Тервел Вълков, специалност *Изкуствен Интелект*, ФН 9MI3400058

**Проект по Препоръчващи системи**

*Тема -* *Експерименти свързани със статия от RecSys2021*

*Analyzing Item Popularity Bias of Music Recommender Systems: Are Different Genders Equally Affected?*

1. Увод

В областта на препоръчващите системи, терминът *popularity bias* се отнася към несъответствия между популярностите на елементите в списъците с препоръки – най-често това означава непропорционално по-голям брой популярни артикули отколкото по-малко популярните. Такъв феномен е забелязан в много различни области, като например препоръчване на филми или на музика. *Collaborative filtering* алгоритмите са особено уязвими, тъй като данните върху които те биват тренирани обикновено вече проявяват непропорционалност (има повече интеракции между потребители и популярните артикули отколкото за непопулярните).

Разпределението на популярности на предмети в повечето сфери показва long-tail характеристика, тоест голям брой предмети присъстват в опашката на разпределението (в случая, имат малки популярности). Повечето изследвания на *popularity bias* използват прости статистически метрики и не взимат предвид характеристиките на потребителите, поради което не става ясно как различни потребителски групи биват засегнати. В статията, върху която е базиран този проект, е направен опит за адресиране на тези недостатъци, чрез изразяване на отклонението в популярността въз основа на медианата и няколко статистически момента, както и метрики за сходство между разпределения на популярността, като опитът се фокусира конкретно на пола като потребителска характеристика.

1. Данни

В статията се ползва dataset-а *LFM-2b*[3], който съдържа над 2 милиарда събития от онлайн платформата за музика *Last.fm*. Той е разработен от екип съдържащ повечето от авторите и на статията, за целите на предишно изследване[2], което е цитирано и в тази статия.

Тъй като целият набор от данни е масивен – над 130 гигабайта, за научни цели се предлага едно негово подмножество, наречено *LFM-2b-DemoBias*. Докато цялото множество съдържа над 120 000 потребителя, *DemoBias* съдържа около 61 000 потребителя и 1.7 милиарда събития. Допълнително за целите на статията е взето още по-малко подмножество – първо е използвана само частта от данните, релевантна за *collaborative filtering*, тоест само интеракциите между потребители и песни, за които има информация за timestamp, и броят слушания на дадена песен от даден потребител. Второ, тези данни са филтрирани, като са взети само интеракции с повече от едно слушане, само песните слушани от поне 5 различни потребителя, само потребителите слушали поне 5 различни песни, и само интеракциите от последните 5 години. Информацията за брой слушания е игнорираната – интеракциите се считат за бинарни, или потребител е слушал дадена песен или не. Това води до около 23 000 потребителя и 1.6 милиона песни. Финално, взима се случайна извадка само на 100 000 песни, при което се получават общо около 20 милиона събития с избраните песни. Подобна процедура е описана в кода към статията[4], в която е въведено множеството LFM-2b.

За съжаление, изглежда множество *LFM-2b-DemoBias* вече не е достъпно на сайта на *LFM-2b* по неизвестни причини – там присъства само пълната версия на данните, Поради тази ситуация ми се наложи да създам собствена извадка от данните, следвайки описаната процедура и кода от [4]. Използвах предложения вариант на данните, адаптиран за изграждане на препоръчващи системи на песни, който съдържа само 3 от 10те файла на пълното множество – демографска информация за потребителите, информация за песните (която не беше нужна за целите на проекта), и интеракциите между потребители и песни, които се състоят от идентификатори съответно на потребителя и на песента, както и брой слушания. В тези данни липсват timestamp-овете – в пълното множество тази информация е в отделен файл за събитията, който е доста голям – докато броя слушания е около 2 гигабайта в архивиран формат и се разархивира до 8, файлът с timestamp-ове е 14 гигабайта архивиран, поради което не съм извършвал опити да го смъкна за да разбера пълния му размер след разархивиране.

Поради това при моето създаване на извадка липсва стъпката с филтриране на старите събития. Също така, тъй като дори и това ограничено подмножество е прекалено голямо за зареждане в паметта на моя компютър, се наложи да ползвам функционалността на *pandas* за зареждане на dataframe от файл чрез четенето му на chunk-ове, което наложи и някои други модификации в кода. Реших да добавя и допълнителен филтър, с който взимам само потребителите за които има информация за пола (тъй като в данните има голям брой записи с липсващ пол).

