestu priporočena novica z naslovom *"Yahoo may build its own YouTube"*, na drugem novica z naslovom *"Minecon live: Watch Mojang's Minecraft event right here"* in na četrtem mestu novica *"Pandora finally comes to Chromecast via Android & iPhone apps"*.

Rezultati, ki so prikazani v Tabeli 5.1 so v nasprotju s pričakovanjem. Glede na porazdelitev klikov, ki je prikazana na Sliki 5.1, bi pričakovali, da bo največ klikov dobila novica priporočena na prvem mestu. Očitno je bila novica na četrtem mestu najboljša, saj je imela višjo stopnjo odzivnosti, kot novici na prvem in drugem mestu, kljub temu, da je imela veliko slabše možnosti (verjetnost klika na četrti poziciji je veliko nižja). Očitno je to slabost obstoječega priporočilnega sistema, saj bi v takšnem primeru bilo smiselno, da bi novico, ki je sorodna, a ne najbolj podobna brani novici,

položaj	naslov	št. klikov	CTR
1	Yahoo may build its own YouTube	8	1.5%
2	Minecon live: Watch Mojang's Min...	4	0.7%
4	Pandora finally comes to Chromec...	11	2%

Tabela 5.1: Stopnja odzivnosti treh novic, ki so bile priporočene 545-krat pod novico z naslovom *"Yahoo's mobile video streaming app is now on Android too"*

priporočili višje. Obstoječ priporočilni sistem bi radi nadgradili tako, da bo sposoben sprotno (ang. online) prerazporejati vrstni red sorodnih novic, glede na verjetnost klika nanj. Na ta način bi uporabniku povečali zaupanje v priporočilni sistem, saj bi bile bolj zanimive in aktualne novice prikazane višje.

5.2 Zasnova rešitve

Konkreten tok priporočil, ki smo ga prikazali v prejšnjem podpoglavju 5.1, bi lahko izboljšali tako, da bi denimo teh 545 priporočil izkoristili kot fazo raziskovanja in nato spremenili vrstni red glede na vrednosti stopnje odzivnosti, ki so prikazane v Tabeli 5.1. To bi bila najpreprostejša rešitev, vendar bi na ta način diskriminirali članke, ki bi nastali po fazi raziskovanja in tiste novice, ki bi morda postale boljše v kasnejši fazi. Zaradi tega so MAB pristop naravna izbira za optimalno razvrščanje sorodnih novic. S pomočjo MAB algoritma bomo ocenili stopnjo odzivnosti vsakega članka, ki ga želimo priporočiti in nato bomo vse članke padajoče razvrstili glede na pričakovano stopnjo odzivnosti.

V tem poglavju bomo predstavili različne metode za ocenjevanje stopnje odzivnosti novic. Metode se ne bodo osredotočale na izbiro najboljšega MAB algoritma, temveč je temeljni izziv ta, kako uporabiti predhodne uporabniške odzive tako, da bomo najbolj obetavne novice predlagali čim višje. Vse spodaj naštete metode bodo uporabljale RPM algoritem, ker se je ta pristop

izkazal kot najobetavnejši pri Jackpot problemu v Poglavju 4. Ta pristop ocenjuje pričakovano stopnjo odzivnosti kot naključno vrednost porazdeljeno glede na beta distribucijo. Beta distribucija je družina zveznih verjetnostnih funkcij, definiranih na intervalu $[0, 1]$, ki jih definirata vhodna parametra α in β. V našem primeru bo parameter α predstavljal število klikov na neko priporočilo in parameter β razliko med številom priporočil in številom klikov (5.1).

$$\begin{aligned} \alpha &= \text{št. klikov} \\ \beta &= \text{št. prikazov} - \text{št. klikov} \end{aligned} \tag{5.1}$$

Metode, opisane v tem poglavju, se bodo razlikovale glede na način in kontekst, na katerega bomo izračunali število klikov in priporočil neke novice. Omenjene metode so bile predstavljene tudi v [19], ki smo ga objavili na delavnici NewsKDD v sklopu konference KDD 2014. Programska koda, ki smo jo uporabljali za razvoj in testiranje opisanih rešitev, je objavljena na repozitoriju programske kode [20].

5.3 Prilagoditev stopnje odzivnosti glede na položaj klika

Iz Slike 5.1 je razvidno, da je najbolj verjetno, da bo priporočilo uporabnika kliknjeno na prvi poziciji. V primeru, da uporabnik klikne priporočilo na nižji poziciji, je takšen klik več vreden od tistega na prvi poziciji. Zaradi tega smo izračunali pozicijski faktor (f_p), ki je razmerje stopenj odzivnosti na prvi poziciji in pozicijo, za katero želimo izračunati ta faktor (5.2).

$$f_p = \frac{\mathrm{CTR}_1}{\mathrm{CTR}_p} \tag{5.2}$$

Tabela 5.2 vsebuje statistiko za prvih 10 položajev in pripadajoče pozicijske faktorje, ki ustrezajo enačbi (5.2). Razlog, zakaj je stopnja odzivnosti na

p	$\mathbf{CTR}_p$	f_p
1	1.085%	1
2	0.766%	1.4164
3	0.278%	3.9029
4	0.477%	2.2746
5	0.0%	3.3516
6	0.245%	4.4286
7	0.191%	5.6806
8	0.160%	6.7812
9	0.167%	6.4970
10	0.0%	6.8745

Tabela 5.2: Povprečen pozicijski faktor za priporočilni sistem, ki ga bomo uporabljali za vrednotenje naših metod.

vsaki peti poziciji 0%, je ta, da so ta priporočila bila oglasna in odzivi niso bili zabeleženi v dnevniški datoteki, do katere smo imeli dostop. Posledično smo stopnjo odzivnosti za vsako peto pozicijo izračunali kot povprečje položaja pred in položaja za tem položajem.

S pomočjo pozicijskega faktorja smo obtežili vsak uporabniški klik. Na ta način smo izenačili klike med seboj, ne glede na to na katerem položaju se je klik zgodil. Za konkreten primer (opisan v Poglavju 5.1) je izračun za prilagojeno število klikov narejen v Tabeli 5.3. Zavedati pa se moramo, da je to zgolj hipotetičen izračun, saj bi se vrstni red priporočil sproti spreminjal v primeru uporabe MAB algoritma. Zaradi tega vhodni parameter α izračunamo kot vsoto produktov vseh klikov, pomnoženih s pozicijskim faktorjem, na katerem se je klik zgodil (5.2).

$$\alpha = \sum_{p=1}^{\text{max pos.}} f_p \cdot \text{n\_clicks}_p \tag{5.3}$$

Rezultati so pokazali, da je uporaba pozicijskega faktorja zelo smiselna, zaradi česar vse nadaljnje metoda uporabljajo to metodo, kljub temu da tega

polozaj	**naslov**	**št. klikov**	**prilago. št. klikov**
1	Yahoo may build its own...	8	8
2	Minecon live: Watch Moj...	4	5.66656
4	Pandora finally comes to...	11	25.0206

Tabela 5.3: Prilagojeno število klikov treh novic, ki so bile priporočene 545-krat pod novico z naslovom *"Yahoo's mobile video streaming app is now on Android too"*

ne bomo eksplicitno omenjali. V nadaljevanju št. klikov pomeni število klikov, ki so pomnoženi z ustreznim pozicijskim faktorjem. Parametru β nismo odšteli prilagojenega števila klikov, ampak samo število klikov, saj bi v tem primeru teoretično lahko bila vrednost β manjša od 0, kar pa ni dovoljena vrednost pri beta funkciji.

5.4 Priporočanje na podlagi globalne stopnje odzivnosti

Prva in hkrati najpreprostejša metoda, ki smo jo uporabili za izračun vhodnih parametrov α in β, je bila globalna stopnja odzivnosti. To pomeni, da pri merjenju odziva (število klikov in prikazov) nismo upoštevali nobenega konteksta. Vrednost parametra α je tako bila število vseh klikov priporočene novice do trenutka priporočila. Ne zanima nas torej kontekst, v katerem priporočilnem toku se je klik zgodil, ampak nas zanimajo vsi kliki na priporočilo te novice v vseh priporočilnih tokovih. Vhodni parameter β smo izračunali kot vsoto vseh priporočil te novice in odšteli število klikov.

Na konkretnem primeru, ki ga vseskozi omenjamo, bi tako preverili, kolikokrat je bila novica z naslovom *"Yahoo may build its own YouTube"* priporočena in koliko klikov je imela v celotnem sistemu. S to metodo ne opazujemo uspešnosti te priporočene novice zgolj pod člankom z naslovom *"Yahoo's mobile video streaming app is now on Android too"*, ampak pod vsemi

članki, pri katerih je bila ta novica kadarkoli prikazana. Enako smo naredili tudi za ostale novice in te vrednosti uporabili kot vhodne parametre RPM algoritma. Algoritem mora hraniti statistike za vse članke, ki so bili kadarkoli predlagani. Če je n število člankov, je potem prostorska zahtevnost te metode $O(n)$.

