Uniwersytet Śląski
Wydział Nauk Ścisłych i Technicznych
Instytut Informatyki

Rozprawa doktorska

mgr Michał Bałchanowski

Ewolucyjna agregacja rang w systemach rekomendacyjnych

Promotor:
prof. dr hab. Urszula Boryczka

Sosnowiec, 2023 r.

Chciałbym podziękować wszystkim,
bez których niniejsza rozprawa by nie powstała.

Serdeczne podziękowania składam mojemu promotorowi
prof. dr hab. Urszuli Boryczce
za poświęcony czas i cenne uwagi, podczas pisania niniejszej rozprawy.

Szczególne podziękowania składam mojej ukochanej ***Agnieszce***
za cierpliwość, wyrozumiałość i motywację.

Rodzicom *za wsparcie przez te wszystkie lata, dedykuję i dziękuję.*

Spis treści

Rozdział 1

Wstęp

Dzięki rozwojowi Internetu mamy dostęp do coraz większej ilości informacji, a obecny trend w postępie digitalizacji wszystkich obszarów naszego codziennego życia, tylko przyspiesza ten proces. Mnogość wyboru sprawia, że podjęcie decyzji jaką książkę przeczytać, jaki film obejrzeć lub jaki przedmiot kupić, staje się coraz większym wyzwaniem. Ze względu na ilość dostępnych danych, nie jesteśmy w stanie przeanalizować wszystkich oferowanych możliwości. Aby ułatwić sobie podjęcie decyzji, często kierujemy się opiniami innych użytkowników, sprawdzając oceny i komentarze dostępne na portalach internetowych. Chcemy mieć pewność, że nasz wybór, będzie właściwy.

Aby wspomóc użytkownika w procesie podejmowania decyzji, zaproponowano systemy rekomendacyjne, których celem jest zasugerowanie użytkownikowi pewnych produktów lub usług, które z dużym prawdopodobieństwem mogą go zainteresować. Nie jest to problem trywialny i od wielu lat prowadzone są prace badawcze w tej tematyce. Najistotniejszym wydarzeniem, które znacznie zwiększyło zainteresowanie tym zagadnieniem, był konkurs zorganizowany 2 października 2006 roku przez firmę *Netflix*, gdzie badaczom, którym udało się wystarczająco zwiększyć jakość generowanych rekomendacji, oferowano do wygrania 1 milion dolarów [23].

Spowodowało to, że na przestrzeni ostatnich lat powstało wiele algorytmów rekomendacyjnych, choć w środowisku naukowym nadal nie ma konsensusu co do tego, które z tych technik są najlepsze, szczególnie w kontekście problemu Top-N rekomendacji (z ang. *Top-N Recommendation Problem*) [12]. Jest to związane między innymi z tym, że istnieje wiele miar służących do określania jakości generowanych rekomendacji, a sam proces ewaluacji może być realizowany na różne sposoby [153].

Dodatkowo algorytmy rekomendacyjne posiadają dużą liczbę parametrów, które przed przystąpieniem do badań muszą zostać odpowiednio dostrojone. W literaturze wielokrotnie [163, 54, 53] zwracano uwagę na problem reprodukowalności wyników eksperymentów, które są raportowane w publikacjach naukowych i autor rozprawy zdaje sobie sprawę z wyzwań i problemów, jakie wiążą się z przeprowadzeniem rzetelnych badań. Z tego względu szczegółowo omówione zostanie środowisko badawcze, które wykorzystano do przeprowadzenia eksperymentów.

Celem zaproponowanego w rozprawie algorytmu, będzie próba przewidzenia listy przedmiotów (rankingu), który z dużym prawdopodobieństwem zainteresuje użytkownika. Do ewaluacji zaproponowanego podejścia, wykorzystane zostaną stosowne miary, które uwzględniają pozycję relewantnych przedmiotów na rekomendowanej liście. Wykorzystanie tych miar do bezpośredniej optymalizacji rankingu nie jest łatwe, ponieważ posiadają one specyficzne właściwości (np. nieróżniczkowalność), które uniemożliwiają zastosowanie klasycznych algorytmów aproksymacyjnych [41]. Z tego względu zaproponowany algorytm, będzie bazował na algorytmie ewolucji różnicowej (z ang. *differential evolution*, DE).

W pracy poruszony zostanie również problem agregacji rang (z ang. *rank aggregation problem*) [63]. Jest to stosunkowo nowe podejście w kontekście systemów rekomendacji, gdzie zamiast jednego algorytmu, wykorzystuje się pewien zbiór algorytmów, który generuje rekomendacje dla danego użytkownika, a następnie wyniki tych algorytmów są agregowane w celu utworzenia nowej rekomendacji. Agregacja nie jest problemem trywialnym, ponieważ nie istnieje jedna, uniwersalna metoda, łączenia takich rankingów ze sobą. Jednak szczegółowe motywacje, którymi kierował się autor podczas wyboru tematyki niniejszej rozprawy, znajdują się w podrozdziale 1.1.

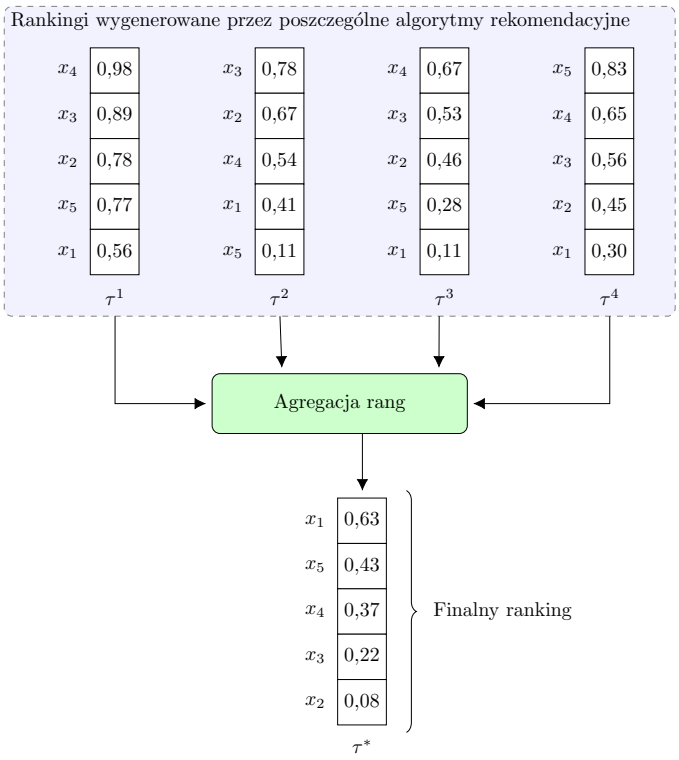
1.1 Motywacja

W literaturze zaproponowano wiele algorytmów rekomendacyjnych, których celem jest zaproponowanie użytkownikowi pewnej uporządkowanej listy przedmiotów [12]. W niniejszej rozprawie taką listę będziemy nazywali rankingiem. Podczas ewaluacji systemów rekomendacyjnych, zazwyczaj generuje się rekomendacje dla poszczególnych użytkowników, następnie oblicza się jakość rekomendacji zgodnie z przyjętą miarą, a otrzymane wyniki uśrednia. Okazuje się jednak, że jeżeli porównamy zaproponowane rekomendacje w kontekście konkretnego użytkownika, to poszczególne algorytmy generują rekomendacje różniące się od siebie, a tym samym różnej jakości.

Tabela 1.1: Trywialny przykład reprezentujący zróżnicowaną jakość rekomendacji w systemie rekomendacyjnym

	Algorytm 1	Algorytm 2	Algorytm 3	Algorytm 4
Użytkownik 1	0,5	0,0	0,0	1,0
Użytkownik 2	0,0	0,5	1,0	0,5
Użytkownik 3	1,0	0,5	1,0	0,5
Użytkownik 4	0,5	1,0	0,0	0,0
średnia	0,5	0,5	0,5	0,5

W tabeli 1.1 zaprezentowany został prosty przykład ilustrujący ten problem (jakość rekomendacji została wyrażona przez wartość z przedziału od 0 do 1). Można zauważyć, że choć skuteczność algorytmów po uśrednieniu jest identyczna, to na poziomie poszczególnych użytkowników jakość rekomendacji jest różna. Ze względu na to, że nie ma jednego, uniwersalnego algorytmu, który generowałby rekomendacje wysokiej jakości dla wszystkich użytkowników w systemie, to w celu poprawienia jakości końcowej rekomendacji można wykorzystać techniki agregujące, które z powodzeniem były wykorzystywane choćby w systemach wyszukiwania informacji [63]. Umożliwiają one fuzję rankingów, które zostały wygenerowane dla danego użytkownika, co w teorii powinno poprawić końcową jakość rekomendacji. Rysunek 1.1 przedstawia ideę agregacji.



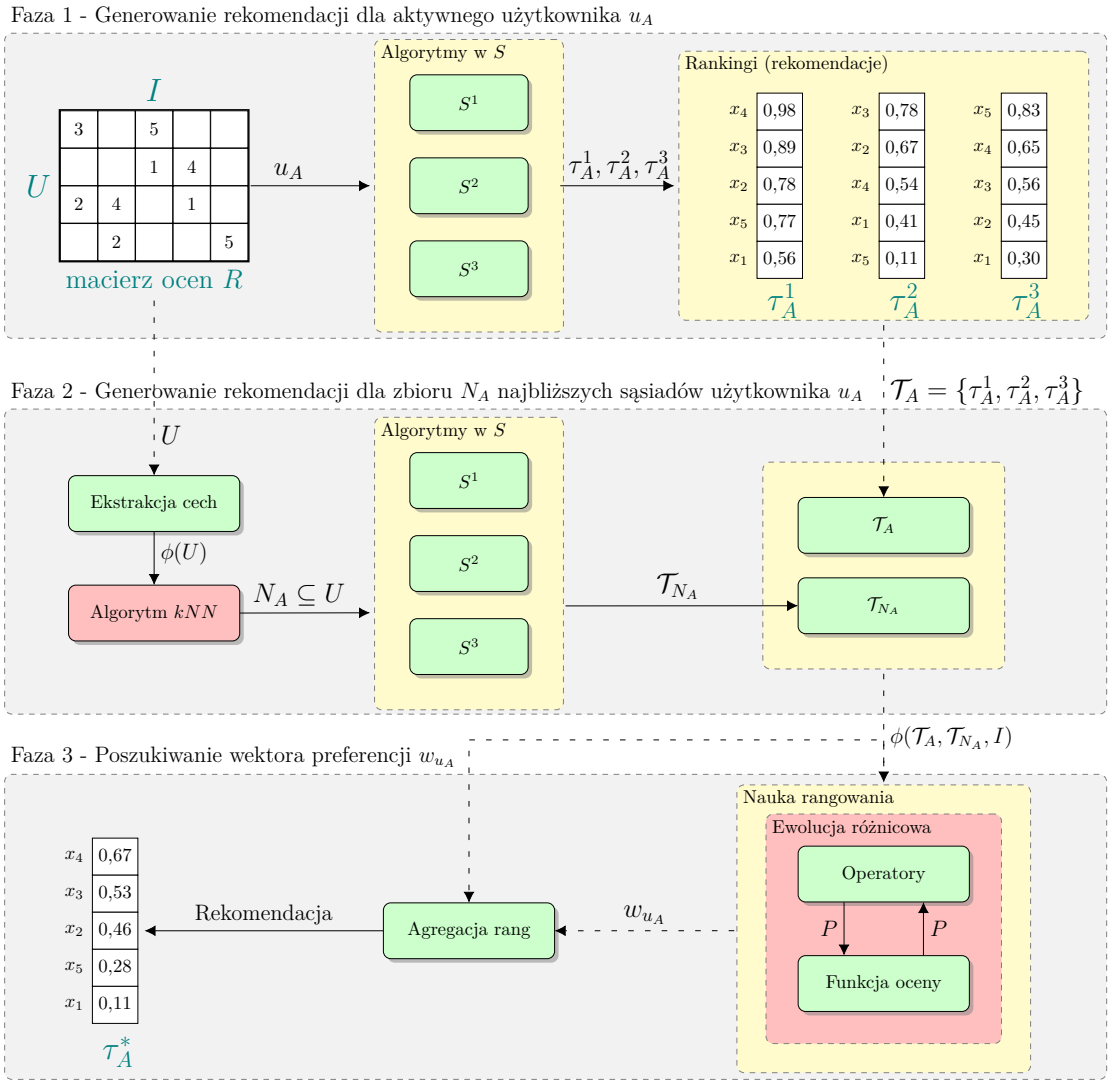
Rysunek 1.1: Przykładowa agregacja czterech rankingów
Opracowanie własne, wykorzystane również w publikacji [20]

Wykorzystanie tej idei w systemach rekomendacji zostało już zasugerowane przez innych badaczy. Przykładowo w pracy [5, s. 417] autor zaproponował wykorzystanie algorytmów agregujących i zaznaczył, że jest to idea zaczerpnięta z *rodziny klasyfikatorów*, gdzie łączy się ze sobą kilka klasyfikatorów, w celu utworzenia optymalnego modelu predykcyjnego. Dodatkowo autor zauważa, że jest to zagadnienie, które w kontekście systemów rekomendacyjnych nie zostało dobrze zbadane i jest ciekawym kierunkiem przyszłych prac badawczych.