Поради работата с крайно различно множество, моята извадка е доста далеч от използваната в статията – броят потребители всъщност е близък (малко под 23 000), но броят песни и събития е много по-малък – само 7140 и около 2 милиона, съответно.

1. Алгоритми

В оригиналната статия са оценявани резултатите на 7 препоръчващи алгоритъма:

* *Random Item* - прост baseline алгоритъм, препоръчващ случайни песни (избягвайки да препоръчва такива които дадения потребител вече е слушал)
* *Most Popular Items* – друг прост baseline алгоритъм, който на всеки потребител препоръчва същото множество от най-популярни песни.
* *Item k-Nearest Neighbors* (ItemKNN)
* *Sparse Linear Method* (SLIM) – базиран на съседство алгоритъм (подобно на kNN), но изчисляващ сходството директно върху данните чрез регресия
* *Alternating Least Squares* (ALS) – подход с факторизиране на матрици, който научава embedding-и за потребителите и предметите, такива че скаларното им произведение да приближава оригиналната матрица
* Matrix factorization with *Bayesian Personalized Ranking* (BPR) – също научава embedding-и, но с функция за оптимизация която цели да ранкира консумираните предмети според предпочитанията на потребителите
* *Variational Autoencoder* (VAE) – алгоритъм базиран на autoencoder-и, който от вектор с интеракции на потребител намира вероятностно разпределение на всички предмети

Към статията липсва код, но в цитираната статия[2] са ползвани същите алгоритми (без *Random Item*) и има линк към GitHub repository с кода за експеримента от нея. В него има имплементации на алгоритмите, но за съжаление тази статия се фокусира на много по-различен експеримент, което затрудни разбирането на кода и опитите да бъде адаптиран за целите на проекта. Освен това, изглежда и че поне някои от тези алгоритми са тежки откъм изчислителна гледна точка – евентуалното успешно пускане на единият от алгоритмите доведе до напълно замръзване на компютърът ми и нужда от ръчен рестарт. ALS и BPR се базират на имплементации от библиотеката *implicit*, и направих опит да използвам директно тях, но срещнах затруднения и за съжаление библиотеката е много бегло документирана.

Поради тези трудности и неуспехи се отказах от варианта с тези алгоритми и реших да използвам имплементации от библиотеката *surprise*[5], конкретно:

* *NormalPredictor* и *BaselineOnly* като два *baseline* алгоритъма, заместващи *Random Item* и *Most Popular*
* *kNNWithMeans* и *kNNBaseline* като два базирани на съседство алгоритъма, като те биват конфигурирани да изчисляват сходство между предмети вместо потребители (тъй като в оригиналната статия се ползва *ItemKNN*)
* *SVD* и *NMF* (*Non-negative Matrix Factorization)* като два алгоритъма с факторизиране на матрици
* допълнително, *Slope One* и *Co-Clustering*

За намиране на добри параметри за тях беше използвана функционалността на *surprise* за *grid search* с крос-валидация.

1. Метрики

В статията се използват два типа статистически метрики. Първите са делта-метрики (%Δ), които са дефинирани чрез разликата на резултатите от прилагането на дадена метрика към разпределенията на популярност върху списъка с препоръки към даден потребител и върху неговата историята на слушания. В случая, популярността на дадена песен е дефинирана като сумата от бройките на всички нейни слушания, а разпределенията просто имат тази популярност за песните които присъстват съответно в историята или препоръките, и 0 за останалите. Конкретните метрики, които биват изследвани са средно, медиана, дисперсия, както и по-нестандартните *skew* и *kurtosis*. Метриките се изчисляват за всеки потребител, след което за всеки тип метрика се взима медианата на стойностите за отделните потребители. Тези финални стойности се взимат както за цялото множество от потребители, така и по отделно за подмножествата от мъже и жени.