Rezultate te metode smo primerjali tudi s požrešno metodo, da smo se prepričali o smiselnosti uporabe RPM algoritma. V nasprotju z RPM algoritmom, kjer je vrstni red odvisen tudi od naključja (še posebno, ko imajo priporočene novice zelo podobno statistiko uspešnosti), požrešna metoda razvršča novice po njihovi stopnji odzivnosti ($\frac{\text{št. klikov}}{\text{št. prikazov}}$).

5.5 Priporočanje na podlagi stopnje odzivnosti na nivoju priporočilnega toka

V nasprotju s prejšnjim pristopom nas sedaj zanima število klikov in število prikazov priporočene novice v kontekstu trenutnega priporočilnega toka. Vrednost parametra α je tako št. klikov na priporočeno novico pod trenutno prikazanim člankom in β št. prikazov. Statistiko prikazov in klikov priporočene novice v drugih priporočilnih tokovih (pod drugimi članki) v tem primeru ignoriramo.

Statistika primera, na katerega se vseskozi referenciramo in je prikazana v Tabeli 5.1, je merjena skozi prizmo te metode. Oziramo se zgolj na en priporočilni tok in merimo odzive priporočenih novic pod enim člankom. Takšno statistiko moramo voditi za vse priporočene novice v vseh priporočilnih tokovih ločeno. Če imamo n člankov, potem imamo tudi n priporočilnih tokov. V najslabšem primeru bi tako bila prostorska zahtevnost $O(n^2)$. To bi bilo res le v primeru, če bi statično inicializirali podatkovno strukturo. Ker pa lahko v vsakem priporočilnem toku hranimo statistiko odzivov le za tiste novice, ki so dejansko bile priporočene (običajno manj kot 100 različnih novic), je v praksi prostorska zahtevnost $O(kn)$, kjer je k povprečno število priporočenih novic, ki so bile priporočene pod posamezno novico.

Specifičnost te metode je, da ob vsakem novem priporočilnem toku (ko imamo opravka z novo novico v sistemu) nimamo statistike za nobeno novico, ki jo želimo priporočiti. Intuitivno bi ta problem lahko rešili tako, da bi v takšni situaciji priporočali na podlagi globalne stopnje odzivnosti. To pomeni, da v primeru, ko želimo sortirati sorodne novice glede na trenutno brano novico in sorodna novica še nima dovolj prikazov in odzivov, uporabimo globalno stopnjo odzivnosti namesto stopnje odzivnosti, značilne za ta priporočilni tok. Kriterij, na podlagi katerega smo se odločili, ali bomo uporabili globalni CTR ali specifičnega, je bila standardna deviacija beta distribucije, ki jo določata parametra α in β. Če je bila standardna deviacija manjša od neke meje, smo uporabili stopnjo odzivnosti specifično na nivoju priporočilnega toka, sicer pa globalno stopnjo odzivnosti. Standardna deviacija je visoka takrat, kadar nimamo veliko predhodnih odzivov in doseže maksimum natanko tedaj, ko še nimamo nobenega odziva. Izkazalo se je, da ta kombinirana metoda ni bila uspešna, je pa predstavljala osnovno idejo, na kateri temelji hierarhična aproksimacija stopnje odzivnosti, ki je opisana v Poglavju 5.6.2.

5.6 Aproksimacija stopnje odzivnosti z vsebinsko podobnimi priporočilnimi tokovi

Ideja za metodi, ki ju bomo opisali v tem podpoglavju, temelji na obstoječe metodi strojnega učenja. To je k-NN (ang. k-nearest neighbours), ki je metoda s področja nadzorovanega strojnega učenja (ang. supervised learning). Ta metoda napoveduje ciljne oznake s pomočjo najsorodnejših primerov iz učne množice glede na primer, ki mu želimo napovedati vrednost. To idejo bi radi uporabili za aproksimacjo stopnje odzivnosti v primeru, ko o neki priporočeni novici ne bomo imeli dovolj predhodnih informacij v kontekstu trenutnega priporočilnega toka. Radi bi torej uporabili odzive priporočene novice v priporočilnih tokovih, ki so vsebinsko sorodni trenutnemu. Namen te metode je predvsem ta, da bi napovedali stopnjo odzivnosti novicam, o

katerih ne vemo nič ali o njih vemo malo in posledično skrajšali fazo raziskovanja.

5.6.1 Vektorska predstavitev novic

Za primerjavo novic glede na vsebino je bilo potrebno vsako novico pretvoriti v vektorski prostor, ki nam je omogočal, da smo podobnost med novicami lahko numerično izmerili. Pred tem je bilo potrebno vsebino novic ustrezno obdelati, tako da smo iz vsake novice pridobili čim več informacije. Prvi korak pri obdelavi novic je bilo razbitje vsebine na posamezne besede (ang. tokenization) [21]. Potem smo izločili vse besede, ki niso informativne (npr. vezniki, osebni zaimki, časovni zaimki), ker se pojavljajo v praktično vseh besedilih. Naslednji korak pri obdelavi besed pa je bilo korenjenje besed. Zaradi tega, ker so novice, ki jih uporabljamo, v angleščini je tudi naslednji primer korenjenja besed na primeru angleških besed avto videti (car, see). Besede car, cars, car's, cars' so štiri različne besede, ki pa jih po procesu korenjenja besed pretvorimo v besedo car. Temu procesu v angleščini pravimo stemming [21]. Na drugi strani pa korenjenju, ki temelji na besediščni in morfološki analizi besed pravimo lematizacija [21]. Primer takšnega korenjenja bi bila pretvorba besede saw $\rightarrow$ see (v primeru glagola v pretekliku) ali saw $\rightarrow$ saw (v primeru samostalniške oblike besede saw).

Nabor vseh besed iz vseh novic, ki smo jih dobili po zgoraj opisanem postopku, nam je predstavljal vrečo besed. Vektor, ki bi predstavljal določeno novico, bi bil tako lahko sestavljen iz števila pojavitev vsake besede. Takšna predstavitev pa je slaba, saj na število pojavitev določene besede vpliva tudi dolžina dokumenta in vse besede enači med seboj. Ta problem pa rešimo tako, da število pojavitev besede v neki novici pomnožimo z utežjo idf_t (inverzna frekvenca dokumenta) [21].

$$\begin{aligned}
&\text{tf-idf}_{\text{t,d}} = \text{tf}_{\text{t,d}} \times \text{idf}_t \\
&\text{idf}_t = \log \frac{N}{\text{df}_t} \\
&\text{tf}_{\text{t,d}} \text{ ... št. ponovitev besede t v dokumentu d} \\
&\text{N ... št. vseh dokumentov/novic} \\
&\text{df}_t \text{... št. različnih dokumentov, kjer se termin t pojavi}
\end{aligned} \tag{5.4}$$

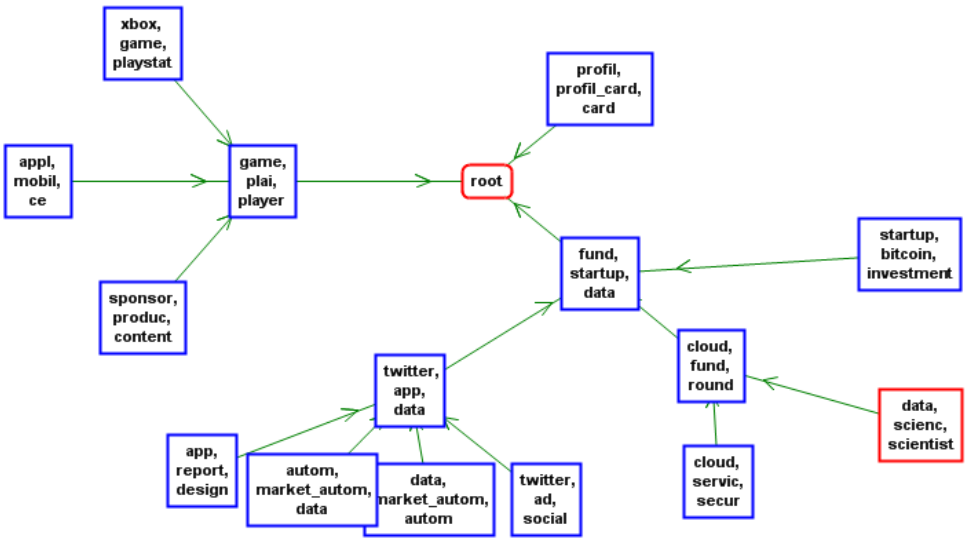
Glede na enačbo (5.4) bodo imele najvišjo vrednost tiste besede, ki se velikokrat pojavijo v nekem dokumentu, v drugih dokumentih pa je ta beseda zelo redka. Nizke vrednosti bodo imele tiste besede, ki se redko (ali nikoli) ne pojavijo v dokumentu oziroma je ta beseda zelo pogosta v vseh dokumentih. Za izračun razdalje med dvema vektorjema novic bi sedaj lahko uporabili katerokoli mero podobnosti, vendar se v problemski domeni besedil običajno uporablja kosinusna podobnost [21], ki je definirana z enačbo (2.1).