Podobne wnioski zostały przedstawione w pracy [126], gdzie autorzy wskazali, że choć w literaturze zaproponowano wiele algorytmów rekomendacyjnych, to często zwracają

one różne rankingi, co stwarza okazję do poprawy końcowego rankingu, poprzez agregację wyników różnych algorytmów. Autorzy wskazują również, że w kontekście systemów rekomendacyjnych powstało stosunkowo mało przeznaczonych do tego celu metod.

Ponadto analizując przegląd literaturowy zaprezentowany w tabeli 1.2 można zauważyć, że liczba prac, w których wykorzystano metaheurystyki w problemie agregacji rang w systemach rekomendacji jest stosunkowo niewielka. Z tego względu autor niniejszej rozprawy dostrzega w tym obszarze **lukę badawczą**, którą ta rozprawa postara się wypełnić, tym samym tworząc podstawę dla kolejnych prac badawczych w tej tematyce. Na diagramie (rys. 1.2) przedstawiona została ogólna idea zaproponowanego podejścia.



Rysunek 1.2: Diagram przestawiający ogólną ideę algorytmu, zaproponowanego w niniejszej rozprawie. Kolorem czerwonym oznaczono główne obszary badawcze. Objaśnienie poszczególnych symboli znajduje się w tabeli 10.3

Tabela 1.2: Przegląd literatury związanej z problemem agregacji rang (w systemach rekomendacji), ze wskazaniem publikacji, w których wykorzystano algorytm metaheurystyczny

Rok	Autorzy	Algorytm metaheurystyczny
2000	Smyth i in. [158]	-
2004	Torres i in. [168]	-
2010	Baltrunas i in. [16]	-
2013	Meena i in. [115]	algorytm genetyczny
2014	Ribeiro i in. [141]	algorytm ewolucyjny
2014	Pessemier i in. [134]	-
2014	Silva i in. [156]	algorytm genetyczny
2016	Oliveira i in. [124]	programowanie genetyczne
2016	Tang i in. [167]	-
2018	Olivera i in. [125]	programowanie genetyczne
2018	Lestari i in. [104]	-
2019	Dong i in. [60]	-
2020	Pujahari i in. [136]	-
2020	Oliveira i in. [126]	-
2021	Yalcin i in. [184]	-
2022	Bałchanowski i in. [21]	ewolucja różnicowa

1.2 Teza rozprawy

Teza rozprawy: *Zaproponowany algorytm ewolucyjnej agregacji rang poprawia jakość generowanej agregacji, w porównaniu z wybranymi metodami zaproponowanymi w literaturze.*

Pod pojęciami:

- *Jakość generowanej agregacji* – należy rozumieć jakość rekomendacji, wygenerowaną na podstawie tej agregacji, wyrażonej przy pomocy miary uśrednionej średnej precyzji (z ang. *mean average precision*, MAP), która została opisana w podrozdziale 3.2.2.
- *Wybranymi metodami* – należy rozumieć pięć metod pozycyjnych, które zostały przedstawione w podrozdziale 5.2.

1.3 Cele rozprawy

Głównym celem rozprawy jest opracowanie autorskiego algorytmu *ewolucyjnej agregacji rang* (EAR), dostosowanego do wykorzystania w systemach rekomendacyjnych i bazującego na algorytmie DE. Zadaniem tego algorytmu będzie agregacja rankingów, wygenerowanych przez poszczególne algorytmy rekomendacyjne i zaprezentowanie użytkownikowi finalnego rankingu, który będzie dostosowany do jego personalnych preferencji. Algorytm ten zostanie porównany z innymi technikami, wykorzystywanymi do tworzenia agregacji. Kolejne etapy realizacji głównego celu pracy obejmują:

1. **Przegląd literatury związanej z tematyką niniejszej rozprawy**. W przeglądzie literaturowym znajdą się ogólne informacje na temat systemów rekomendacji i przedstawiony zostanie problem agregacji rang. Dodatkowo zaprezentowany zostanie problem uczenia się rankingu oraz algorytm DE.

2. **Zaprezentowanie zbioru danych *MovieLens 100k*, wykorzystanego do przeprowadzenia eksperymentów**. Sprawdzenie skuteczności zaproponowanego podejścia, będzie realizowane na rzeczywistym zbiorze danych, który jest ogólnodostępny i bardzo popularny w środowisku badawczym, zajmującym się systemami rekomendacyjnymi. W niniejszej pracy zaprezentowana zostanie również krótka analiza tego zbioru, żeby przybliżyć czytelnikowi jego zawartość.

3. **Dostrojenie parametrów poszczególnych algorytmów wchodzących w skład agregacji**. Algorytmy rekomendacyjne posiadają specyficzne parametry, które przed przystąpieniem do procesu rekomendacji, wymagają odpowiedniego dostrojenia. Z tego względu przedstawiony zostanie wpływ poszczególnych parametrów, na uzyskiwane wyniki.

4. **Włączenie do procesu agregacji rankingów innych użytkowników**. W celu poprawy jakości generowanych rekomendacji, zaprezentowana zostanie modyfikacja uwzględniająca w tym procesie rankingi innych użytkowników. Użytkownicy uwzględnieni w tym procesie, zostaną wyznaczeni przy pomocy algorytmu *kNN*.

5. **Sprawdzenie wpływu różnych wariantów funkcji oceny na jakość agregacji**. Funkcja oceny jest kluczowym elementem każdego algorytmu metaheurystycznego. Z tego względu przetestowane zostaną różne warianty tej funkcji, w celu zbadania wpływu jej poszczególnych wariantów, na jakość generowanych rekomendacji.

Część zaprezentowanych w niniejszej rozprawie wyników badań, powstało na bazie wcześniejszych prac autora [21, 22, 36].

1.4 Układ rozprawy

Rozprawa składa się z dziesięciu rozdziałów i została podzielona na dwie główne części: teoretyczną (rozdziały: 2, 3, 4, 5, 6, 7) i badawczą (rozdziały: 8, 9, 10). Na końcu rozprawy znajduje się *dodatek A* (A) oraz *słownik symboli* (10.3).

W rozdziale 2 znajdują się podstawowe informacje dotyczące systemów rekomendacyjnych. Przedstawiona zostanie w nim formalna definicja tych systemów, wraz z ich podstawową klasyfikacją. Dodatkowo omówione zostaną główne wyzwania, z jakimi muszą mierzyć się projektanci. Zaprezentowany zostanie również przegląd wybranych algorytmów rekomendacyjnych. Ponadto w rozdziale 3 opisane zostaną przykładowe miary, które wykorzystywane są do oceny jakości generowanych rekomendacji.

Rozdział 4 poświęcony będzie metaheurystykom, a w szczególności algorytmowi ewolucji różnicowej. W pierwszej kolejności opisane zostaną ogólne informacje dotyczące metaheurystyk, wraz z ich rysem historycznym oraz przykładami zastosowań. Następnie zaprezentowany zostanie przegląd literaturowy, nawiązujący do wykorzystania ich w systemach rekomendacyjnych. Na końcu tego rozdziału znajdzie się omówienie ogólnego schematu algorytmu DE.

W rozdziale 5 opisana zostanie idea agregacji rang. Jest to stosunkowe nowe i mało przebadane podejście, szczególnie w kontekście systemów rekomendacyjnych. Celem tej idei jest połączenie różnych źródeł informacji, zaprezentowanych w formie uporządkowanej listy elementów (rankingu) i utworzenie na ich podstawie nowej listy, która w teorii powinna być *lepsza* od list indywidualnych (bazowych).

Rozdział 6 omawia zagadnienie nazywane w literaturze nauką rangowania (z ang. *learning to rank* [67]). Technika ta wywodzi się głównie z systemów wyszukiwania informacji i polega ona na zastosowaniu algorytmów uczenia maszynowego, zwykle bazujących na uczeniu nadzorowanym do stworzenia modelu rangującego, którego zadaniem jest przewidzenie uporządkowania elementów na liście. Tego typu podejście można również zastosować w systemach rekomendacyjnych, gdzie po utworzeniu takiej listy sugerowanych przedmiotów są one następnie prezentowane aktywnemu użytkownikowi w formie rekomendacji.

Rozdział 7 poświęcony zostanie omówieniu zaproponowanego algorytmu. Przedstawione zostaną w nim różne warianty funkcji oceny, które następnie będą przebadane w fazie eksperymentalnej. Zaprezentowana zostanie również autorska modyfikacja, która w procesie agregacji uwzględnia rankingi innych użytkowników. W tym rozdziale przedstawiona zostanie również architektura zaproponowanego systemu rekomendacyjnego.

W rozdziale 8 przeprowadzona zostanie krótka analiza zbioru danych *MovieLens 100k*, który wykorzystany został do przeprowadzenia badań. Dodatkowo omówione zostaną w nim szczegóły dotyczące dostrajania parametrów algorytmów rekomendacyjnych, wchodzących w skład agregacji. Przedstawione zostanie również środowisko badawcze wraz

z metodologią prowadzenia badań.

Rozdział 9 w całości poświęcony zostanie badaniom eksperymentalnym. Zaprezentowane zostaną badania dotyczące wpływu różnych wariantów funkcji oceny na jakość tworzonej agregacji. Przedstawiony zostanie ponadto wpływ rankingów innych użytkowników oraz rankingów wygenerowanych losowo na jakość agregacji.

Ostatni rozdział (10) jest krótkim podsumowaniem całej rozprawy, gdzie przedstawione zostaną wnioski końcowe. Znajdzie się tam nawiązanie do tezy rozprawy oraz informacje dotyczące realizacji celów dodatkowych. Pod koniec tego rozdziału zaprezentowane zostaną sugestie, dotyczące możliwych modyfikacji zaproponowanego algorytmu oraz kierunki przyszłych prac badawczych.

Rozdział 2

Systemy rekomendacyjne

Ten rozdział poświęcony zostanie omówieniu najważniejszych zagadnień związanych z systemami rekomendacyjnymi. W pierwszej kolejności zaprezentowane zostaną ogólne informacje dotyczące tego typu systemów. Następnie opisana zostanie formalna definicja systemu rekomendacyjnego, wraz z podaniem kilku przykładów zastosowań komercyjnych. W tym rozdziale przedstawiona zostanie również ich ogólna klasyfikacja i omówione zostaną główne wyzwania, z jakimi muszą mierzyć się projektanci. Na końcu tego rozdziału, zaprezentowany zostanie krótki przegląd wybranych algorytmów rekomendacyjnych.

2.1 Przeciążenie informacyjne

Rosnąca ilość dostępnych danych, a tym samym związane z tym liczne trudności użytkowników z ich należytym przefiltrowaniem oraz zrozumieniem jest znanym problemem w dzisiejszym świecie. Często taką sytuację określa się terminem *przeciążenia informacyjnego*, jednak nie ma jednej konkretnej definicji takiej stanu. Oczywiście z łatwością można przytaczać liczby i wskazywać jak ogromną ilością danych obecnie dysponujemy, jednak należy pamiętać o tym, że nie tylko kryteria ilościowe są tutaj istotne, ponieważ dane mogą być również bardzo złożone i różnorodne [19]. W szczególności ostatnie lata pokazały, że przekazywane informacje muszą być nie tylko aktualne, ale również powinny być rzetelne, a spowodowanie zamieszania informacyjnego może przyczynić się do dezorientacji użytkownika i spowodować spadek zaufania do danego systemu [177].

Jednak analiza tego typu danych nie jest trywialna. Wprowadzono nawet termin *Big Data*, który najczęściej odnosi się właśnie do dużych zbiorów danych, które są trudne w analizie i wyciąganiu z nich informacji oraz wiedzy. Problem ten jednak poruszany jest w wielu dziedzinach, bo choć analiza tego typu danych często wymaga wyrafinowanych technik, to mogą one być źródłem cennych informacji [144].

2.2 Informacje ogólne

Jednym z rodzajów systemów, które wspomagają użytkownika w procesie podejmowania decyzji są to systemy rekomendacyjne. Ich zadaniem jest na bazie historycznej aktywności użytkowników, przewidzenie ich przyszłych preferencji. Bez wątpienia są one coraz powszechniej wykorzystywane w różnych dziedzinach naszego życia. Od kupowania przedmiotów na portalach aukcyjnych, przez wybór kolejnego filmu do obejrzenia, po dodawanie nowych znajomych na portalach społecznościowych. Rosnąca popularność tego typu serwisów sprawiła, że istnieje realne zapotrzebowanie na systemy rekomendacyjne działające szybko i podnoszące nie tylko jakość generowanych rekomendacji, ale które zapewniają również ich oryginalność (z ang. *novelty*) i różnorodność (z ang. *diversity*) [47].

W literaturze przyjmuje się, że zadaniem systemu rekomendacyjnego jest zaproponowanie użytkownikowi pewnych przedmiotów, które z dużym prawdopodobieństwem mogą go zainteresować. Przy czym termin *przedmiot* jest tutaj terminem ogólnym, który opisuje pewien obiekt, będący składową systemu informatycznego i jego znaczenie w dużej mierze zależy od domeny, w której system rekomendacyjny jest wykorzystywany. Może on oznaczać np.: produkty, artykuły, filmy, muzykę lub osoby.