Счита се, че позитивно *%Δ средно* и *%Δ медиана* сочат към това, че на потребителя като цяло са били препоръчани повече популярни песни, а позитивна *%Δ дисперсия* означава че списъкът на препоръки е бил по-разнообразен откъм популярности отколкото историята. Това също би могло да означава увеличаване на отклонението към популярни предмети, тъй като най-популярните са разпределени нарядко в диапазона на популярност. Позитивен *%Δ skew* обозначава, че дясната опашка на препоръките е по-тежка от тази на потребителската история, което означава склонност повече предмети да имат по-ниска популярност от диапазона. Финално, позитивен *%Δ kurtosis* показва че опашките на разпределението на популярност на препоръките са по-тежки, и самото разпределение е един вид по-близко до равномерното разпределение.

Вторият тип са метрики за сравняване на целите разпределения на популярност - *Kullback–Leibler Divergence (KL)* и *Kendall's Tau (KT)*. Тяхното приложение е по-специфично обаче – не се прилагат директно върху разпределенията както делта-метриките, а върху техни decile-binned версии, избрани така че във всеки bin да се съдържат песни със сума на популярностите около 10% от сумата на всички популярности. От информацията в статията не успях да разбера как всъщност се прилагат метриките върху отделните bin-ове, поради което в проекта съм включил само Kendall’s Tau, приложена върху целите разпределения.

Допълнително, за оценка на самите алгоритми, в статията се ползва ранкираща метрика - *NDCG@10*. Поради липса на описание как всъщност се прилага и липса на опит с тази метрика, реших да ползвам различен тип оценка – *RMSE*, която може директно да се оцени чрез функция на *surprise*.

1. Реализация

Първата част от реализацията е създаването на малка извадка от данните, която да ползваме след това. Това се случва в *item\_sampler.ipynb*, като първо взимаме извадка на 100 000 песни, и след това създаваме dataframe-и с демографската информация на потребителите и с интеракциите (като прочитаме файла с тях на chunk-ове, поради големия му размер), взимайки само записите които имат повече от 1 слушане, песен измежду тези в извадката, и потребител с информация за пола. След това премахваме песните с под 5 слушания и потребителите с под 5 песни, към dataframe-а с интеракциите добавяме информацията за пола, и го записваме като .*tsv* файл.

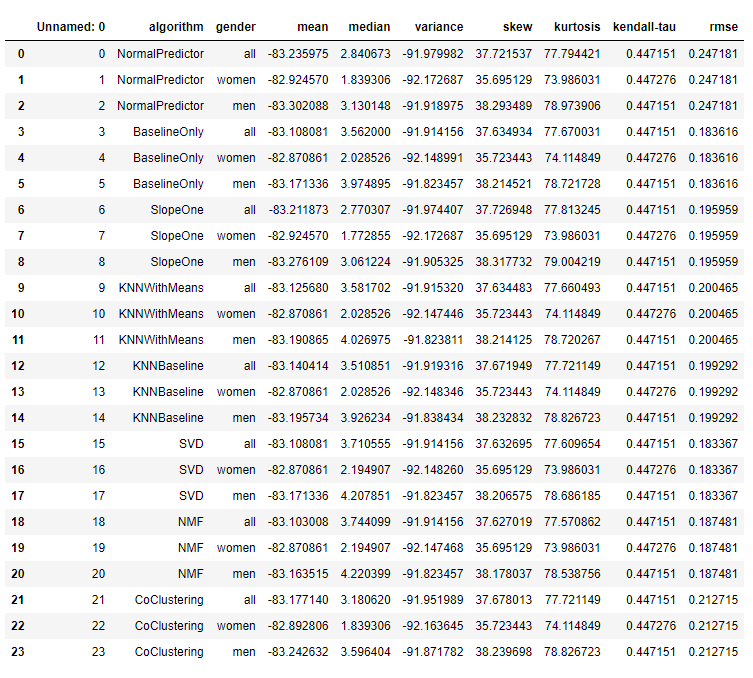
След като вече имаме извадката, остатъкът от експеримента се извършва с *music\_gender\_bias.ipynb*. Зареждаме данните, като от тях си създаваме два dataframe-а – единия само с интеракциите, другия само с потребителските идентификатори и половете. Изчисляваме популярностите на песните и намираме историите на слушания за всеки потребител.