Naša raziskava se ni ukvarjala s problemom skalabilnosti. Posledično nam ni bilo potrebno optimizirati iskanja sorodnih novic. Vseeno pa bi na tem mestu omenili, da bi v primeru produkcijske implementacije bilo vredno implementirati metodo, ki ne zahteva primerjave vseh novic med seboj. Za rešitev tega problema, bi lahko uporabili LSH [22] pristop, ki omogoča iskanje sorodnih novic v linearnem času. Tak pristop sicer ni popolnoma točen, vendar je prihranek časa tolikšen, da bi ga v primeru velikega števila novic ($> 1\ 000\ 000$) bilo smiselno uporabiti.

5.6.2 Hierarhična aproksimacija stopnje odzivnosti

Ob koncu Poglavja 5.5 smo opisali kombiniran pristop med priporočanjem na podlagi stopnje odzivnosti v kontekstu priporočilnega toka in globalno stopnjo odzivnosti. Ta metoda je delovala tako, da smo uporabili globalno stopnjo odzivnosti v primeru, ko stopnja odzivnosti v kontekstu trenutnega priporočilnega toka ni bila zaupanja vredna (standardna deviacija beta distri-

bucije je bila višja od empirično postavljene meje). Pri tem pristopu smo to idejo implementirali hierarhično. Ko o priporočeni novici nismo imeli dovolj predhodnih informacij smo stopnjo odzivnosti aproksimirali glede na odzive te novice v vsebinsko podobnih priporočilnih tokovih. Če je bila nedoločenost še vedno prevelika, smo uporabili odzive iz še več sorodnih priporočilnih tokov. Najvišji nivo hierarhične aproksimacije pa je uporaba vseh priporočilnih tokov, kar pomeni, da uporabljamo enake podatke kot pri metodi, ki uporablja globalno stopnjo odzivnosti.



Slika 5.2: Hierarhična struktura člankov

Slika 5.2 prikazuje hierarhično strukturo novic iz učne množice, ki smo jo izdelali s pomočjo polavtomatskega urejevalnika ontologij OntoGen [23]. Iz te strukture lahko s pomočjo ključnih besed vidimo tudi vsebino novic, ki jih uporabniki lahko prebirajo na spletni strani. V korenu hierarhične strukture (root) so vsebovane vse novice. Te novice bi na najvišjem nivoju lahko razdelili na novice o igrah, start-up podjetjih in investicijah ter tretjo nehomogeno skupino. Ta skupina ima sicer ključne besede *profil* in *profile_card*, vendar bi to skupino najbolje opisali kot novice, ki niso neposredno povezane s start-up

podjetji ali igrami.

Hierarhično strukturo novic, ki je prikazana na Sliki 5.2 smo izvozili iz orodja OntoGen. Za vsako skupino iz končnega vozlišča smo izračunali centroid, ki je povprečje vseh novic, ki pripadajo izbrani skupini. Algoritem je torej deloval tako, da je najprej uvrstil trenutni priporočilni tok v ustrezno skupino. To je naredil tako, da je izračunal razdaljo od trenutno brane novice do vseh centroidov. Priporočilni tok pripada tisti skupini, do katere je razdalja najmanjša. Nato je algoritem izračunal za vsako novico, ki jo želimo priporočiti, varianco glede na informacije, ki jih imamo o predhodnih priporočilih te novice v kontekstu trenutnega priporočilnem toku. Če je varianca manjša od meje, smo uporabili vse predhodne odzive te priporočene novice v vseh priporočilnih tokovih, ki so v isti skupini. V primeru, da je varianca še vedno večja od meje smo postopek ponovili tako, da smo pridobili podatke iz skupine višje glede na hierarhijo. Ta postopek smo ponavljali, dokler nismo imeli toliko podatkov o eni novici, da je bila standardna deviacija nižja od prej določene meje oziroma nismo dosegli začetnega vozlišča (root).

Oglejmo si delovanje tega primera na hipotetičnem primeru. Pri tem hipotetičnem primeru bomo uporabljali mejo zaupanja 0.0001. Denimo, da je neka priporočena novica A bila v priporočilnem toku x do tega trenutka priporočena 10-krat. Od tega je samo en uporabnik kliknil na to novico. Vhodna parametra imata torej vrednost $\alpha = 1$ in $\beta = 10 - 1 = 9$. Beta distribucija s takšnima vhodnima parametroma ima srednjo vrednost (stopnjo zaupanja) 0.1 in varianco 0.008. Ker je varianca večja od meje, moramo priporočilni tok x uvrstiti v ustrezno skupino priporočilnih tokov. Denimo, da priporočilni tok x spada v skupino s ključnimi besedami app, report in design. V tej skupini je do tega trenutka bila novica A priporočena 150-krat in kliknjena 3-krat. Varianca je 0.00013, kar je še vedno večje od meje. Zaradi tega poiščemo statistiko odzivov za priporočeno novico A tudi v skupini s ključnimi besedami *twitter, app, data*, kar pomeni en nivo višje od prejšnje skupine. V tej skupini pa je bila novica A priporočena 750-krat in kliknjena 10-krat. Varianca je manjša od meje in zaradi tega, novici A aproksimiramo

stopnjo odzivnosti z vhodnimi parametri $\alpha = 10$ in $\beta = 750 - 10 = 740$.

Kar se tiče računske in prostorske zahtevnosti je ta algoritem asimptotično gledano enako zahteven. Je pa računsko veliko bolj zahteven z zornega kota, da je potrebno ob vsaki novi novici vsebino članka pretvoriti v vektorski prostor in izračunati razdalje do vseh centroidov. Na prostorsko zahtevnost vpliva tudi dejstvo, da je potrebno hraniti tudi statistiko odzivov vseh priporočenih novic za vsako skupino. To pa ne poveča asimptotične vrednosti prostorske zahtevnosti (poveča se zgolj za konstanto). Tega podatka ni potrebno hraniti, saj bi se ga dalo tudi izračunati v vsakem koraku, vendar bi s tem povečali računsko zahtevnost algoritma.

5.6.3 Aproksimacija stopnje odzivnosti s pomočjo k-najbližjih priporočilnih tokov

Pri prejšnji metodi smo iz množice člankov izdelali statično hierarhično strukturo, na podlagi katere smo določali sorodnost člankov. S pomočjo centroidov, smo vsaki novici določili v katero skupino sodi in novice, ki so bile v isti skupini, so bile sorodne. Morda je bil takšen pogled na iskanje sorodnih novic preveč splošen in neprilagodljiv. Zaradi tega ta metoda deluje tako, da vsakemu priporočilnemu toku poiščemo k-najbližjih priporočilnih tokov in s pomočjo statistike odzivov iz teh priporočilnih tokov aproksimiramo stopnjo odzivnosti. Pristop smo preizkusili na dva načina. Prvi način je bil, da smo preizkusili različne meje in v primeru prevelike variance beta distribucije uporabili statistiko iz sorodnih priporočilnih tokov. Drug način pa je bil, da smo v vsakem primeru aproksimirali ciljno stopnjo odzivnosti s pomočjo sorodnih podatkovnih tokov, ne glede na to, koliko informacij o novici, ki jo želimo priporočiti, že imamo.

Podobno kot v prejšnjem poglavju bomo delovanje tega algoritma opisali s pomočjo hipotetičnega primera. Denimo, da želimo izračunati stopnjo odzivnosti za priporočeno novico A v priporočilnem toku x in $k = 50$. V primeru, da uporabljamo drug način, nam ni potrebno preverjati, ali je varianca beta distribucije, ki uporablja vhodne podatke, ki temeljijo na stopnji

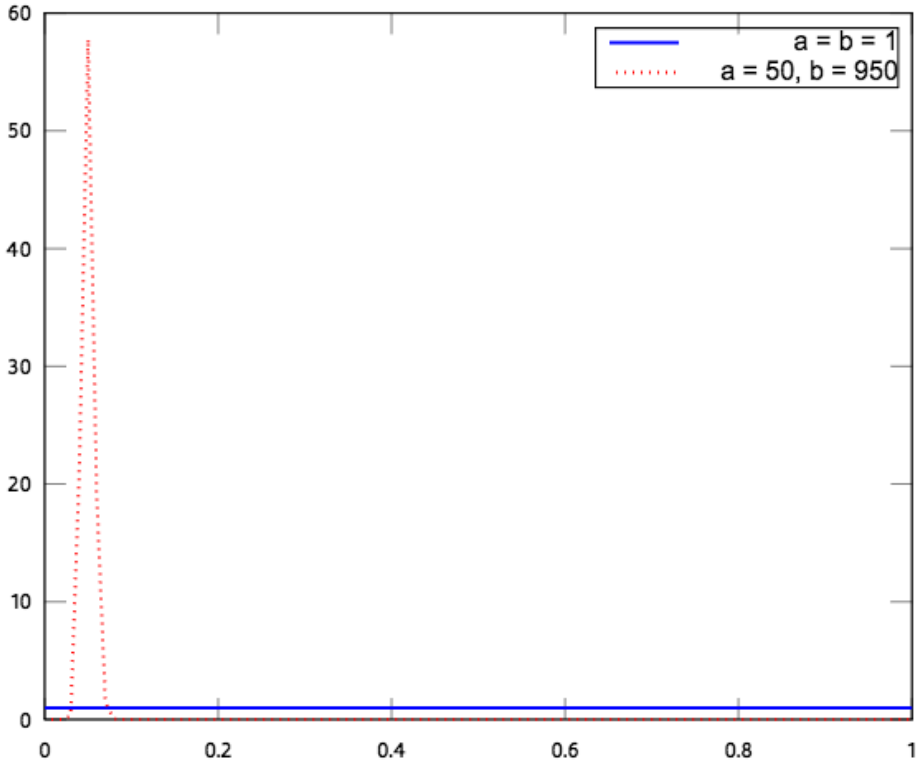
odzivnosti iz trenutnega priporočilnega toka večja od meje ali ne. Zaradi tega je prvi korak ta, da poiščemo 50 najsorodnejših priporočilnih tokov in vsa priporočila in odzive priporočene novice A v vseh teh 50 priporočilnih tokovih. Vrednost vhodnega parametra α je tako vsota vseh klikov na novico A v trenutnem priporočilnem toku in vseh 50 najsorodnejših priporočilnih tokovih. Vrednost parametra β pa je vsota vseh prikazov v istih priporočilnih tokovih minus vsota vseh klikov.