W systemach rekomendacyjnych możemy wyróżnić dwa główne podejścia do generowania rekomendacji. Mogą one opierać się na próbie predykcji, jaką ocenę (np. w skali od 1 do 5), wystawiłby użytkownik danemu przedmiotowi w systemie [191]. Mogą też próbować przewidzieć pewien zbiór przedmiotów, najczęściej prezentowany w postaci uporządkowanej listy, które zostałyby polecone użytkownikowi. Problem ten w literaturze nosi nazwę problemu top-N rekomendacji (z ang. *Top-N recommendation problem*) [89].

Dodatkowo rozróżniamy dwa główne sposoby pobierania danych: jawny i niejawny. Jawna informacja zwrotna pozyskiwana jest od użytkownika w sposób bezpośredni. Użytkownik wprowadza do systemu konkretne informacje, które wyrażają stopień preferencji względem danego przedmiotu, przykładowo wystawiając mu ocenę, wprowadzając komentarz lub pisząc recenzję. Zasadniczą zaletą tego typu informacji zwrotnej jest to, że w łatwiejszy sposób możemy stwierdzić, czy interakcja użytkownika z danym przedmiotem była pozytywna, czy też negatywna. Przykładowo, jeżeli użytkownik ma możliwość wprowadzenia do systemu oceny w skali od 1 do 5 i wybierze ocenę 5, to z dużym prawdopodobieństwem można założyć, że jest to przedmiot, który użytkownik lubi [88].

Niejawna informacja zwrotna pozyskiwana jest poprzez analizę zachowania użytkownika w systemie, czyli np. kliknięcia w konkretny produkt, wyświetlenia strony, dodania przedmiotu do koszyka. Ten typ informacji zwrotnej jest łatwiejszy do pozyskania, ze względu na to, że nie ma potrzeby proszenia użytkownika o wejście w interakcję z systemem (np. skomentowania, ocenienia przedmiotu itp.). Zasadniczą wadą takiego podejścia jest brak informacji o tym, czy interakcja z danym przedmiotem była pozytywna czy też negatywna [123]. Przykładowo użytkownik mógł dodać przedmiot do koszyka przez

przypadek, a sam fakt otwarcia jakiejś strony nie świadczy o tym, że użytkownik dany przedmiot lubi. Z tego względu implementacja systemów bazujących na tym typie danych wiąże się z szeregiem wyzwań, a w literaturze zaproponowano wiele specjalnych algorytmów [139], które są dostosowane do tego typu danych.

Systemy rekomendacyjne możemy podzielić również na spersonalizowane i niespersonalizowane. Niespersonalizowany system rekomendacyjny to taki, który na podstawie globalnych zachowań wszystkich użytkowników w systemie próbuje wyciągnąć jakieś wnioski, przykładowo rekomendując użytkownikowi filmy, które są najczęściej oglądane. Obecnie jednak przeważnie wykorzystuje się systemy spersonalizowane, które na podstawie historycznej aktywności danego użytkownika tworzą jego profil preferencji, który jest następnie wykorzystywany do generowania rekomendacji [95].

2.3 Formalna definicja systemu rekomendacyjnego

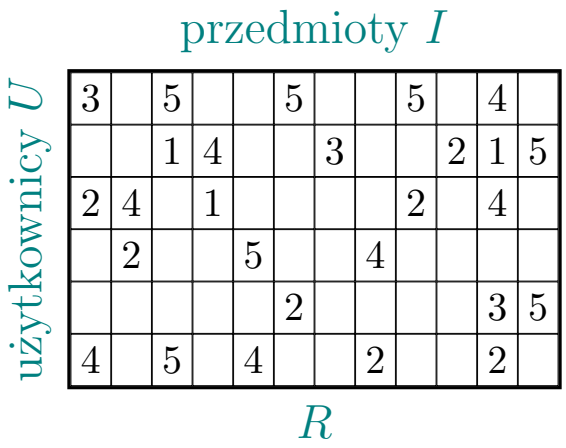
W systemie rekomendacyjnym wyróżniamy pewien zbiór użytkowników $U = \{u_1, u_2, \ldots, u_{|U|}\}$ i pewien zbiór przedmiotów $I = \{x_1, x_2, \ldots, x_{|I|}\}$. Wszystkie interakcje pomiędzy użytkownikami i przedmiotami są zapisane w macierzy R (rys. 2.1) i mogą one przyjmować wartości ze zbioru L (np. $L = \{0, 1\}$). Gdy dany użytkownik $u \in U$ wejdzie w interakcję z przedmiotem $x \in I$, to taka interakcja będzie oznaczana jako r_{ux}. Dane mogą być reprezentowane więc jako trójka $(u,\ x,\ r_{ux})$. Dodatkowo podzbiór użytkowników, którzy ocenili przedmiot x, będzie oznaczany jako U_x i analogicznie podzbiór przedmiotów, które zostały ocenione przez użytkownika u, będzie oznaczany jako I_u [6].

W sytuacji, gdy w systemie mamy dostępne oceny, jakie użytkownicy wystawili poszczególnym przedmiotom (np. oceny wystawione obejrzanym filmom w skali od 1 do 5), to najczęściej problem rekomendacji przedstawiany jest jako problem regresji lub klasyfikacji, którego celem jest zamodelowanie funkcji, która przewiduje ocenę $f(u, x)$, jaką użytkownik u, wystawił przedmiotowi x [59]. Następnie funkcja ta jest wykorzystywana do zarekomendowania aktywnemu użytkownikowi u_A przedmiotu x^*, którego przewidywana ocena ma największą wartość:

$$x^* = \arg\max_{x \in I \setminus I_u} f(u_A, x). \tag{2.1}$$

2.4 Wykorzystanie komercyjne

Systemy rekomendacyjne bez wątpienia odniosły sukces w zastosowaniach komercyjnych i obecnie wiele znanych serwisów internetowych wykorzystuje technologię rekomendacji [10]. Jedną z pierwszych firm, która wprowadziła tą technologię na większą skalę była firma *Amazon* [110]. Do generowania rekomendacji, wykorzystali oni prostą technikę bazującą na obliczaniu podobieństwa pomiędzy przedmiotami występującymi w systemie,



3		5			5			5		4	
		1	4			3			2	1	5
2	4		1					2		4	
	2			5			4				
					2					3	5
4		5		4			2			2	

Rysunek 2.1: Macierz ocen użytkownik-przedmiot
Opracowanie własne

a same rekomendacje wyświetlały się na głównej stronie ich serwisu aukcyjnego oraz podczas przeglądania poszczególnych produktów. W ślad za tą firmą poszły inne znane marki, takie jak np. *eBay*, który wprowadził rekomendacje wyświetlające się po tym, jak użytkownik dokonał zakupu danego przedmiotu, co umożliwiało polecenie mu innych powiązanych produktów [174].

Innym rodzajem serwisu internetowego, który wykorzystuje mechanizm rekomendacji, jest np. platforma *YouTube* [56]. Treści są tam prezentowane głównie w postaci przesyłanych strumieniowo filmów wideo, co rodzi liczne wyzwania związane z analizą tego typu zawartości. Wykorzystuje się w tym celu techniki analizy obrazu, które umożliwiają ekstrakcje cech, które następnie mogą być wykorzystywane do scharakteryzowania poszczególnych filmów i uwzględnione w procesie generowania rekomendacji [57].

Kolejnym obszarem, w którym z powodzeniem wykorzystano systemy rekomendacyjne jest branża muzyczna. Takie portale jak *Spotify* czy *Apple Music*, podsuwają użytkownikowi kolejne utwory na bazie analizy jego wcześniej przesłuchanych piosenek, co umożliwia skonstruowanie profilu preferencji muzycznych danego użytkownika. Udostępniają one również funkcjonalności, umożliwiające słuchanie innych popularnych utworów, które są obecnie odtwarzane przez innych użytkowników. Rekomendowane mogą być również grupy utworów związanych z danych gatunkiem lub artystą. Dodatkowo generowanie rekomendacji odbywa się nie tylko na bazie analizy preferencji poszczególnych użytkowników, ale wykorzystywane są tutaj dużo bardziej wyrafinowane techniki, umożliwiające analizę sygnału audio oraz powiązanych z utworami metadanych [103].

Niestety trudno jest przytoczyć dokładne statystyki, dotyczące realnego wpływu, jaki systemy rekomendacyjne mają na np. zwiększenie sprzedaży, ponieważ firmy niechętnie dzielą się takimi danymi. Część uzyskanych informacji pochodzi z blogów technologicznych, gdzie twórcy dzielą się pewnymi spostrzeżeniami, dotyczącymi działania tego typu systemów w swoich firmach. Przykładowo w poście [189], napisanym przez pracowników

firmy *Netflix* autorzy wskazali, że 75% oglądanych filmów na ich platformie pochodzi z rekomendacji. Autorzy zaznaczyli, że poziom ten udało się im osiągnąć dzięki ciągłej ewaluacji poziomu satysfakcji użytkowników i projektowanie modyfikacji w taki sposób, aby sprostać ich oczekiwaniom.

Pewną ciekawą obserwacją, świadczącą o istotności zastosowania tego typu technologii w biznesie jest to, że zazwyczaj poświęca się im dosyć dużą ilość miejsca na portalach internetowych i serwisach streamingowych [86]. Przykładowo na platformie *Netflix*, filmy wyświetlające się na głównej stronie w większości pochodzą z rekomendacji. Na portalach aukcyjnych po zakupie danego przedmiotu, możemy zobaczyć inne sugerowane przedmioty, które zostały kupione razem z nim. Podczas dodawania nowego znajomego na portalu społecznościowym, wyświetla się nam lista innych potencjalnych znajomych.

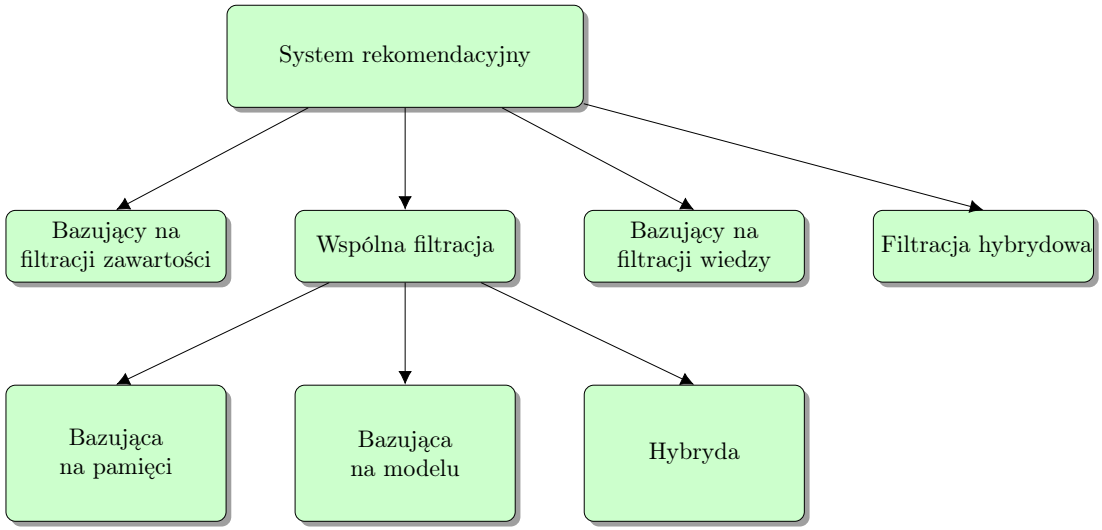
Rekomendacje stały się czymś tak powszechnym, że obecnie ciężko jest wyobrazić sobie korzystanie z serwisów internetowych, które nie wykorzystywałyby tego typu technologii. Choć my, jako użytkownicy, nie zawsze zwracamy na to uwagę.

2.5 Klasyfikacja systemów rekomendacji

W systemach rekomendacyjnych wyróżniamy następujące podejścia do generowania rekomendacji: filtracja bazująca na zawartości, wspólna filtracja, filtracja bazująca na wiedzy i złączenie różnych technik, czyli podejście hybrydowe [101]. Metody bazujące na filtracji zawartości, wykorzystują cechy opisujące poszczególne przedmioty w systemie i na podstawie wybranej miary podobieństwa, wyszukują najbardziej podobne przedmioty, które następnie są rekomendowane użytkownikowi. W metodzie wspólnej filtracji dla aktywnego użytkownika wyszukuje się innych użytkowników, którzy w podobny sposób wchodzą w interakcje, ze wspólnie ocenionymi przedmiotami (np. podobnie oceniają obejrzane filmy).

Pewnym nowym i coraz bardziej popularnym podejściem jest zastosowanie systemów bazujących na wiedzy, bo choć posiadanie bazy wiedzy wiąże się z licznymi wyzwaniami, to systemy tego typu umożliwiają rozwiązywanie specyficznych problemów, z którymi nie radzą sobie techniki opisane wcześniej. W praktyce jednak każda z opisanych metod posiada pewne wady i w komercyjnych zastosowaniach najczęściej wykorzystywane są podejścia hybrydowe.