За разлика от оригиналната статия, в която информацията за брой слушания не бива използвана, тук реших да опитам да я употребя. Формално, брой слушания не е точно експлицитен рейтинг, и всъщност някои от алгоритмите в оригиналната статия (като тези от библиотеката *implicit*) са точно алгоритми използващи имплицитни данни. За съжаление, *surprise* е библиотека поддържаща само ескплицитни рейтинги, което затруднява ползването ѝ тук.

Направих опит да използвам бройките директно като рейтинги, при което оценката за грешка на алгоритмите беше доста лоша, най-вероятно тъй като така се получава масивна скала за рейтинги – някои потребители са слушали песни над 1000 пъти. Поради това реших да създам изкуствени рейтинги на база на бройките, като ги скалирах в интервала [1, 20].

След като данните са готови, може да преминем към тренирането на алгоритмите. В статията е приложена сложна схема за разделянето на данни – първо се извършва разделяне на *train-validation-test* множества, но не върху цялото множество данни, а специфично върху потребителите, като 60% от тях са в обучаващото множество, а валидационното и тестовото имат по 20%. Самите интеракции се разделят на базата на това разделение на потребителите. След това се извършва 5-fold крос валидация, като при валидацията и тестването има още специфики – за всеки потребител случайно се избират 80% от интеракциите му за трениране и 20% за тестване, като това се прави както за валидационното множество, така и за тестовото.

За целите на проекта беше избрана доста по-проста схема, като просто интеракциите директно се разделят на тренировъчно и тестово множество, 80% на 20%. След като всеки алгоритъм се обучи, взимаме топ-10те му препоръки за всеки потребител от тестовото множество, и изчисляваме различните метрики, като резултатите записваме в dataframe и евентуално във файл.

1. Резултати и заключение

За разлика от оригиналната статия, където различните алгоритми имат много различни резултати за отделните метриките, тук резултатите са много близки, най-вероятно тъй като избраните алгоритми от *surprise* не се толкова радикално различни като тези в оригиналната статия, и съответно правят подобни препоръки.

Освен това, изборът на *surprise* в случая не е напълно подходящ, тъй като това е библиотека, създадена за работа само със данни с ескплицитни рейтинги, а в нашите данни имаме само информация за брой слушания, която е по-скоро имплицитен рейтинг.

При разработката на проекта бяха срещнати много препятствия, които затрудниха опита за пресъздаване на експеримента в статията. Първо, използваните данни (или по-точно, използваното подмножество на данните) всъщност изглежда че не са достъпни, поради което се наложи създаването на собствена извадка от данните. Второ, бяха срещнати трудности при използването на същите алгоритми като в статията, поради което бяха приложени други алгоритми, от *surprise*, които не са напълно подходящи за този тип препоръки. Трето, бяха срещнати и проблеми при разбирането и използването на метриките, поради липса на достатъчно детайли за това в статията.

Все пак, проектът доведе и до научаване на някои полезни аспекти на работата с библиотеки като *pandas* и *numpy*, както и знания за предварителната обработка на данни за ползването им в препоръки, и за изчисляване на различни статистически метрики върху резултатите на една препоръчваща система.

1. Източници
2. Oleg Lesota et al., *Analyzing Item Popularity Bias of Music Recommender Systems: Are Different Genders Equally Affected?*, RecSys 2021 – Late Breaking Results

[<https://arxiv.org/pdf/2108.06973.pdf>]

1. Alessandro B. Melchiorre et al., *Investigating Gender Fairness of Recommendation Algorithms in the Music Domain*, Johannes Kepler University Linz, Austria & Linz Institute of Technology, AI Lab, Austria

[<http://www.cp.jku.at/people/schedl/Research/Publications/pdf/melchiorre_ipm_2021.pdf>]

1. LFM-2b Dataset - Corpus of Music Listening Events for Music Recommendation and Retrieval

[<http://www.cp.jku.at/datasets/LFM-2b/>]

1. *Investigating Gender Fairness of Recommendation Algorithms in the Music Domain*, GitHub, item\_sampler.ipynb [<https://github.com/CPJKU/recommendation_systems_fairness/blob/main/notebooks/item_sampler.ipynb>]
2. Surprise - a Python scikit for building and analyzing recommender systems that deal with explicit rating data

[<http://surpriselib.com/>]

[<https://surprise.readthedocs.io/en/stable/>]