Pri tej metodi velja omeniti dve stvari, na kateri je potrebno biti pozoren pri implementaciji. Prvič, potrebno je izračunati k-najbližjih sosedov na začetku in najbližje sosede shraniti. Izračun najbližjih sosedov je računsko zahteven in bi bila metoda časovno prezahtevna, če bi od vsaki aproksimaciji α in β računali najbližje sosede trenutnemu priporočilnemu toku. Druga težava, ki izhaja iz rešitve prve pa je, da je potrebno seznam najbližjih sosedov vsakega priporočilnega toka osveževati ob vsaki novi novici. Sami smo ta problem rešili s pomočjo Python knjižnice Gensim [24], ki je namenjena obdelavi naravnega jezika. Ta knjižnica nam je omogočila izdelavo indeksa novic, ki je v konstantnem času vračal k-najbolj sorodnih novic izbrani novici. Ob vsaki novi novici pa smo ta index osvežili tako, da smo dodali nov članek in njemu pripadajočih k-najsorodnejših novic. Poleg tega pa je bilo potrebno osvežiti tudi vse tiste novice, ki jim je ta nova novica bolj sorodna kot katera izmed njihovih obstoječih k-najsorodnejših novic.

5.7 Ocenjevanje začetnih vrednosti vhodnih parametrov α in β

Do sedaj še nismo omenili problema izbora začetnih vrednosti vhodnih parametrov α in β. Edini pogoj, ki smo ga omenili je, da morata bili parametra večja od nič. Najpreprostejša inicializacija je takšna, da imata parametra α in β začetno vrednost ena. S tem dosežemo, da je vrednost parametrov vedno večja od nič, tudi pri novici, ki še ni bila nikoli kliknjena oz. nikoli prikazana. Ko je $\alpha = 1$ in $\beta = 1$, je vrednost naključne spremenljivke, ki je

porazdeljena glede na beta distribucijo, enakomerno porazdeljena na intervalu $(0, 1)$. Ob predpostavki, da praktično nobena priporočena novica nima stopnje odzivnosti višje od 5%, ima vsaka nova novica zelo veliko možnosti, da bo priporočena višje od novice, ki ima stopnjo odzivnosti 5%. Iz Slike 5.3



Slika 5.3: Funkcija gostote verjetnosti (ang. probability density function) beta porazdelitev z vhodnimi parametri $\alpha = 1, \beta = 1$ in $\alpha = 50, \beta = 950$

lahko vidimo, da bo naključna spremenljivka, ki je porazdeljena glede na beta distribucijo z vhodnimi parametri $\alpha = 50$ in $\beta = 950$, zavzela vrednost okrog 0.05 z veliko gotovostjo. Po drugi strani pa bo druga naključna spremenljivka porazdeljena enakomerno na intervalu (0,1). Zaradi tega, bo v večini primerov ($> 90\%$) nova novica priporočena višje od obstoječe (zelo dobre) novice. Cilj MAB algoritmov je najti optimalno porazdelitev med raziskovanjem in izkoriščanjem. S takšno začetno porazdelitvijo pa se zdi, da so nove

novice preveč privilegirane. Predhodne metode, ki smo jih že opisali, smo testirali tako, da smo preizkusili različne začetne vrednosti in opazovali, s kakšnimi začetnimi vrednostmi dosežemo najboljše rezultate. Pri tej metodi bi radi določili najbolj optimalni začetni vrednosti za vhodna parametra α in β glede na dane stopnje odzivnosti.

Denimo, da imamo n novic in njihove pripadajoče stopnje odzivnosti so $(X_1, X_2, ...X_n)$. Izračunati želimo verjetnost za poljubni vrednosti parametrov α in β ob znanih stopnjah odzivnosti $P(\alpha, \beta \mid X_1, X_2, ...X_n)$. S pomočjo Bayesovega teorema lahko to verjetnost zapišemo tudi drugače (enačba (5.5)).

$$P(\alpha, \beta \mid X_1, X_2, ...X_n) = \frac{P(X_1, X_2, ...X_n \mid \alpha, \beta) \times P(\alpha, \beta)}{P(X_1, X_2, ...X_n)} \tag{5.5}$$

Predpostavili bomo, da so stopnje odzivnosti $(X_1, X_2, ...X_n)$ neodvisne in uporabili Naivnega Bayesa [25] v enačbi (5.6).

$$P(\alpha, \beta \mid X_1, X_2, ...X_n) \propto P(\alpha, \beta) \prod_{i=1}^{n} P(X_i \mid \alpha, \beta) \tag{5.6}$$

Glede na to, da verjetnost vedno zasede prostor med 0 in 1, je produkt takšnih elementov zelo majhno število. Zaradi dejstva, da si ne želimo računskih napak, enačbo logaritmiramo (5.7).

$$\log P(\alpha, \beta \mid X_1, X_2, ...X_n) \propto P(\alpha, \beta) + \sum_{i=1}^{n} P(X_i \mid \alpha, \beta) \tag{5.7}$$

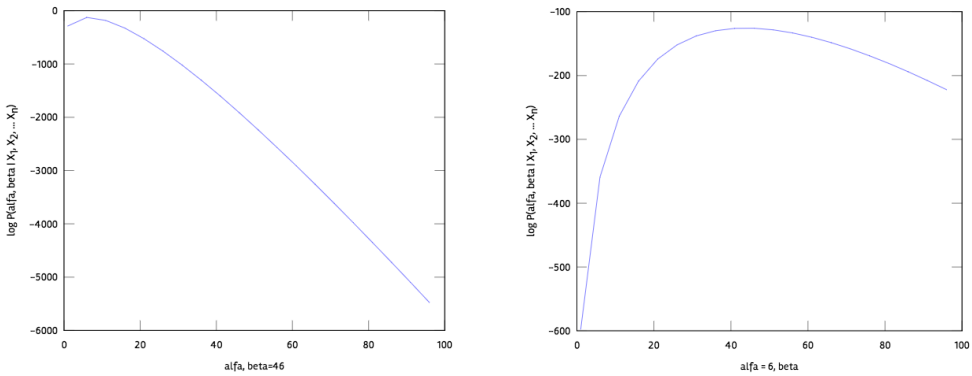
Verjetnosti $P(X_i \mid \alpha, \beta)$ ne moremo neposredno izračunati. Lahko pa izračunamo, kolikšna je verjetnost, da naključna spremenljivka X zavzame vrednost na intervalu (a, b] (5.8). V našem premeru je naključna spremenljivka porazdeljena glede na beta distribucija s parametroma α in β.

$$\begin{aligned} P(a < X \leq b) &= F_X(b) - F_X(a) \\ F_X(x) &= P(X \leq x) \end{aligned} \tag{5.8}$$

Interval (a,b] smo pri našem izračunu določili tako, da je bil $a = X_i$ in $b = a+$ 0.01. To pomeni, da je interval predstavljal 1% definicijskega območja beta distribucije, ki je določena na intervalu (0,1). Zaradi takšnega izračuna smo izgubili na točnosti izračuna verjetnosti, vendar bomo s spodnjim primerom pokazali, da ohranjamo trend. Prav tako lahko člen $P(\alpha, \beta)$ zanemarimo, saj bosta v nadaljevanju parametra α in β izbrana naključno in je to konstanta, ki je pri vseh členih enaka. Končna enačba, ki smo jo uporabili za izračun verjetnosti parametrov α in β ob znanih stopnjah odzivnosti je (5.9).

$$\log P(\alpha, \beta \mid X_1, X_2, ...X_n) \propto \sum_{i=1}^{n} (F_{\alpha,\beta}(X_i + 0.01) - F_{\alpha,\beta}(X_i)) \tag{5.9}$$

Denimo, da imamo sto novic, katerih stopnja odzivnosti je naključno porazdeljena glede na beta distribucijo, ki ima vrednost parametrov ($\alpha = 6$ in $\beta = 46$). Najprej smo izračunali vrednosti enačbe (5.9) za vrednosti $\alpha \in 1, 6, 11, ..., 91, 96$ in fiksno vrednost $\beta = 46$. Nato smo postopek ponovili, le da smo sedaj fiksirali $\alpha = 6$ in izračunali vrednost enačbe za različne vrednosti parametra β. Kot vidimo iz grafov 5.4a ter 5.4b, funkcija v obeh



(a) Vrednost (5.9) pri fiksnem $\beta = 46$ (b) Vrednost (5.9) pri fiksnem $\alpha = 6$

Slika 5.4: Vrednost enačbe (5.9) v odvisnosti od parametrov α in β

primerih doseže maksimum pri vrednostih $\alpha = 6, \beta = 46$, kar je pravilno. Smiselno je, da je verjetnost naključnih parametrov za dane stopnje odzivnosti največja pri vrednosti parametrov α in β, s katerimi so bile stopnje

odzivnosti ustvarjene. S tem primerom smo pokazali, da je bila naša matematična izpeljava smiselna. Če pa želimo poiskati najboljša parametra α in β, pa moramo poiskati maksimum funkcije (5.9). Ta problem smo rešili numerično s Powellovo metodo [26]. Ta metoda sicer ni najbolj učinkovita numerična metoda za iskanje maksimuma funkcije, vendar ima lastnost, da ne zahteva, da je funkcija odvedljiva in za delovanje ne potrebuje odvoda funkcije.