Opisane wcześniej metody nie wyczerpują wszystkich podejść, które zaproponowano w literaturze. Dodatkowo można wyróżnić techniki wykorzystujące: kontekst [4], ontologię [92], sieci społeczne [162], zaufanie użytkowników [130], bazujące na sesji [176] lub biorące pod uwagę kilka kryteriów [78]. W kolejnych podrozdziałach omówione zostaną jednak tylko podstawowe techniki, a ogólny podział tych systemów został zaprezentowany na rysunku 2.2.



Rysunek 2.2: Podział algorytmów rekomendacyjnych
Opracowanie własne

2.5.1 Oszacowania bazowe

Przed przejściem do opisu poszczególnych technik wykorzystywanych w systemach rekomendacji, warto na początku wspomnieć o *oszacowaniach bazowych* [99]. Są to metody, które są wykorzystywane głównie w systemach niespersonalizowanych lub w sytuacji, kiedy nie posiadamy żadnych informacji na temat aktywnego użytkownika u_A. Najprostszym oszacowaniem bazowym jest oszacowanie, które przewiduje jaką ocenę użytkownik u_A, wystawiłby przedmiotowi x, według następującego wzoru:

$$b_{u_A,x} = \mu, \tag{2.2}$$

gdzie μ jest to średnia wartość oceny, jaka została wystawiona przez wszystkich użytkowników w U. Oszacowania bazowe mogą uwzględniać tylko średnią ocen wystawioną przez danego użytkownika $\bar{r}_{u_A}$ lub średnią ocen, wystawioną przez wszystkich użytkowników w systemie, ale tylko dla danego przedmiotu $\bar{r}_x$, czyli:

$$b_{u_A,x} = \bar{r}_{u_A}, \tag{2.3}$$

$$b_{u_A,x} = \bar{r}_x. \tag{2.4}$$

Często wykorzystuje się oszacowanie bazowe, zapisane w następującej formie:

$$b_{u_A,x} = \mu + b_{u_A} + b_x, \tag{2.5}$$

gdzie b_{u_A} i b_x są to oszacowania bazowe dla użytkownika u_A i przedmiotu x.

2.5.2 Metody bazujące na zbiorowej filtracji

Metoda zbiorowej filtracji (z ang. *collaborative filtering*) jest to jedna z najbardziej popularnych metod, którą opisano już w 1994 roku w pracy [140]. W technice tej, algorytmy dzielimy na trzy główne kategorie: metody wykorzystujące pamięć (z ang. *memory–based collaborative filtering*), metody wykorzystujące model (z ang. *model–based collaborative filtering*) oraz metody hybrydowe (z ang. *hybrid recommenders*) [161].

Metody bazujące na pamięci wykorzystują historyczne oceny, które zostały wystawione przez aktywnego użytkownika u_A do obliczenia podobieństwa pomiędzy innymi użytkownikami (z ang. *user–based*) lub przedmiotami (z ang. *item–based*). Głównym wyzwaniem w tej technice jest właściwe dobranie miary podobieństwa, ponieważ na jej podstawie określane będzie sąsiedztwo N.

W metodach wykorzystujących pamięć i bazujących na obliczaniu podobieństwa pomiędzy użytkownikami, wyszukuje się pewną grupę użytkowników, którzy zostaną wykorzystani do wygenerowania rekomendacji dla aktywnego użytkownika u_A. Metoda ta bazuje na prostej idei, że z większym prawdopodobieństwem będziemy preferować przedmioty, które zostały wybrane przez inne, podobne do nas osoby lub których sposób oceniania jest zbliżony do naszego.

Do określenia podobieństwa pomiędzy użytkownikami wykorzystuje się funkcję podobieństwa $s : U \times U \to \mathbb{R}$, która następnie jest wykorzystywana we wzorze (2.6) jako waga do określenia stopnia, w jakim oceny poszczególnych użytkowników w zbiorze N, będą wpływać na przewidywaną ocenę. W celu predykcji jaką ocenę użytkownik u_A, wystawi przedmiotowi x, w pierwszej kolejności oblicza się podobieństwo pomiędzy użytkownikiem u_A, a jego sąsiadami w zbiorze N. Następnie oblicza się średnią ważoną na podstawie następującego wzoru:

$$p_{u_A,x} = \bar{r}_{u_A} + \frac{\sum_{u \in N} s(u_A, u)(r_{u,x} - \bar{r}_u)}{\sum_{u \in N} |s(u_A,\ u)|}, \tag{2.6}$$

gdzie:

$r_{u,x}$ – ocena wystawiona przez użytkownika u dla przedmiotu x,

$\bar{r}_{u_A}$ – średnia wszystkich ocen wystawionych przez użytkownika u_A,

$\bar{r}_u$ – średnia wszystkich ocen wystawionych przez użytkownika u,

$s(u_A, u)$ – funkcja podobieństwa pomiędzy użytkownikiem u_A, a użytkownikiem $u \in N$,

N – zbiór sąsiadów $N \subseteq U$.

Kolejną kwestią jest dobranie liczby użytkowników, którzy powinni być uwzględnieni w sąsiedztwie N. W niektórych systemach zaproponowano wykorzystanie wszystkich

użytkowników, czyli $N = U$. Jednak zauważono, że użytkownicy z małym stopniem korelacji wprowadzają zbyt dużo szumu informacyjnego [80]. Należy również pamiętać, że komercyjne systemy rekomendacyjne przetwarzają miliony użytkowników, a rekomendacje muszą być generowane w czasie rzeczywistym. Z tego względu bardziej wskazanym jest ograniczenie sąsiedztwa do pewnej określonej liczby k użytkowników lub zawężenie liczby użytkowników, poprzez określenie minimalnego progu stopnia podobieństwa pomiędzy nimi.

Do obliczenia podobieństwa pomiędzy użytkownikami można wykorzystać współczynnik korelacji Pearsona, który określa poziom zależności liniowej między zmiennymi losowymi, a wartość tego współczynnika mieści się w przedziale $[-1, 1]$. W kontekście systemów rekomendacyjnych, oblicza się stopień korelacji pomiędzy użytkownikami u oraz v w następujący sposób:

$$s\left(u,v\right) = \frac{\sum_{x \in I_u \cap I_v} \left(r_{u,x} - \bar{r}_u\right)\left(r_{v,x} - \bar{r}_v\right)}{\sqrt{\sum_{x \in I_u \cap I_v} \left(r_{u,x} - \bar{r}_u\right)^2} \sqrt{\sum_{x \in I_u \cap I_v} \left(r_{v,x} - \bar{r}_v\right)^2}}, \tag{2.7}$$

gdzie:

I_u, I_v – zbiory przedmiotów ocenionych przez użytkowników u oraz v,

$r_{u,x}$, $r_{v,x}$ – ocena wystawiona przedmiotowi x przez użytkownika u oraz v,

$\bar{r}_u$, $\bar{r}_v$ – średnia wszystkich ocen wystawionych przez użytkownik u oraz v.

Innym sposobem na obliczenie podobieństwa pomiędzy użytkownikami jest wykorzystanie miary bazującej na algebrze liniowej i obliczeniach w przestrzeni wektorowej [65]. Przykładem takiej miary jest odległość kosinusowa, która wykorzystuje reprezentację wektorową użytkowników do obliczania podobieństwa i wyrażana jest następującym wzorem:

$$s(u,v) = \frac{r_u \cdot r_v}{||r_u||_2 ||r_v||_2} = \frac{\sum_x r_{u,x} r_{v,x}}{\sqrt{\sum_x r_{u,x}^2} \sqrt{\sum_x r_{v,x}^2}}, \tag{2.8}$$

gdzie:

r_u, r_v – wektory ocen użytkowników u oraz v,

$||r_u||_2$, $||r_v||_2$ – norma L2 dla wektorów r_u, r_v,

$r_{u,x}$, $r_{v,x}$ – oceny wystawione przez użytkowników u oraz v dla przedmiotu x.

Do generowania rekomendacji w metodach wykorzystujących pamięć, można również wykorzystać podejście bazujące na podobieństwie przedmiotów [147]. Przed przystąpieniem do obliczenia $p_{u,x}$, należy wybrać pewien zbiór Z najbardziej podobnych przedmiotów do przedmiotu x. Następnie na bazie tych przedmiotów należy obliczyć średnią ważoną zgodnie ze wzorem:

$$p_{u,x} = \frac{\sum_{h \in Z} r_{u,h} s(x,h)}{\sum_{h \in Z} |s(x,h)|}, \tag{2.9}$$

gdzie:

$r_{u,h}$ – jest to ocena wystawiona przedmiotowi h przez użytkownika u,

$s(x,h)$ – funkcja podobieństwa pomiędzy przedmiotami x oraz h,

Z – zbiór przedmiotów podobnych do przedmiotu x.

Analogicznie, tak jak w przypadku obliczania podobieństwa pomiędzy użytkownikami, wykorzystywane są tutaj pewne miary określające stopień podobieństwa pomiędzy przedmiotami. Do obliczenia podobieństwa pomiędzy dwoma przedmiotami x i h, można wykorzystać miarę korelacji Pearsona:

$$s\left(x,h\right) = \frac{\sum_{u \in U_x \cap U_h} \left(r_{u,x} - \bar{r}_x\right)\left(r_{u,h} - \bar{r}_h\right)}{\sqrt{\sum_{u \in U_x \cap U_h} \left(r_{u,x} - \bar{r}_x\right)^2} \sqrt{\sum_{u \in U_x \cap U_h} \left(r_{u,h} - \bar{r}_h\right)^2}}, \tag{2.10}$$

gdzie:

U_x, U_h – zbiory użytkowników, którzy ocenili przedmioty x oraz h,

$r_{u,x}$, $r_{u,h}$ – ocena wystawiona przez użytkownika u dla przedmiotów x oraz h,

$\bar{r}_x$, $\bar{r}_h$ – średnie ocen wystawionych dla przedmiotów x oraz h.

Do obliczania podobieństwa można również wykorzystać odległość kosinusową. Pomiędzy wektorami ocen dla przedmiotów x oraz h obliczenia realizuje się zgodnie z następującym wzorem:

$$s(x,h) = \frac{r_x \cdot r_h}{||r_x||_2 ||r_h||_2} = \frac{\sum_u r_{u,x} r_{u,h}}{\sqrt{\sum_u r_{u,x}^2} \sqrt{\sum_u r_{u,h}^2}}, \tag{2.11}$$

gdzie:

r_x, r_h – wektory ocen dla przedmiotu x oraz h,

$||r_x||_2$, $||r_h||_2$ – norma L2 dla wektorów r_x, r_h,

$r_{u,x}$, $r_{u,h}$ – oceny wystawione przez użytkownika u dla przedmiotów x oraz h.

Do generowania rekomendacji można wykorzystać również metody bazujące na modelu, których celem jest na bazie danych treningowych, wychwycenie pewnych wzorców zachowań użytkowników i ich generalizację [39]. Przykładami technik wykorzystujących to podejście są techniki: grupowania [27], faktoryzacji macierzy [98] oraz prostego klasyfikatora bayesowskiego [118]. Choć algorytmy bazujące na tym podejściu lepiej radzą sobie z takimi problemami jak rzadkość danych i skalowalność, to ich zasadniczą wadą jest to, że są dużo bardziej złożone i trudniejsze w poprawnej implementacji.

2.5.3 Metody bazujące na filtracji zawartości

Idea tej metody jest podobna do metod wykorzystujących zbiorową filtrację. W odróżnieniu jednak od zbiorowej filtracji, nie wykorzystuje się tutaj informacji na temat innych użytkowników, a jedynie bazuje się na cechach (atrybutach), które opisują poszczególne przedmioty [132]. Przykładem takiego systemu może być wypożyczalnia filmów, gdzie klientowi polecane są filmy z udziałem jego ulubionego aktora, lub które zostały nakręcone przez tego samego reżysera. W tej technice często wykorzystuje się również analizę opisu danego przedmiotu lub wystawionych recenzji, ale wtedy niezbędne jest zastosowanie technik służących do ekstrakcji cech z tego typu danych [108].

Metoda ta posiada swoje zalety. Przede wszystkim umożliwia ona rozwiązanie problemu zimnego startu [109], do której dochodzi najczęściej podczas zakładania nowego konta, gdzie nie posiadamy żadnych informacji na temat aktywnego użytkownika. Możemy wtedy podczas rejestracji poprosić go o wypełnienie krótkiej ankiety, w której określi swoje preferencje dotyczące przedmiotów występujących w systemie. Na tej podstawie zostanie stworzony jego wstępny profil, który posłuży do wygenerowania rekomendacji. Dodatkowo metoda ta umożliwia rozwiązanie problemu, w którym do systemu został dodany nowy przedmiot, a który nie został jeszcze oceniony przez innych użytkowników.

Podstawową wadą tej metody jest to, że bazuje ona tylko na cechach (atrybutach) przedmiotów. Z tego względu cechy te muszą zostać przygotowane z najwyższą starannością, a to często wymaga zatrudnienia eksperta domenowego, który odpowiednio przeanalizuje istniejące cechy w systemie i usunie cechy niskiej jakości lub utworzy nowe. Dodatkowo metodę tą cechuje nadmierna specjalizacja w generowanych rekomendacjach, a system rekomendacyjny rzadko rekomenduje coś nowego lub zaskakującego [47]. Jest to spowodowane tym, że system wyszukuje przedmioty jak najbardziej do siebie podobne, co z czasem prowadzi do rekomendowania takich samych przedmiotów. Do określania podobieństwa pomiędzy przedmiotami można wykorzystać standardowe miary podobieństwa.