Zgoraj opisani postopek za določanje parametrov α in β ob naboru novic za katere poznamo njihovo stopnjo odzivnosti $(X_1, X_2, ..., X_n)$ smo uporabili na dva načina. Prvi in bolj preprost način je bil, da smo vzeli globalne stopnje odzivnosti vseh novic iz testne množice in izračunali parametra α_0 in β_0. Ti vrednosti sta predstavljali začetni vrednosti vseh novic, katerim smo potem prišteli število klikov oz. število prikazov neke novice v določenem priporočilnem toku. Če imamo neko priporočeno novico, ki je bil prikazana pod neko novico x-krat in kliknjena y-krat, potem je $\alpha = \alpha_0 + y$ in $\beta = \beta_0 + x - y$.

Prva metoda izračuna zelo splošni začetni vrednosti za vse priporočene novice hkrati. Z drugo metodo pa želimo biti bolj specifični, saj imajo priporočene novice lahko v različnih priporočilnih tokovih zelo različne stopnje odzivnosti. Prav tako se stopnje odzivnosti za posamezen priporočilni tok spreminjajo skozi čas. Zaradi tega druga metoda ob vsakem priporočilu izračuna novi začetni vrednosti α_0 in β_0. Ti začetni vrednosti sta izračunani glede na stopnje odzivnosti novic, ki jih želimo priporočiti glede na trenutni priporočilni tok. V tem primeru so stopnje odzivnosti $(X_1, X_2, ..., X_n)$ tistih novic, ki so sorodne trenutno prikazani novici. Končni vrednosti α in β smo izračunali enako kot v zgornjem primeru. Metodi se razlikujeta le v načinu izračuna začetnih vrednosti. Pri prvi metodi sta začetni vrednosti skozi celotno izvajanje algoritma enaki. V drugem primeru pa se izračunata novi začetni vrednosti za vsako priporočilo ločeno glede na stopnje odzivnosti novic, ki jih želimo priporočiti.

Poglavje 6

Evalvacija in rezultati

6.1 Nabor podatkov

Za izvedbo evalvacije naših metod smo imeli na voljo dnevniško datoteko (ang. log file) vseh zahtev za priporočila in klikov na posamezna priporočila. Podatki so bili vzeti z mobilne spletne strani [18] enega izmed večjih ponudnikov spletnih novic s področja tehnologije in novo nastalih podjetij. Vsaka vrstica v dnevniški datoteki je samostojni JSON objekt in predstavlja dogodek, ki se je zgodil na strežniku. Za nas sta bila pomembna dva tipa dogodkov in sicer:

Zahteva za priporočila

Ta dogodek se zgodii vsakokrat, ko nek uporabnik odpre novo novico. Vsakokrat, ko se zgodi ta zahteva, uporabnik prejme deset priporočil. V primeru, da je uporabnik zelo zainteresiran za sorodne vsebine in doseže konec seznama desetih priporočil, se izvede nova zahteva na strežnik za naslednjih deset sorodnih novic. Ker so vse te zahteve ločene, pa čeprav gre za isti priporočilni tok, smo morali te zahteve združiti, ko smo obdelovali podatke.

Uporabniški klik na priporočeno novico

Takrat, ko uporabnik klikne na priporočeno novico, se ta dogodek zabeleži na strežniku. Vsako priporočilo je označeno s svojim enoličnim

identifikatorjem, ki je hranjen tudi pri uporabniškem kliku. S pomočjo tega enoličnega identifikatorja lahko povežemo konkreten prikaz priporočila z uporabniškim klikom. Glede na to, da se priporočila nalagajo v blokih po deset, smo sami izračunali položaj na katerem se je klik zgodil, saj ta podatek ni bil na voljo v dnevniški datoteki.

Za testiranje naših metod smo uporabili podatke v času od 22.2.2014 do 5.5.2014. V tem časovnem okviru je bilo na mobilni spletni strani [18] prikazanih 22.521 različnih priporočilnih tokov. V teh priporočilnih tokovih je bilo priporočenih 33.579 različnih novic. V tem obdobju je bilo izvedenih približno 5 milijonov prikazov priporočilnih tokov, več kot 50 milijonov priporočil in zabeleženih okog 200 tisoč klikov na priporočila.

Pred izvedbo testiranja smo podatke preuredili tako, da smo združili vse priporočene novice in njihove odzive za posamezen ogled novice. To konkretno pomeni, da smo za ogled neke novice, ki se je zgodil ob točno določenem času, vedeli, katere novice so bile priporočene na kateri poziciji in na katere izmed njih je uporabnik kliknil.

6.2 Postopek testiranja

V našem primeru imamo zelo specifičen problem in nismo mogli uporabiti nobene izmed znanih metod za testiranje priporočilnih sistemov ali metod strojnega učenja. Običajno testiramo te probleme tako, da na testni množici napovemo ciljni razred (klasifikacija) oziroma ciljno vrednost (regresija) nekega vektorja značilnosti in to napoved modela primerjamo s pripadajočo oznako. Naš problem si lahko predstavljamo tudi kot rangiranje zadetkov spletnega iskalnika. Za neko iskano geslo obstaja na tisoče zadetkov in veliko izmed njih je tudi relavantnih. Uporabniki pa običajno pregledajo samo zadetke na prvih straneh. Verjetnost zadetka na 10 000 - em mestu je nič [27]. Denimo, da uporabnik klikne zadetke na 1., 3. in 7. mestu ob iskanju nekega poljubnega niza v spletnem brskalniku. Ob predpostavki, da uporabnik pregleduje zadetke zaporedno od prvega proti zadnjemu, lahko trdimo, da je

bil zadetek na 7. mestu boljši od zadetkov na 2., 4., 5. in 6. mestu [27]. To ugotovitev lahko uporabimo tudi na našem problemu, če uporabniku priporočimo n novic in uporabnik klikne j-to novico ($j < n$). Ne vemo sicer, kako kvalitetna so bila priporočila višja od j-tega mesta, saj jih uporabnik morda sploh ni pregledal. Trdimo pa lahko, da je bilo uporabniku priporočilo na j-tem mestu bolj všeč od priporočil, ki so bila predlagana pred j-tim priporočilom in niso bila kliknjena.

Testiranje metod, opisanih v prejšnjem poglavju smo izvedli tako, da smo simulirali zaporedje priporočil v enakem vrstnem redu kot so se zgodila na originalnem priporočilnem sistemu. Za vsako priporočilo smo poznali novice, ki so bile priporočene in katere izmed priporočenih novic so bile kliknjene. V fazi testiranja smo s pomočjo metod prerazporedili novice, ki so bile v originalnem priporočilnem sistemu pozicionirane višje od pozicije, kjer se je zgodil klik. Na primer, če je uporabnik kliknil novico na 5.mestu, smo s pomočjo ene izmed metod opisanih v Poglavju 5 prerazporedili prvih pet priporočenih novic. V primeru, da je bilo klikov na priporočene novice več, smo vedno prerazporedili vse novice do klika na najnižjem položaju. Če bi v prejšnjem primeru uporabnik poleg priporočila na 5. mestu, kliknil tudi priporočilo na 7. mestu, bi v fazi testiranja prerazporedili prvih sedem novic. Primerjali smo položaje kliknjenih novic originalnega priporočilnega sistema in položaje kliknjenih novic, ki bi jih določili s pomočjo različnih metod. Pri testiranju smo se pretvarjali kot, da bi šlo za resničen priporočilni sistem, kjer smo beležili vse stopnje odzivnosti priporočenih novic. Naslednje priporočilo, ki smo ga izvedli v fazi testiranja, je že uporabilo povratne informacije, ki so bile pridobljene ob prejšnjem priporočilu.