2.5.4 Metody bazujące na wiedzy

Metody bazujące na wiedzy generują rekomendacje opierając się na dostępnej wiedzy na temat przedmiotów i użytkowników. Metoda ta jest szczególnie użyteczna w przypadku, kiedy użytkownicy weszli w interakcję z niewielką liczbą przedmiotów. Zastosowanie w takiej sytuacji innych technik opisanych w poprzednich podrozdziałach (np. techniki wspólnej filtracji), może okazać się nieskuteczne, ze względu na niewielką ilość dostępnych danych.

Przykładowo, większość użytkowników kupujących samochody lub mieszkania robi to tak sporadycznie, że ciężko jest na podstawie takiej aktywności zbudować dobry profil preferencji użytkownika. Ponadto same przedmioty mogą posiadać wiele specyficznych cech, których odpowiednie odfiltrowanie może być kluczowe dla użytkownika. Przykładowo

klienci kupujący samochód nie chcą kupić dowolnego samochodu, który im system rekomendacyjny zaproponuje, tylko mają bardzo specyficzne wymagania, które dany pojazd musi spełniać (np. kolor, liczba drzwi, pojemność bagażnika). Powoduje to, że w takich systemach użytkownicy z dużo większym prawdopodobieństwem będą aktywnie korzystać z systemu rekomendacyjnego, wybierając dostępne kryteria lub uzupełniając wybrane pola, aby zawęzić wybór do swoich upodobań. Z tego względu niezwykle ważne jest właściwe zaprojektowanie interfejsu użytkownika, aby interakcja z użytkownikiem przechodziła bez zakłóceń.

Podsumowując, wykorzystanie systemów rekomendacyjnych opartych na wiedzy jest szczególnie uzasadnione w sytuacjach, kiedy użytkownicy mogą bezpośrednio określić swoje wymagania dotyczące rekomendowanych produktów, a same produkty są rzadko wybierane przez użytkowników i posiadają wiele specyficznych cech [8].

2.5.5 Metody hybrydowe

Chociaż w poprzednich rozdziałach opisane zostały pojedyncze techniki wykorzystywane do generowania rekomendacji, to w praktyce często wykorzystuje się wiele połączonych ze sobą strategii. Nazywane jest to *podejściem hybrydowym*. Metody hybrydowe mają na celu usprawnienie działania pojedynczych algorytmów, które choć posiadają różne wady, to kolektywnie mogą generować rekomendację lepszej jakości. Przykładowo w konkursie *Netflix Prize* nagrodę dodatkową wygrał zespół, który podniósł wskaźnik jakości rekomendacji (RMSE) o 9% i aby to osiągnąć zespół ten połączył ze sobą aż 107 algorytmów [24]. W tabeli 2.1 zaprezentowane zostały podstawowe metody hybrydyzacji.

Tabela 2.1: Przegląd podstawowych metod hybrydyzacji [45]

Metoda hybrydyzacji	Opis
Ważona	Wyniki rekomendacji z kilku technik są łączone razem w celu uzyskania pojedynczej rekomendacji
Przełączania	System zmienia wykorzystywaną technikę rekomendacji w zależności od sytuacji
Mieszana	Użytkownikowi w tym samym momencie prezentowane są rekomendacje wygenerowane przez kilka algorytmów
Kombinacji cech	Cechy z różnych metod rekomendacyjnych są łączone
Kaskady	Kolejne algorytmy rekomendacyjne przetwarzają wygenerowane rekomendacje
Rozszerzająca cechy	Obliczanie cechy lub zestawu cech, które następnie stanowią część danych wejściowych do następnej techniki
Meta-poziom	W wyniku zastosowania jednej techniki rekomendacji powstaje model, który wykorzystywany jest przez inną technikę

2.6 Główne wyzwania w systemach rekomendacji

Od samego początku istnienia systemów rekomendacyjnych mierzono się z licznymi problemami, które choć zostały już w większości zaadresowane w literaturze [18, 119, 68], to nadal stanowią główne wyzwania, z którymi muszą mierzyć się projektanci. System rekomendacyjny musi nie tylko generować rekomendację na podstawie bardzo rzadkich danych (zazwyczaj ponad 99%), ale musi to robić szybko i dokładnie dla dużej liczby użytkowników.

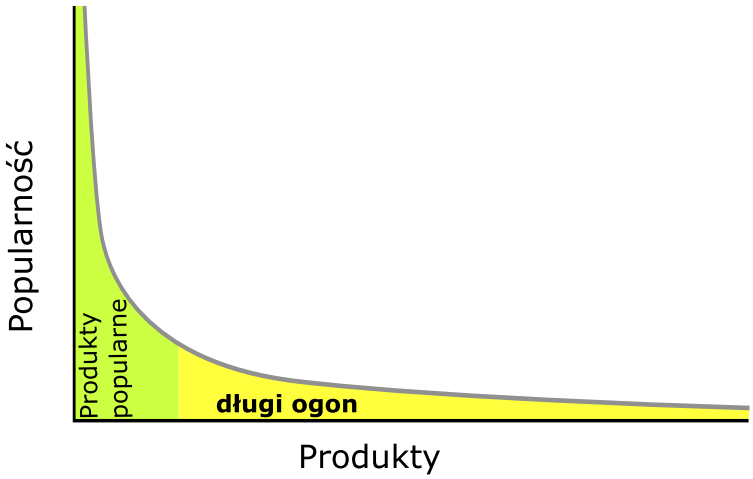
Istotnym problemem jest również problem zimnego startu, czyli sytuacji, kiedy system musi wygenerować rekomendację na podstawie szczątkowych informacji o użytkowniku lub nawet ich braku. Dodatkowo chęć ciągłego poprawiania jakości generowanych rekomendacji, wiąże się z wykorzystaniem coraz to większej ilości informacji, które pochodzą z różnych źródeł. Oprócz samych trudności z przetwarzaniem tego typu danych, dochodzi również problem ich odpowiedniego przechowywania z zachowaniem prywatności użytkowników.

Nie ma jednego, uniwersalnego algorytmu, który rozwiązywałby wszystkie przedstawione wcześniej problemy. Z tego względu najbardziej popularnymi rozwiązaniami stosowanymi w kontekście systemów rekomendacyjnych są tzw. *rozwiązania hybrydowe*. W kolejnych podrozdziałach zaprezentowany zostanie krótki opis najważniejszych problemów i wyzwań, a ich pełny opis można znaleźć w pracy [142].

2.6.1 Rzadkość danych

Jednym z największych wyzwań występującym w systemach rekomendacji jest rzadkość danych [11]. Użytkownicy oceniają tylko niewielką część przedmiotów, która znajdują się w systemie, co powoduje że macierz ocen jest bardzo rzadka (większość wartości, to wartości puste). Wiążą się z tym liczne trudności, ponieważ duża liczba przedmiotów jest oceniona tylko kilka razy, przez co są mało popularne i trudne do zarekomendowania przez system. Powstaje przez to problem tzw. długiego ogona (z ang. *long tail*), czyli sytuacji, w której większość przedmiotów istniejących w systemie jest kupowana bardzo rzadko. Przykładowa wizualizacja tego zjawiska została zaprezentowana na rysunku 2.3.

Dobry system rekomendacyjny generuje jednak rekomendację różnorodne, gdzie przedmioty, które zostały ocenione przez niewielką liczbę użytkowników, również mają szansę zostać zarekomendowane. Ma to nie tylko znaczenie dla samych użytkowników, którzy dzięki temu mogą odkrywać nowe przedmioty, ale również ma to znaczenie dla firmy implementującej dany system rekomendacyjny, ponieważ dzięki temu można zwiększyć sprzedawany asortyment. Ciekawą pracą w tej tematyce jest książka [14], gdzie autorzy dokładnie omawiają to zagadnienie z perspektywy biznesowej i jak wpływa ono na aktualną ekonomię.



Rysunek 2.3: Rysunek przedstawiający koncepcję *długiego ogona*
Opracowanie własne

Aby przeciwdziałać problemowi rzadkości danych, jedną z podstawowych technik jest technika redukcji wymiaru macierzy, która umożliwia reprezentowanie przedmiotów i użytkowników przez małą liczbę cech ukrytych (ang. *latent features*). Ogólna idea tej metody została opisana w podrozdziale 2.7.1. Aktualny przegląd literaturowy dotyczący prac związanych z problemem rzadkości danych w systemach rekomendacji można znaleźć w [84].

2.6.2 Skalowalność i prywatność

Ze względu na to, że macierz ocen jest bardzo rzadka, to systemy rekomendacyjne często zbierają dodatkowe informacje na temat użytkowników, a sam zakres przetwarzanych tych danych jest bardzo zróżnicowany [72]. Przykładowo odwiedzając portale internetowe lub korzystając z aplikacji mobilnych, często jesteśmy informowani na temat zakresu zbieranych o nas danych, chociaż większość użytkowników nie przykłada do tego specjalnej uwagi. Dla korporacji jest to jednak niezwykle cenne źródło informacji, dzięki któremu mogą lepiej poznać zachowania użytkowników korzystających z danego systemu i w efekcie podsuwać bardziej spersonalizowane treści.

Jednak wraz z dynamicznym wzrostem ilości dostępnych danych, pojawia się problem ich efektywnego przetwarzania. Z tego względu ważne jest projektowanie systemów, które nie tylko będą generować rekomendację wysokiej jakości, ale które będą również robiły to dostatecznie szybko, aby nie zaburzyć interakcji użytkownika z systemem rekomendacyjnym. Przykładowo algorytmy bazujące na metodzie wspólnej filtracji, analizują zachowanie innych użytkowników w celu wygenerowania trafnych rekomendacji. W sytuacji, gdy na portalach internetowych znajdują się miliony kont użytkowników, obliczenie miary podobieństwa pomiędzy wszystkimi parami użytkowników może zająć znaczną ilość czasu. Jednym ze sposobów radzenia sobie z tym problemem jest zawężenie liczby obiektów

(użytkowników lub przedmiotów), na których dokonywane będą obliczenia. Można to zrealizować poprzez wykorzystanie algorytmów grupujących, które redukują liczbę obiektów na podstawie których obliczana będzie miara podobieństwa [148].

Warto tutaj wspomnieć również o tym, że zbieranie takiej ilości danych wiąże się z licznymi wyzwaniami, dotyczącymi ich właściwego przechowywania oraz przetwarzania. Z tego względu mogą one stanowić zagrożenie dla prywatności użytkowników w sytuacji, gdy osoby niepowołane uzyskają do nich dostęp. W pracy [72] autorzy dokonali szczegółowego przeglądu istniejących zagrożeń oraz aktualnych rozwiązań tego problemu, wraz z analizą jaki wpływ mają one na użytkowników.

2.6.3 Problem zimnego startu

Problem zimnego startu jest dobrze znany w środowisku badawczym związanym z systemami rekomendacyjnym i występuje on w sytuacji, gdy nie posiadamy wystarczającej ilości danych na temat interakcji, które zachodzą pomiędzy użytkownikami i przedmiotami. Chociaż techniki takie jak filtracja na podstawie zawartości lub wiedzy mogą być bardziej odporne na ten problem, to potrzebują one specyficznych danych, które nie zawsze są dostępne. W systemach rekomendacji możemy wyróżnić dwa główne typy zimnego startu:

- Dla przedmiotów – mogą to być przedmioty, które zostały dopiero dodane do systemu lub które istnieją w systemie od wielu miesięcy, ale zostały ocenione przez niewielką grupę użytkowników. Ta niewielka liczba interakcji powoduje, że przedmioty te są rekomendowane z dużo mniejszym prawdopodobieństwem. Algorytmy bazujące na wspólnej filtracji, obliczają podobieństwo pomiędzy użytkownikami zazwyczaj wykorzystując do tego celu wspólnie ocenione przedmioty. Problem ten nosi również nazwę *stronniczość popularności* (z ang. *popularity bias*), gdzie przedmioty, które są częściej oceniane przez użytkowników, będą rekomendowane z większym prawdopodobieństwem [3].

- Dla użytkowników – użytkownicy podczas zakładania nowego konta nie posiadają w systemie żadnych interakcji. Powoduje to generowanie rekomendacji, które nie są spersonalizowane i jeżeli modelowanie preferencji użytkownika będzie trwało zbyt długo, może to doprowadzić do jego zniechęcenia. Z tego względu zaproponowano specjalne miary, które mimo niewielkiej aktywności użytkownika, potrafią stosunkowo dobrze odwzorować jego początkowe preferencje [32]. Inną strategią jest wykorzystanie pewnej wstępnej ankiety, którą użytkownik wypełnia podczas zakładania nowego konta i wskazuje w niej pewien zestaw preferencji, dotyczący przedmiotów znajdujących się w systemie. Oczywiście proces ten również nie może być zbyt długi, aby wprowadzone przez niego dane były rzetelne [138].

Aby wyjść naprzeciw problemowi zimnego startu w literaturze zaproponowano wiele wyrafinowanych technik [149, 109, 165], które w większości bazują na podejściu hybrydowym, a pełny przegląd aktualnej literatury znajduje się w [131].

2.7 Przegląd wybranych algorytmów rekomendacyjnych

W rozdziale 2.5 przedstawione zostały podstawowe algorytmy wykorzystywane do generowania rekomendacji dla aktywnego użytkownika. Przeważnie analizują one macierz ocen R i wyszukują podobnych użytkowników lub podobne przedmioty, na podstawie których generowane są następnie rekomendacje. W tym rozdziale zaprezentowane zostaną bardziej zaawansowane techniki, które w większości bazują na idei faktoryzacji macierzy R. W pierwszej kolejności omówiona zostanie ogólna idea algorytmów tego typu, a następnie przedstawione zostaną algorytmy wykorzystujące tą technikę.

2.7.1 Rozkład macierzy według wartości osobliwych

Metoda rozkładu macierzy według wartości osobliwych (ang. *singular value decomposition*, SVD) jest to metoda, stosowana m. in. w analizie statystycznej, służąca do redukcji wymiaru macierzy. Macierz rzeczywistą A można przedstawić w postaci następującego rozkładu:

$$A = U\Sigma V^T. \tag{2.12}$$

gdzie:

U i V – macierze ortogonalne ($U^{-1} = U^T,\ V^{-1} = V^T$),

Σ – macierz diagonalna, taka że $\Sigma = diag(\sigma_i)$, gdzie σ_i oznacza nieujemne wartości szczególne (osobliwe) macierzy A.

Powyższej metody nie możemy zastosować bezpośrednio na macierzy R, ponieważ w systemach rekomendacji macierz ta jest niekompletna (zawiera wartości puste). Jest to spowodowane tym, że użytkownicy $u \in U$ ocenili tylko niewielką część przedmiotów $x \in I$. Z tego względu faktoryzację dokonuje się poprzez zastosowanie pewnych algorytmów, które generują najlepsze przybliżenie (aproksymację) oryginalnej macierzy. Można to wyrazić następującym wzorem:

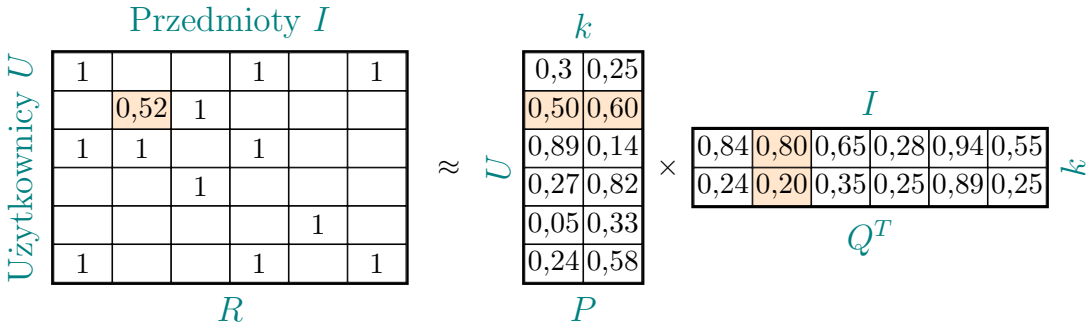
$$R \approx PQ^T. \tag{2.13}$$

W tej dekompozycji P jest macierzą reprezentującą cechy użytkownika o rozmiarze $|U| \times k$, a Q jest to macierz reprezentująca cechy przedmiotów o rozmiarze $|I| \times k$. Następnie, aby

określić preferencję użytkownika u, w stosunku do przedmiotu x, należy:

$$s(x|u) = p_u \cdot q_x, \tag{2.14}$$

gdzie p_u jest to wektor cech dla użytkownika u, a q_x jest to wektor cech dla przedmiotu x. Dzięki takiemu rozwiązaniu użytkownicy i przedmioty są reprezentowane przez niewielką liczbę cech, nazywanych też cechami ukrytymi (z ang. *latent features*) [98]. Technika ta stała się bardzo popularna w systemach rekomendacji za sprawą Simona Funka [135], który wykorzystał ją w konkursie Netfixa, gdzie osiągnęła zadowalające wyniki.



Rysunek 2.4: Przykład faktoryzacji macierzy użytkownik-przedmiot
Opracowanie własne

2.7.2 Spersonalizowany ranking bayesowski

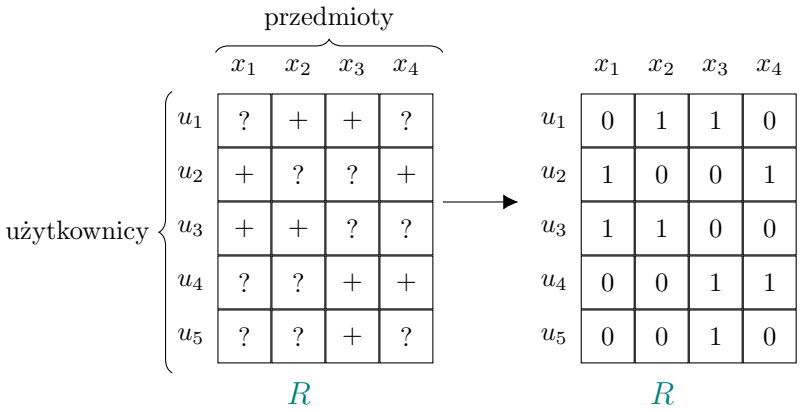
Spersonalizowany ranking bayesowski (z ang. *bayesian personalized ranking*, BPR) jest to metoda, która generuje rekomendacje na bazie macierzy R [139]. Główną ideą tej techniki jest specyficzne traktowanie sytuacji, w której użytkownik nie wszedł w interakcję z danym przedmiotem. Zazwyczaj brak takiej interakcji jest uzupełniany w macierzy R wartością 0 (rys. 2.5), co często jest interpretowane jako negatywna informacja zwrotna (użytkownik nie lubi danego przedmiotu). Może to jednak nie oddawać prawdziwych preferencji aktywnego użytkownika u_A, względem nieocenionego przedmiotu x, ponieważ przedmioty z którymi ten użytkownik nie wszedł w interakcję są mieszaniną przedmiotów, których użytkownik faktycznie nie lubi oraz tych, które mógłby polubić, gdyby miał okazję je ocenić.

Z tego względu nie każdy przedmiot, który został nie oceniony, powinien być traktowany jako przedmiot *negatywny*. Zadaniem systemu bazującego na tej metodzie, będzie wygenerowanie spersonalizowanego rankingu $>_u \subset I^2$, gdzie $>_u$ musi spełniać własności porządku liniowego:

$$\forall x, h \in I : x \neq h \Rightarrow x >_u h \ \vee h >_u x, \tag{2.15}$$

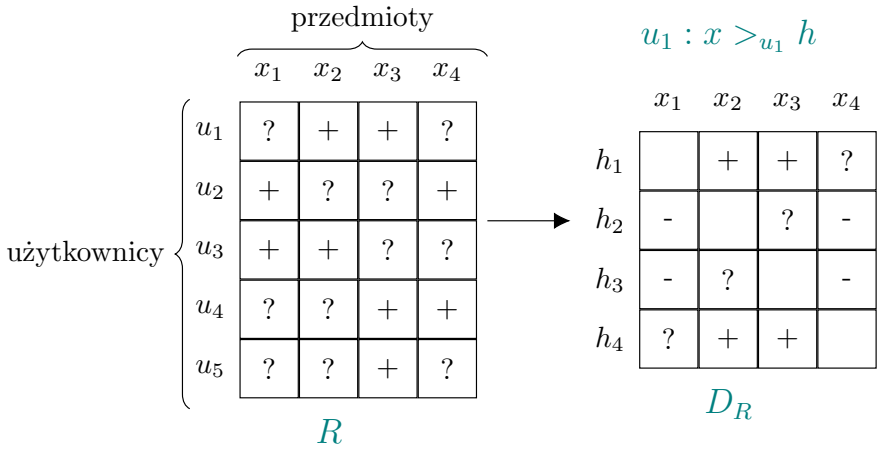
$$\forall x, h \in I : x >_u h \wedge h >_u x \Rightarrow x = h, \tag{2.16}$$

$$\forall x, h, k \in I : x >_u h \wedge h >_u k \Rightarrow x >_u k. \tag{2.17}$$



Rysunek 2.5: W macierzy R symbol „+" oznacza, że użytkownik wszedł w interakcje z danym przedmiotem, natomiast symbol „?" oznacza brak interakcji
Opracowanie własne na podstawie [139]

W metodzie tej porównuje się ze sobą pary przedmiotów i na tej podstawie rekonstruowana jest część $>_u$, gdzie $>_u$ reprezentuje pożądaną spersonalizowaną strukturę preferencji dla użytkownika u. Jeżeli użytkownik u, wszedł w interakcje z przedmiotem x, to zakładamy, że użytkownik preferuje ten przedmiot, ponad wszystkie inne przedmioty, z którymi nie wszedł w interakcję.



Rysunek 2.6: Po lewej stronie widoczna jest macierz ocen R, a po prawej przekształcenie do postaci macierzy D_R, która określa preferencję pomiędzy parami przedmiotów $x >_u h$ dla konkretnego użytkownika u
Opracowanie własne na podstawie [139]

Na rysunku 2.6 w macierzy D_R symbol „+" oznacza, że użytkownik u preferuje przedmiot x nad przedmiot h, natomiast symbol „-" oznacza, że preferuje przedmiot h nad przedmiot x. Na tym rysunku zaprezentowano również przykład, gdzie użytkownik u_1 wszedł w interakcję z przedmiotem x_2, ale nie z przedmiotem x_1. Z tego względu

zakładamy, że użytkownik ten preferuje przedmiot x_2 nad x_1: $x_2 >_u x_1$. Należy jednak pamiętać, że dla dwóch przedmiotów, z którymi użytkownik wszedł w interakcje, nie możemy określić żadnej preferencji. Ta sama sytuacja występuje w momencie, kiedy użytkownik nie wszedł w interakcję z żadnym z przedmiotów z pary. Formalnie zbiór treningowy $D_R : U \times I \times I$, będzie oznaczany jako [139]:

$$D_R := \{(u, x, h) | x \in I_u^+ \wedge h \in I \setminus I_u^+\}, \tag{2.18}$$

gdzie I_u^+ oznacza przedmioty, które użytkownik u polubił. I oznacza wszystkie przedmioty, a $I \setminus I_u^+$ oznacza wszystkie przedmioty z wyłączeniem przedmiotów, które użytkownik polubił. Bayesowskie sformułowanie spersonalizowanego rankingu dla wszystkich przedmiotów $x \in I$ ma na celu maksymalizację prawdopodobieństwa a posteriori [139]:

$$p(\Theta| >_u) \alpha p(>_u |\Theta) p(\Theta), \tag{2.19}$$

gdzie Θ reprezentuje parametry dowolnego modelu rekomendacyjnego (np. faktoryzacji macierzy). Kryterium optymalizacyjne *BPR-OPT* można wyrazić następującym wzorem [139]:

$$\begin{aligned} BPR - OPT :&= ln\ p(\Theta| >_u) \\ &= ln\ p(>_u |\Theta) p)(\Theta) \\ &= ln \prod_{(u,x,h) \in D_R} \sigma(\hat{y}_{ux} - \hat{y}_{uh}) p(\Theta) \\ &= \sum_{(u,x,h) \in D_R} ln\ \sigma(\hat{y}_{ux} - \hat{y}_{uh}) + ln\ p(\Theta) \\ &= \sum_{(u,x,h) \in D_R} ln\ \sigma(\hat{y}_{ux} - \hat{y}_{uh})\ -\ \lambda_\Theta ||\Theta||^2, \end{aligned} \tag{2.20}$$

gdzie $\hat{y}_{ux}$ i $\hat{y}_{uh}$ są to przewidziane oceny dla przedmiotów x oraz h, które wystawił użytkownik u, a λ_Θ to parametry regularyzacji, charakterystyczne dla danego danego modelu.

2.7.3 Ważona regularyzowana faktoryzacja macierzy

Ważona regularyzowana faktoryzacja macierzy (ang. *weighted regularized matrix factorization*, WRMF) została zaproponowana w pracy [83]. W metodzie tej w pierwszej kolejności macierz ocen R jest przekształcana na wartości binarne p_{ux}, które oznaczają preferencję użytkownika u, względem przedmiotu x. Realizowane jest to zgodnie z poniższym wzorem:

$$p_{ux} = \begin{cases} 1 & r_{ux} > 0 \\ 0 & r_{ux} = 0. \end{cases} \tag{2.21}$$

Jeżeli użytkownik u ocenił przedmiot x ($r_{ux} > 0$), oznacza to, że użytkownik u preferuje przedmiot x ($p_{ux} = 1$). Jeżeli jednak użytkownik u, nigdy nie ocenił przedmiotu x, zakładamy, że go nie preferuje ($p_{ux} = 0$). Podobnie jednak jak w przypadku algorytmu *BPR*, brak oceny nie zawsze musi oznaczać, że użytkownik nie lubi danego przedmiotu. Użytkownik mógł po prostu nigdy nie mieć możliwości ocenienia tego przedmiotu, ze względu na np. jego wysoką cenę, ograniczoną dostępność lub małą popularność.