V primeru popolnega prerazporejanja bi se klik vedno zgodil na prvem mestu oziroma v primeru več klikov na prvih mestih. Zaradi tega smo za mejenje uspešnosti naših metod uporabili naslednji dve metriki:

Povprečna izboljšava pozicije [28]

Ta metrika primerja pozicijo klika v originalnem priporočilnem sistemu s pozicijo, na katero je bila ta ista novica priporočena s pomočjo metod

opisanih v Poglavju 5.

$$\frac{1}{N}\sum_{i=1}^{N}(\text{position}(\text{original})_i - \text{position}(\text{recommended})_i) \tag{6.1}$$

Natančnost na prvi poziciji (Precision @1) [21]

Merili smo, koliko novic, ki so bile kliknjene v originalnem priporočilnem sistemu, smo priporočili na prvo pozicijo. To število smo normalizirali s številom vseh priporočil.

$$\frac{\text{št. priporočil na prvi poziciji}}{\text{št. vseh priporočil}} \tag{6.2}$$

Običajno pri testiranju metod strojnega učenja podatke razdelimo v tri množice (učno, testno in validacijsko). Učno množico uporabljamo za učenje modela strojnega učenja, ki ga preverimo na testni množici. Validacijsko množico običajno uporabljamo za poročanje končnega rezultata modela in je ne uporabljamo za treniranje modela. Glede na to, da mi evalviramo kvaliteto priporočil nekoliko drugače, smo podatke razdelili v dve množici (učno in validacijsko). V našem primeru smo to delitev naredili zaradi izračuna statistik in vhodnih parametrov. Približno dve tretjini priporočil iz originalnega priporočilnega sistema smo umestili v učno množico (vsa priporočila od 22. 2.2014 do vključno 10.4.2014) in preostalo tretjino v validacijsko množico (ob 11.4.2014 do 5.5.2014). Na učni množici smo opravili izračun vseh statistik, ki smo jih potrebovali za delovanje naših metod. Prav tako smo osnovno testiranje izvedli na učni množici in poiskali optimalne vrednosti vhodnih parametrov. Na validacijski množici smo izvedli zgolj končna testiranja, kjer smo uporabili najboljše parametre za posamezno metodo.

6.3 Predstavitev rezultatov in interpretacija

6.3.1 Primerjava metod

V tem poglavju bomo prikazali vrednosti metrik povprečne izboljšave pozicije in natančnosti na prvi poziciji za vsako izmed opisanih metod v Poglavju 5.

Te vrednosti so bile izmerjene na validacijski množici in so povprečni rezultat desetih ponovitev vsake metode. Testiranje vsake metode smo ponovili desetkrat, saj RPM pristop aproksimira stopnjo odzivnosti kot naključno vrednost, porazdeljeno glede na beta distribucijo z vhodnima parametroma α in β. To pomeni, da bi lahko bili rezultati neke metode ob zgolj eni ponovitvi naključni. Z večkratno ponovitvijo testiranja pa smo to možnost bistveno zmanjšali. Rezultati, ki so predstavljeni v tem podpoglavju, so bili pridobljeni s končnim testiranjem na validacijski množici. Vhodni parametri metod in predhodne statistike, ki smo jih potrebovali, so bile izračunane predhodno na učni množici. S tem smo preprečili, da bi se prenasičili (ang. overfitting). Med seboj smo primerjali osem metod. Večina metod za svoje delovanje potrebuje začetne vrednosti vhodnih parametrov α in β, ki se jim prišteje število klikov in število prikazov. Prav tako smo pri implementaciji dodali parameter, ki določa število dni, za katere opazujemo odzive priporočil. Če imamo vrednost tega parametra nastavljeno na zadnjih deset dni, nas ne zanima, kako se je priporočena novica obnašala več kot deset dni v preteklosti. Ta parameter deluje enako kot časovno okno pri Jackpot problemu, opisanem v Poglavju 4. Določene metode pa potrebujejo za svoje delovanje tudi mejo (ang. threshold). Vsi ti parametri so bili izračunani na učni množici.

V spodnjem odstavku bomo za vsako metodo navedli kakšne vrednosti smo uporabili pri končnem testiranju na validacijski množici.

Požrešna metoda + Globalni CTR

Požrešna metoda ni bila posebej opisana v Poglavju 5. Ta metoda ne uporablja MAB pristopa, ampak razporedi priporočene novice glede na trenutno vrednost stopnje odzivnosti. Pri tej konkretni metodi smo merili stopnjo odzivnosti globalno. Torej enako kot v Poglavju 5.4. Pri njej nismo uporabili časovnega okna, torej smo opazovali celotno zgodovino odzivov.

Globalni CTR + pozicijska prilagoditev

Ta metoda je takšna kot smo jo opisali v Poglavju 5.4. Uporabili smo

vrednosti $\alpha = 10$ in $\beta = 10$. Pri tej metodi nismo uporabili časovnega okna, torej smo opazovali celotno zgodovino odzivov.

Izračun CTR-ja na nivoju priporočilnih tokov

Ta metoda je opisana v Poglavju 5.5. Uporabili smo vrednosti $\alpha = 10$ in $\beta = 10$ in upoštevali odzive za priporočene novice v zadnjih 3-eh dneh.

Hierarhična aproksimacija Metoda je opisana v Poglavju 5.6.2. Za začetno vrednost smo uporabili $\alpha = 1$ in $\beta = 1$. Uporabljali smo vso zgodovino odzivov in mejno vrednost 0.001.

k-najbližjih priporočilnih tokov

To metodo smo opisali v Poglavju 5.6.3. Kot boljši pristop se je v fazi testiranja pokazal kombiniran pristop. Torej, ko je varianca za izbrana parametra α-število klikov in β-število prikazov manjša od meje, uporabimo ti vrednosti. V nasprotnem primeru pa aproksimiramo α in β s k-najbližjimi priporočilnimi tokovi. Za začetni vrednosti smo uporabili $\alpha = 10$ in $\beta = 10$ in upoštevali odzive za priporočene novice v zadnjih treh dneh. Meja, ki smo jo uporabili je bila 0.008. Najboljša vrednost za parameter k je bila 10. To pomeni, da smo parametra α in β aproksimirali s pomočjo desetih najbližjih priporočilnih tokov.

Ocenjevanje začetnih vrednosti α in β (na začetku)

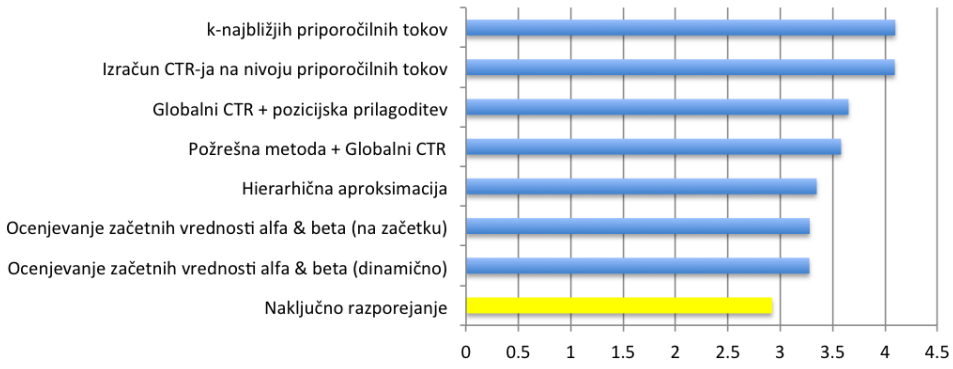
Prva izmed metod opisanih v Poglavju 5.7. Pri tej metodi smo vzeli globalne stopnje odzivnosti vseh priporočenih člankov in aproksimirali začetne vrednosti parametrov α in β. Vrednosti, ki smo jih izračunali in uporabili, sta bili $\alpha = 3$ in $\beta = 256$. Uporabili smo celotno zgodovino odzivov.

Ocenjevanje začetnih vrednosti α in β (dinamično) Druga opisana metoda v Poglavju 5.7, ki dinamično izračunava vrednosti parametrov, ne potrebuje začetnih vrednosti parametrov α in β. Uporabili smo celotno zgodovino odzivov.

Naključno razporejanje

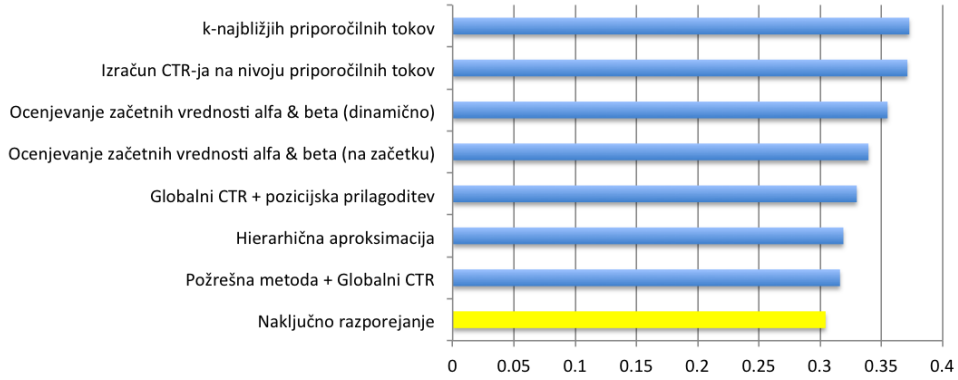
To je najpreprostejša možna hevristika, ki razporedi priporočene novice v naključnem vrstnem redu. To metodo smo uporabili, da smo lahko primerjali naše metode s to najpreprostejšo.