Z tego względu zdefiniowane zostaną zmienne c_{ux}, które będą określać różne stopnie pewności dotyczące zaobserwowania p_{ux} i będą one określane według następującego wzoru:

$$c_{ux} = 1 + \alpha r_{ux}. \tag{2.22}$$

Celem algorytmu WRMF jest znalezienie wektora $w_u \in \mathbb{R}^f$, dla każdego użytkownika u i wektora $z_x \in \mathbb{R}^f$, dla każdego przedmiotu x. Iloczyn skalarny pomiędzy tymi wektorami, będzie oznaczał preferencję użytkownika u, względem przedmiotu x, co można wyrazić następującym wzorem:

$$p_{ux} = w_u^T z_x. \tag{2.23}$$

Te czynniki są obliczane poprzez rozwiązanie następującego problemu optymalizacyjnego [83]:

$$\min_{w_*, z_*} \sum_{u,x} c_{ux}(p_{ux} - w_u^T z_x)^2 + \lambda(\sum_u ||w_u||^2 + \sum_x ||z_x||^2), \tag{2.24}$$

gdzie λ oznacza parametr regularyzacyjny.

Rozdział 3

Ewaluacja systemów rekomendacji

Poprawna ewaluacja systemu rekomendacyjnego jest niezwykle ważna, ponieważ umożliwia twórcom podejmowanie właściwych decyzji projektowych. Do oceny jakości rekomendacji, wykorzystuje się różne metryki jakości, których zadaniem jest przeważnie określenie jak dobrze system potrafi przewidzieć preferencje aktywnego użytkownika, co do przedmiotów, które chciałby on zobaczyć w przyszłości. Taką ewaluację można przeprowadzić w dwóch trybach: *online* i *offline* [190].

W trybie *online* najczęściej wykorzystuje się testy A/B, w celu zbadania bezpośredniego wpływu jaki system rekomendacyjny ma na użytkownika. Jednak ze względu na potrzebę aktywnego uczestnictwa użytkowników w tego typu testach, oraz liczne wyzwania z tym związanem, w literaturze najczęściej ewaluację przeprowadza się w trybie *offline* [133]. Z tego względu ten rozdział poświęcony będzie tylko temu trybowi ewaluacji. Realizowany jest on poprzez ocenę tego jak wygenerowane przez system rekomendacje, pokrywają się z tym, co użytkownik posiada w swoim zbiorze testowym.

Warto przy tym wspomnieć, że dokładność rekomendacji nie jest jedynym kryterium, jakie wykorzystuje się do oceny efektywności działania danego systemu. W literaturze zaproponowano również inne miary, takie jak: oryginalność, różnorodność, uczciwość i zaufanie [153]. Ze względu na to, że w badaniach eksperymentalnych wykorzystywane zostaną wybrane miary dotyczące jakości generowanych rekomendacji, to w kolejnych podrozdziałach zostaną one dokładniej omówione.

3.1 Przewidywanie oceny

Jednym z podstawowych podejść wykorzystywanym do ewaluacji systemów rekomendacyjnych i chyba najbardziej popularnym jest podejście, gdzie system próbuje przewidzieć oceny (z ang. *rating prediction task*), jakie aktywny użytkownik u_A, wystawiłby poszczególnym przedmiotom [159]. W tym przypadku system rekomendacyjny generuje przewidywaną ocenę $\hat{r}_{ux}$ dla par użytkownik-przedmiot (u, x), ze zbioru testowego $Test$ dla których prawdziwa wartość oceny r_{ux} jest znana. Do ocenienia jakości predykcji, wykorzystuje się miarę średniego błędu bezwzględnego (z ang. *mean absolute error*, MAE), wyrażoną

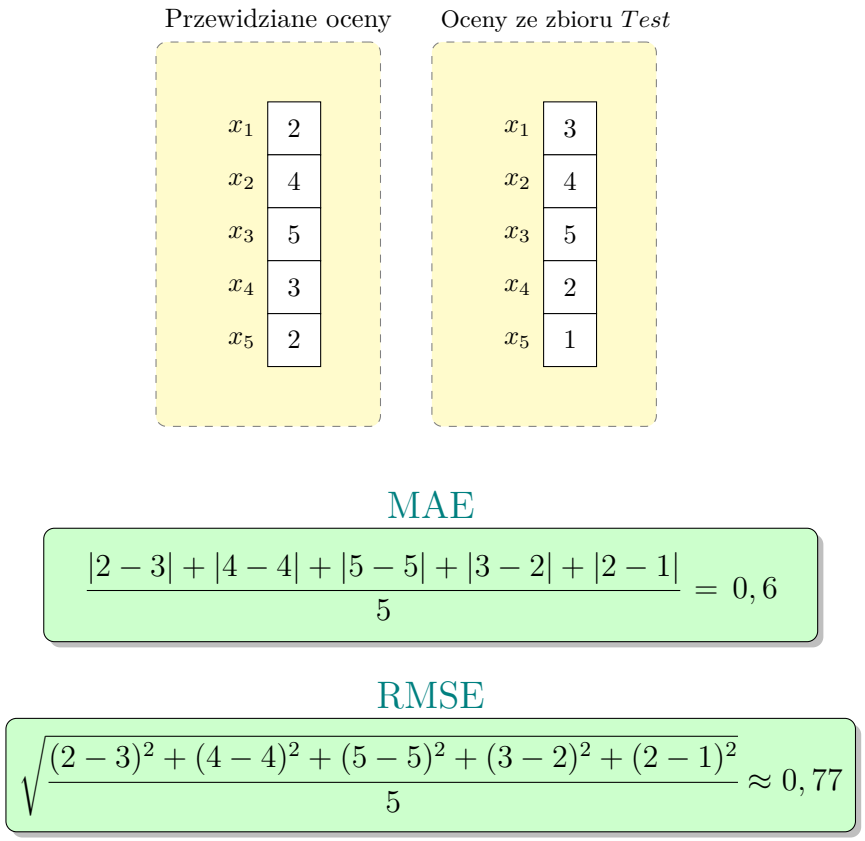
następującym wzorem:

$$MAE = \frac{1}{|Test|} \sum_{(u,x) \in Test} |\hat{r}_{u,x} - r_{u,x}|. \tag{3.1}$$

W literaturze [153] częściej jednak wykorzystywana jest miara pierwiastka z sumy błędu średniokwadratowego (z ang. *rooted mean squared error*, RMSE), ponieważ w większym stopniu kara ona niewłaściwe predykcje. Ten sposób oceniania został spopularyzowany głównie poprzez konkurs *Netflix Prize* [26], gdzie właśnie miara RMSE była wykorzystywana do ewaluacji wyników poszczególnych algorytmów i wyrażona jest ona następującym wzorem:

$$RMSE = \sqrt{\frac{1}{|Test|} \sum_{(u,x) \in Test} (\hat{r}_{u,x} - r_{u,x})^2}. \tag{3.2}$$

Na rysunku 3.1 zaprezentowany został prosty przykład obliczania tych miar.



Rysunek 3.1: Przykład obliczenia miary MAE oraz RMSE na podstawie przewidzianych ocen przez algorytm rekomendacyjny oraz ocen, które znajdują się w zbiorze testowym aktywnego użytkownika u_A
Opracowanie własne

3.2 Przewidywanie rankingu

W tym przypadku system rekomendacyjny nie przewiduje jaką ocenę wystawiłby użytkownik danemu przedmiotowi, a raczej stara się przewidzieć ich kolejność w jakiej zostaną one zaprezentowane użytkownikowi. Zdaniem niektórych badaczy [91], przedstawienie problemu rekomendacji w taki sposób jest bardziej uzasadnione, ponieważ jest zbliżone do praktycznego zastosowania systemów rekomendacyjnych, gdzie przeważnie rekomendacje otrzymujemy w postaci pewnej uporządkowanej listy przedmiotów.

Warto pamiętać o tym, że kolejność elementów znajdujących się na takiej liście ma znaczenie, ponieważ z dużo większym prawdopodobieństwem użytkownik wybierze produkt znajdujący się na początku takiej listy, niż ten, który znajduje się na końcu. Do oceniania rekomendacji tego typu wykorzystuje się miary opisane poniżej.

3.2.1 Precyzja i zwrot

Precyzja (z ang. *precision*) i zwrot (z ang. *recall*) są to miary często wykorzystywane przy ocenie jakości klasyfikacji [121, s. 327]. W kontekście systemów rekomendacyjnych, precyzja reprezentuje procent przedmiotów relewantnych, które pojawiły się w zarekomendowanym rankingu, natomiast zwrot reprezentuje procent relewantnych przedmiotów, które zostały zarekomendowane [7].

Ze względu na to, że w systemach rekomendacyjnych znajdują się tysiące przedmiotów, a tylko niewielką część z nich jest rekomendowana użytkownikowi, często rozpatruje się tylko pierwsze K elementów, co jest oznaczane jako $Precyzja@K$. Przykładowo $Precyzja@10$ oznacza obliczenie precyzji tylko dla 10 pierwszych przedmiotów, które znajdują się w zarekomendowanym rankingu. Precyzję i zwrot w systemach rekomendacji oblicza się według następujących wzorów [126]:

$$Precyzja@K(\tau_i^r) = \frac{|Rel(u_i) \cap \tau_i^r@K|}{|\tau_i^r@K|}, \tag{3.3}$$

$$Zwrot@K(\tau_i^r) = \frac{|Rel(u_i) \cap \tau_i^r@K|}{|Rel(u_i)|}, \tag{3.4}$$

gdzie $Rel(u_i)$ jest zbiorem elementów relewantnych dla użytkownika u_i, a $\tau_i^r@K$ oznacza pierwsze K elementów w rankingu τ_i^r, w którym znajdują się zarekomendowane przedmioty.

3.2.2 Średnia precyzja i uśredniona średnia precyzja

Zasadniczą wadą zwykłej precyzji jest to, że nie bierze ona pod uwagę pozycji elementów relewantnych, w jakiej znajdują się one w rankingu. Z tego względu do oceny jakości rekomendacji częściej stosuje się miarę średniej precyzji (z ang. *average precision*, AP), która

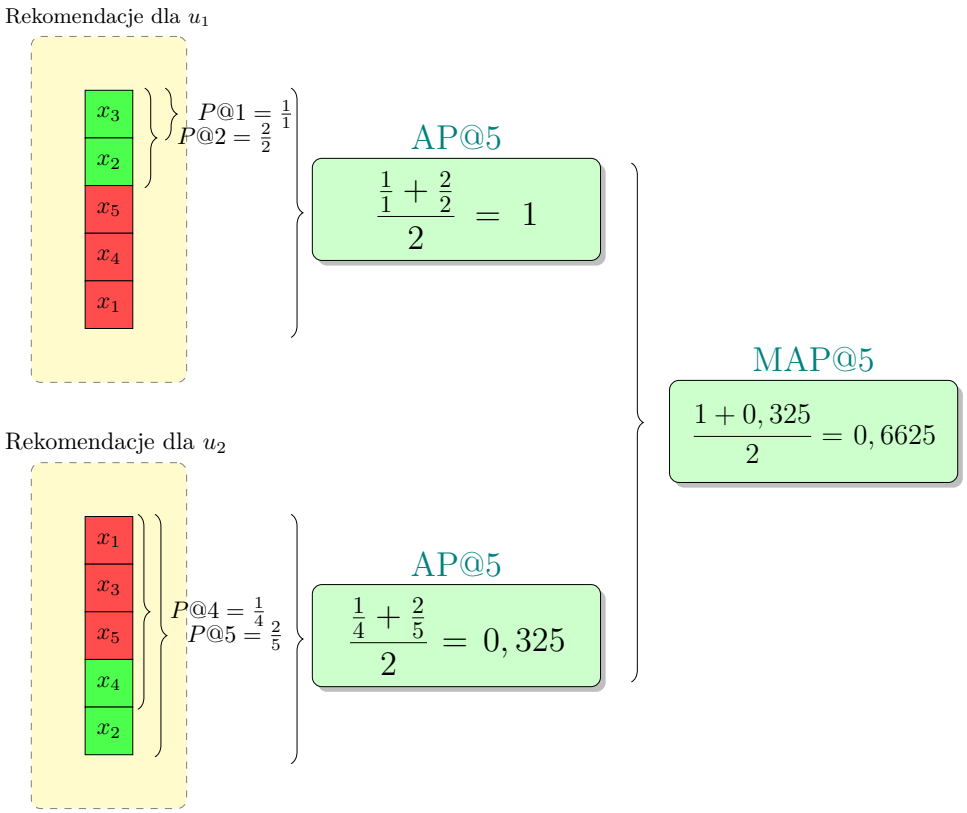
uśrednia wartości precyzji, obliczanej dla poszczególnych pozycji w rekomendowanym rankingu. Średnią precyzję definiujemy zgodnie ze wzorem [126]:

$$AP@K(\tau_i^r) = \sum_{z=1}^{K} \frac{Precyzja@z(\tau_i^r) \times rel_{u_i}(x_z)}{\min(|Rel(u_i)|, |r_i^r|)}, \tag{3.5}$$

gdzie K oznacza rozmiar rekomendowanej listy, a $rel_{u_i}(x_z)$ jest to funkcja charakterystyczna, która zwraca wartość 1 w przypadku, gdy przedmiot x_z dla użytkownika u_i jest relewantny (w przeciwnym wypadku 0). Zaletą tej miary jest to, że penalizuje ona niepoprawne uporządkowanie przedmiotów w rankingu. Aby to lepiej zobrazować, na rysunku 3.2 zaprezentowany został prosty przykład obliczania tej miary.