Na Sliki 6.1 vidimo, da smo v povprečju največji pozicijski napredek dosegli z metodo k-najbližjih priporočilnih tokov. S to metodo smo kliknjene novice v povprečju priporočili več kot štiri pozicije višje. Zelo podobne rezultate smo dosegli s priporočanjem na podlagi predhodnih odzivov na nivoju priporočilnih tokov. S tema dvema metodama smo v povprečju izboljšali pozicijo za približno 40% glede na naključno razporejanje. Precej slabše sta se odrezali metodi ocenjevanje začetnih vrednosti parametrov α in β ter hierarhična aproksimacija stopnje odzivnosti. Ti dve metodi sta dosegli manjšo pozicijsko izboljšavo od požrešne metode, ki uporablja globalne stopnje odzivnosti. Prav tako lahko iz te meritve vidimo, da pozicijski faktor in MAB pristop izboljšata požrešno metodo za približno 2%.



Slika 6.1: Povprečna izboljšava pozicije za različne metode

Slika 6.2 prikazuje, kolikšen odstotek priporočil, ki so bila kliknjena pri originalnem priporočilnem sistemu, bi posamezna metoda postavila na prvo mesto. Tudi skozi prizmo te metrike smo dobili najboljše rezultate pri metodi k-najbližjih priporočilnih tokov V primerjavi z naključnim razporejanjem smo



Slika 6.2: Natančnost na prvi poziciji za različne metode

s to metodo dosegli za več kot 22% boljši rezultat. Metoda dinamičnega ocenjevanja začetnih parametrov α in β se izkaže kot boljša, če opazujemo samo kolikšen odstotek kliknjenih priporočil postavi na prvo pozicijo. Naključno razporejanje izboljša za več kot 16%.

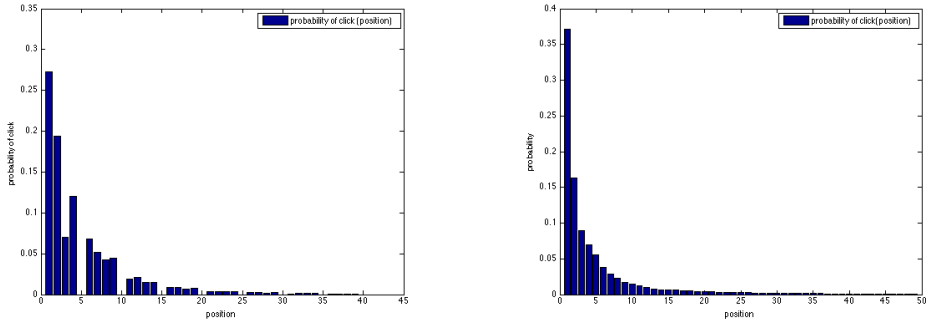
Iz rezultatov je razvidno, da je smiselno meriti stopnjo odzivnosti priporočenih novic na nivoju priporočilnih tokov. Najboljše rezultate smo dosegli z metodo k-najbližjih priporočilnih tokov, vendar je razlika med to metodo in drugo najboljšo (Izračun CTR-ja na nivoju priporočilnih tokov) zanemarljiv. Pomembno je poudariti, da metoda k-najbližjih tokov v 75% odstotkih priporočil vrača povsem enako priporočilo kot metoda Izračun CTR-ja na nivoju priporočilnih tokov. Zgolj v četrtini primerov (za mejo 0.008) je varianca večja od meje in se priporočilo aproksimira s pomočjo desetih najbližjih tokov. Očitno pa so v tej četrtini primerov priporočila nekoliko boljša in zaradi tega pride do te majhne razlike med metodama. Zanimivo je tudi dejstvo, da se omenjena metoda obnaša precej slabše, če aproksimiramo stopnjo odzivnosti z večjim številom priporočilnih tokov. Še precej slabše rezultate smo dosegli z metodo, ki hierarhično aproksimira stopnjo odzivnosti. Očitno je, da je takšna statična struktura presplošna in ima vsaka skupina (tudi na najnižjem nivoju) preveliko število članov. Glede na dejstvo, da se metoda k-najbližjih priporočilnih tokov obnaša slabše, če povečamo$k$, ni nepričakovano,

da so ti rezultati še toliko slabši.

Metodi za ocenjevanje začetnih vrednosti parametrov α in β se prav tako nista pokazali kot zelo uspešni. Razlog za to je, da smo z začetnimi vrednostmi, ki odražajo (povprečno) stopnjo odzivnosti, podaljšali fazo raziskovanja (ang. exploration). Če denimo ocenimo začetni vrednosti na $\alpha = 2$ in $\beta = 100$, se novici, ki sta bili prikazani 50-krat in imata za en klik razlike, praktično ne bosta razlikovali. Naša motivacija za to metodo je bila ravno ta, da bi takšno ne-razlikovanje lahko bilo dobro, vendar rezultati kažejo obratno. Zanimivo pa je, da se omenjena metoda obnaša precej boljše pri odstotku kliknjenih priporočil na prvi poziciji. To pomeni, da je ta metoda prav tako sposobna identificirati tiste najboljše novice.

V naši raziskavi smo preizkusili veliko metod za ocenjevanje vhodnih parametrov α in β beta distribucije. Izkazalo se je, da je zelo pomembno, da merimo stopnjo odzivnosti glede na kontekst priporočilnih tokov. Ugotovili smo, da je aproksimacija vhodnih parametrov s pomočjo vsebinsko sorodnih priporočilnih tokov smiselna samo v primeru, ko to počnemo z majhnim številom sorodnih priporočilnih tokov. V splošnem pa bi lahko trdili, da so na naši podatkovni množici priporočilni tokovi med seboj neodvisni. Zaradi tega se metode, ki so temeljile na vsebinski podobnosti priporočilnih tokov, niso izkazale kot zelo uspešne. Kljub temu, da smo najboljše rezultate dosegli z metodo k-najbližjih priporočilnih tokov, je izboljšava metode Izračun CTR-ja na nivoju priporočilnih tokov zanemarljiva. Potrebno je upoštevati tudi dejstvo, da je metoda k-najbližjih priporočilnih tokov bistveno zahtevnejša za implementacijo. Prav tako je izračun primerjave novic po vsebini časovno gledano zelo potratna operacija, medtem ko pri metodi Izračun CTR-ja na nivoju priporočilnih tokov hranimo zgolj statistiko predhodnih odzivov. Zaradi tega je po našem mnenju najboljša metoda Izračun CTR-ja na nivoju priporočilnih tokov, ki v povprečju izboljša pozicijo klika za 40% glede na naključno metodo.

Na Sliki 6.3 vidimo primerjavo porazdelitve klikov na originalnem priporočilnem sistemu in izboljšanem priporočilnem sistemu, ki izračunava sto-



(a) Originalnem priporočilni sistem (b) Izboljšan priporočilni sistem

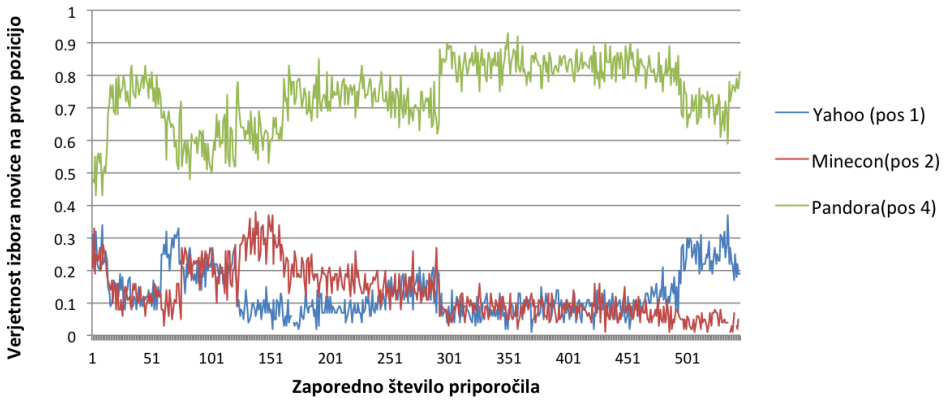
Slika 6.3: Primerjava distribucije klikov glede na položaj na originalnem in izboljšanem priporočilnem sistemu z metodo Izračun CTR-ja na nivoju priporočilnih tokov

pnje odzivnosti na nivoju priporočilnih tokov. Vidimo lahko, da je verjetnost klika na prvem mestu za več kot 10% višja pri izboljšanem priporočilnem sistemu. Kljub temu, da izboljšanega priporočilnega sistema nismo preizkusili v produkciji pa iz rezultatov lahko trdimo, da bi z našim pristopom povečali zaupanje v naš priporočilni sistem. Zagotovo bi višje priporočali tiste novice, ki so uporabnikom bolj zanimive. Posledično pa je velika verjetnost, da bi se povečala tudi stopnja odzivnosti celotnega priporočilnega sistema, saj večina uporabnikov pregleda zgolj prvih nekaj priporočenih novic. Z našo izboljšavo bi na prva mesta postavili bolj relevantne priporočene novice.

6.3.2 Prikaz delovanja MAB algoritma na konkretnem priporočilnem toku

V Poglavju 5.1 smo opisali konkreten priporočilni tok, kjer se je priporočena novica na četrtem mestu odrezala boljše kot novica na prvem mestu. Točne podatke o stopnjah odzivnosti treh priporočenih novic po 545 priporočilih so navedeni v Tabeli 5.1. Osnovni priporočilni sistem ni spreminjal vrstnega reda priporočenih novic, saj je deloval na podlagi podobnosti in ni upošteval kolikokrat je bila posamezna novica kliknjena in prikazana. Na Sliki 6.4

smo grafično prikazali, s kolikšno verjetnostjo bi priporočili posamezno priporočeno novico na prvo mesto z našim izboljšanim priporočilnim sistemom. Ta meritev je bila opravljena z metodo, ki uporablja stopnje odzivnosti glede na kontekst priporočilnega toka.



Slika 6.4: Simulacija delovanja MAB algoritma za konkreten priporočilni tok

Skozi celotno zaporedje priporočil bi z najvišjo verjetnostjo bila na prvem mestu priporočena novica z naslovom *"Pandora finally comes to Chromecast via Android & iPhone apps"*. Ta novica je bila na originalnem priporočilnem sistemu priporočena na četrtem mestu, a je imela najvišjo stopnjo odzivnosti. Iz konkretne simulacije lahko vidimo, da bi izboljšan priporočilni sistem identificiral najbolj obetavno priporočeno novico že po nekaj priporočilih. Še več, takšna izboljšava se prilagaja tudi trenutnim trendom. Vidimo, da je ob koncu (okrog 500-ega priporočila v zaporedju) novica, ki je bila na originalnem priporočilne sistemu prikazana na prvi poziciji, dobila večje število klikov in se je verjetnost, da bi bila ta novica prikazana na prvi poziciji nekoliko povečala.

Poglavje 7

Sklepne ugotovitve

V tej magistrski nalogi smo pregledali obstoječe metode priporočilnih tokov in MAB algoritme. Ti dve področji smo povezali na konkretnem Zemanta Streams priporočilnem sistemu, ki temelji na vsebinski podobnosti. Obstoječ vsebinski priporočilni sistem smo nadgradili tako, da smo sorodne priporočene novice prerazporedili s pomočjo MAB pristopa. Ta pristop deluje na način, da na prvo mesto priporoči tisto novico, ki je v danem trenutku glede na predhodne odzive najbolj obetavna.

Obstoječ vsebinski priporočilni sistem smo izboljšali tako, da smo izboljšali položaj tistih novic, ki imajo višjo verjetnost klika. V okviru naše magistrske naloge smo se osredotočili na optimizacijo stopnje odzivnosti. Naš pristop pa je mogoče posplošiti in ga je mogoče uporabiti za optimizacijo tudi drugih metrik obstoječega priporočilnega sistema. Na primer, da imamo spletno trgovino, ki ima priporočilni sistem, ki zelo dobro priporoči artikle, ki so zanimivi trenutnemu uporabniku. Naš cilj pa bi bil optimizirati ta priporočilni sistem tako, da bo dobiček čim višji. Ni nujno, da je artikel, ki ga priporočilni sistem priporoči kot najboljšega, tudi najbolj dobičkonosen. Ta priporočilni sistem bi lahko s pomočjo MAB algoritma izboljšali tako, da bi priporočali artikle, ki so uporabniku zanimivi in hkrati nam kot prodajalcu prinašajo večji zaslužek.

Pri izboljšavi Zemanta Streams priporočilnega sistema smo uporabili RPM

algoritem. To je MAB algoritem, ki ocenjuje stopnjo odzivnosti s pomočjo naključne spremenljivke, porazdeljene glede na beta distribucijo z vhodnima parametroma α in β. RPM algoritem smo uporabili zato, ker se je izkazal kot najboljši na primeru Jackpot. Z njim smo dosegli 85% maksimalne možne nagrade. Pri izboljševanju obstoječega priporočilnega sistema smo se ukvarjali predvsem s problemom določanja vrednosti vhodnih parametrov α in β beta distribucije. Kot najboljši pristop se je izkazal ta, kjer sta vrednosti α in β določeni s številom klikov oziroma številom neuspešnih priporočil (priporočilo je neuspešno, ko je priporočano, a ga uporabnik ne klikne) pod trenutno prikazano novico. Ta pristop izboljša povprečen položaj kliknjenega priporočila za 40% glede na naključno razporejene priporočene sorodne novice.

Metode za izboljšavo obstoječega priporočilnega sistema [19] smo objavili na delavnici NewsKDD v sklopu konference KDD2014. Naša rešitev problema Jackpot pa je bila izbrana za najboljšo študentsko rešitev Celtrinega programerskega izziva 2014.

Literatura

[1] G. Linden, B. Smith, J. York, "Amazon. com recommendations: Item-to-item collaborative filtering", v reviji Internet Computing, IEEE, vol. 7(1), str. 76-80, 2003.

[2] Y. Koren, R. Bell, C. Volinsky, "Matrix factorization techniques for recommender systems", v reviji Computer, vol. 8, str. 30-37, 2009.

[3] C. Anderson, "The Long Tail: Why the Future of Business is Selling Less of More", v Hyperion Books, 2006.

[4] Dostopno na `http://www.fri.uni-lj.si/si/raziskave/studentski_izzivi/celtrin_izziv/`, 3.6.2015.

[5] A. Gediminas, A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", v reviji Knowledge and Data Engineering, IEEE Transactions, vol. 17(6), str. 734-749, 2005.

[6] J. Leskovec, "Recommender Systems: Content-based Systems & Collaborative Filtering", gradivo pri predmetu CS246 Stanford University, 2014.

[7] J. Leskovec, A. Rajaraman, J. D. Ullman, "Recommendation Systems", v knjigi Mining of massive datasets, Cambridge University Press, str. 305-340, 2014.

[8] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, "Collaborative filtering recommender systems", v reviji Foundations and Trends in Human-Computer Interaction, vol. 4(2), str. 81-173, 2011.

[9] Dostopno na `http://blog.zemanta.com/zemanta-tools-for-bloggers/stream/`, 1.6.2015.

[10] P. Auer, N. Cesa-Bianchi,P. Fisher, "Finite-time analysis of the multiarmed bandit problem", v reviji Machine learning, vol. 47(2-3), str. 235-256, 2002.

[11] S. Scott,"A modern Bayesian look at the multi-armed bandit", v zborniku Applied Stochastic Models in Business and Industry, vol. 26(6), str. 639-658, 2010.

[12] Dostopno na `http://research.microsoft.com/en-us/projects/bandits/MAB-2.jpg`, 3.6.2015.

[13] W. H. Press,"Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research', v zborniku Proceedings of the National Academy of Sciences, vol. 106(52), str. 22387–22392, 2009.

[14] Dostopno na `http://www.google.com/websiteoptimizer`, 1.6.2015.

[15] L. Li, W. Wei, J. Langford, R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation', v zborniku Proceedings of the 19th international conference on World wide web, str. 661-670, 2010.

[16] J. White,"Bandit algorithms for website optimization", izdano pri O'Reilly Media, Inc., 2012.

[17] Dostopno na `https://github.com/andrazhribernik/jackpot`, 3.6.2015.

[18] Dostopno na: `http://venturebeat.com/`, Maj 2014.

[19] A. Hribernik, L. Dali, D. Omerčević, D. Lavbič, "Applying Multi-Armed Bandit on top of content similarity recommendation engine", na delavnici NewsKDD v sklopu konference KDD 2014, Avgust 2014.

[20] Dostopno na: `https://bitbucket.org/dlavbic/predicting-optimal-related-web-article-with-multi-armed-bandit`, 1.6.2015.

[21] C. D. Manning, P. Raghavan, and H. Schutze, "Introduction to information retrieval", Cambridge university press Cambridge, 2008.

[22] M. S. Charikar, "Similarity estimation techniques from rounding algorithms" v zborniku Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, str. 380-388, 2002.

[23] B. Fortuna, M. Grobelnik, D. Mladenič, "OntoGen: Semi-automatic Ontology Editor ", na konferencei HCI International 2007, Peking, Julij 2007.

[24] R. Řehůřek, P. Sojka, "Software Framework for Topic Modelling with Large Corpora", v zborniku LREC 2010 Workshop on New Challenges for NLP Frameworks, str. 45-50, Maj 2010.

[25] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval" v knjigi Machine learning: ECML-98, 4–15, 1998.

[26] M. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives." v reviji The computer journal, vol. 7(2), str. 155-162, 1964.

[27] T. Joachims. , "Optimizing search engines using clickthrough data" v zborniku 8. ACM SIGKDD international conference on Knowledge discovery and data mining, str. 133–142, 2002.

[28] B. Libby, "Modeling Clickthrough Probabilities.", 2009.