Średnia precyzja opisana powyżej, wykorzystywana jest do ewaluacji rekomendacji w kontekście jednego użytkownika. Zazwyczaj jednak chcemy uśrednić wyniki działania danego algorytmu dla całego systemu rekomendacyjnego, czyli dla wszystkich użytkowników w zbiorze U. Z tego względu zaproponowano uśrednioną średnią precyzję (z ang. *mean average precision*, MAP), wyrażoną następującym wzorem [126]:

$$MAP@K = \frac{1}{|U|} \sum_{i=1}^{|U|} AP@K(\tau_i). \tag{3.6}$$



Rysunek 3.2: Przykład obliczenia miary AP i MAP dla dwóch użytkowników
Opracowanie własne

3.2.3 Znormalizowany zdyskontowany skumulowany zysk

Miary AP i MAP często są wykorzystywane przy ocenach binarnych. Jednak w sytuacji, kiedy w systemie występują różne poziomy istotności i posiadamy informację na temat tego, jak bardzo dany przedmiot jest relewantny (np. w skali od 1 do 5), to warto wykorzystać miarę znormalizowanego zdyskontowanego skumulowanego zysku (z ang. *normalized discounted cumulative gain*, NDCG) [87]. Podobnie jak w przypadku miary AP, celem tej miary jest premiowanie przedmiotów, które znajdują się wysoko (bliżej pierwszej pozycji) na rekomendowanej liście.

Aby lepiej zrozumieć sposób obliczania tej miary, w pierwszej kolejności należy zwrócić uwagę na sposób obliczania miary skumulowanego zysku (z ang. *cumulative gain*, CG), która wyrażona jest następującym wzorem:

$$CG(\tau_i^r) = \sum_{x_j \in \tau_i^r} rel_{u_i}(x_j), \tag{3.7}$$

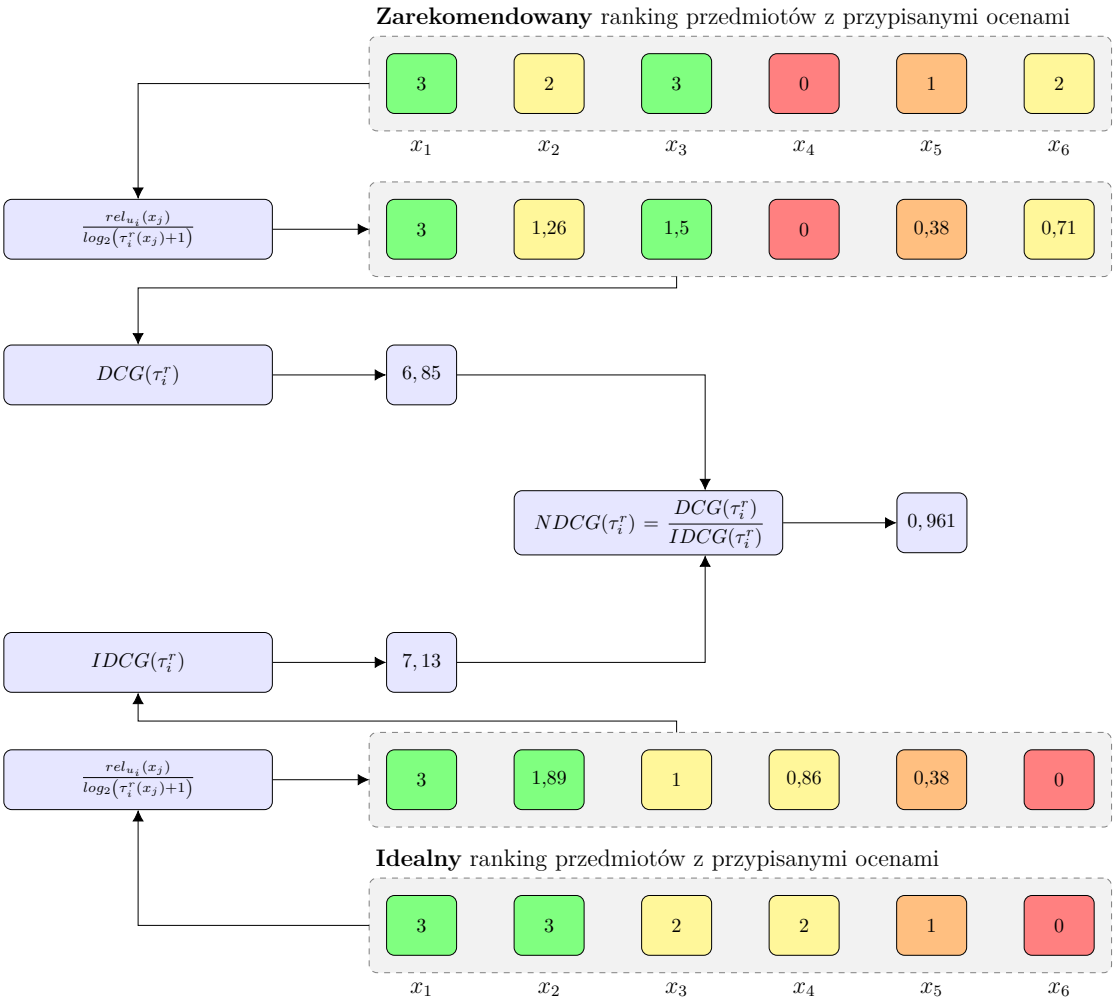
gdzie $rel_{u_i}(x_j)$ oznacza stopień relewantności danego przedmiotu x_j dla użytkownika u_i. Niestety zasadniczą wadą miary CG jest to, że nie uwzględnia ona pozycji relewantnych przedmiotów. Z tego względu zaproponowano zdyskontowany skumulowany zysk (z ang. *discounted cumulative gain*, DCG) wyrażony następującym wzorem [126]:

$$DCG(\tau_i^r) = \sum_{x_j \in \tau_i^r} \frac{rel_{u_i}(x_j)}{log_2\left(\tau_i^r(x_j) + 1\right)}. \tag{3.8}$$

W mianowniku tego wzoru występuje logarytm, który penalizuje stopień relewantności proporcjonalnie do pozycji przedmiotu na rekomendowanej liście. Niestety miary DCG nie można porównywać pomiędzy użytkownikami, ze względu na to, że każdy z użytkowników będzie posiadać różną liczbę relewantnych przedmiotów w swoich rekomendacjach. Z tego względu należy dokonać normalizacji i do tego celu wykorzystuje się idealny (wzorcowy) zdyskontowany skumulowany zysk (z ang. *ideal discounted cumulative gain*, IDCG), który wykorzystywany jest jako współczynnik normalizujący. Następnie, aby uzyskać miarę NDCG, należy [126]:

$$NDCG(\tau_i^r) = \frac{DCG(\tau_i^r)}{IDCG(\tau_i^r)}. \tag{3.9}$$

Prosty przykład reprezentujący obliczanie tej miary, przedstawiony jest na rysunku 3.3.



Rysunek 3.3: Przykład obliczenia miary NDCG. Kolory: zielony, żółty, pomarańczowy i czerwony oznaczają różne stopnie relewantności danego przedmiotu
Opracowanie własne na podstawie [166]

Rozdział 4

Ewolucja różnicowa

W tym rozdziale zaprezentowane zostanie krótkie wprowadzenie do algorytmów metaheurystycznych, które wykorzystywane są do rozwiązywania problemów optymalizacyjnych w przypadkach, w których użycie technik dokładnych jest nieefektywne. Przedstawiony zostanie również krótki przegląd literaturowy, związany z wykorzystaniem algorytmów ewolucyjnych w systemach rekomendacyjnych oraz omówiony zostanie algorytm ewolucji różnicowej (z ang. *differential evolution*, DE).

4.1 Wprowadzenie do metaheurystyk

Istnieje wiele problemów praktycznych, dla których znalezienie dokładnego rozwiązania w akceptowalnym czasie jest często niemożliwe. Naukowcy przez lata prowadzili intensywne prace badawcze w tym obszarze, czego efektem było wypracowanie pewnego podziału tych problemów na klasy.

Klasa P składa się z problemów, które są rozwiązywalne w czasie wielomianowym, czyli są to problemy, które można rozwiązać w czasie $O(n^k)$ dla pewnej stałej k, gdzie n jest rozmiarem danych wejściowych. Do klasy NP należą problemy, które są *weryfikowalne* w czasie wielomianowym. Mówiąc prościej, jeżeli dostalibyśmy rozwiązanie danego problemu, to moglibyśmy zweryfikować jego poprawność w czasie wielomianowym. Bardziej szczegółowy opis tych problemów wraz z formalną definicją i licznymi przykładami można znaleźć w [51].

Problemy optymalizacyjne możemy podzielić na dyskretne oraz ciągłe. Przykładem problemu dyskretnego jest popularny problem komiwojażera, który nieformalnie opisuje problem, w którym wędrowny sprzedawca ma za zadanie odwiedzić wszystkie wybrane miasta i wrócić do miasta początkowego, pokonując jak najkrótszą trasę (choć kryteria optymalizacyjne mogą być różne). Formalną definicję tego problemu można znaleźć w [79]. Problemy ciągłe natomiast w odróżnieniu od dyskretnych, przyjmują wartości ze zbioru liczb rzeczywistych.

Tak naprawdę z problemami optymalizacyjnymi mamy do czynienia każdego dnia, tylko nie zawsze potrafimy je dostrzec i w odpowiedni sposób sformułować. W dużo większym stopniu zdają sobie z nich sprawę inżynierowie, którzy pracując przy złożonych

systemach informatycznych, mają często do czynienia ze skomplikowanymi modelami, które ze względu na dużą liczbę parametrów są trudne do odpowiedniego dostrojenia [114]. Problemy tego typu zapoczątkowały badania nad algorytmami aproksymacyjnymi, które umożliwiały znalezienie przybliżonego rozwiązania w akceptowalnym czasie [50]. W prostych słowach można powiedzieć, że algorytmy te poświęcają jakość rozwiązania na rzecz przyspieszenia obliczeń.

Na początku zaproponowano proste heurystyki, które można było zastosować tylko do konkretnych problemów optymalizacyjnych. Następnie w celu ich uogólnienia zaproponowano metaheurystyki, które były już algorytmami dużo bardziej uniwersalnymi, a ich ogólna idea polegała na iteracyjnym poprawianiu danego rozwiązania do momentu osiągnięcia kryterium stopu. Z czasem zaczęto wykorzystywać je również w różnych dziedzinach np. w inżynierii [151], systemach produkcji [127] i przemyśle [152].

Mimo intensywnych i wieloletnich badań nad metaheurystykami, nie zaproponowano uniwersalnego algorytmu, który można by było zastosować do wszystkich problemów optymalizacyjnych (zgodnie z teorią *No Free-Lunch* [180]). Spowodowało to, że na przestrzeni ostatnich lat powstało wiele nowych technik i ich licznych modyfikacji [2]. Warto jednak zaznaczyć, że niektórzy naukowcy kwestionują zasadność powstawania kolejnych metaheurystyk, zwracając uwagę na to, że tworzenie niezliczonej liczby algorytmów na podstawie procesów zachodzących w naturze, powoli odsuwa nas od naukowego rygoru [164].

Samych początków tych technik można dopatrywać się w algorytmach stochastycznych, które po raz pierwszy zostały opisane już w 1951 roku [143]. W rozwoju metaheurystyk niezwykle ważne było zaproponowanie algorytmów ewolucyjnych (z ang. *evolutionary algorithms*, EA), które były zbiorem technik optymalizacyjnych, a ich ogólna zasada działania opierała się na inspiracjach biologicznych, które zostały zaczerpnięte z procesów ewolucyjnych opisanych przez Charlesa Darwina w 1859 roku [55]. W tych algorytmach występuje pewna populacja osobników, która poprzez odpowiednie kodowanie, reprezentuje dopuszczalne rozwiązania danego problemu. Populacja ta jest cały czas zmieniana poprzez wykorzystanie mechanizmów takich jak mutacja, krzyżowanie oraz selekcja, a do kolejnych iteracji przechodzą osobnicy lepiej przystosowani.

W 1975 roku John Holland zaproponował algorytm genetyczny (z ang. *genetic algorithm*, GA), który przyczynił się do znacznego spopularyzowania metaheurystyk. Na podstawie tego algorytmu zaproponowano również programowanie genetyczne (z ang. *genetic programming*, GP), które jest ciekawym przykładem zastosowania tego typu technik, ponieważ wykorzystywane jest do automatycznego generowania programów komputerowych [157].

Kolejnym zaproponowanym w literaturze algorytmem, który był szczególnie istotny z punktu widzenia podstaw matematycznych, był algorytm symulowanego wyżarzania (z ang. *simulated annealing*, SA) [97]. Został on zaproponowany w 1983 roku i był on rozwinięciem metody Monte Carlo z 1953 roku [116]. Zasada działania tego algorytmu

bazuje na inspiracjach zaczerpniętych z procesów wyżarzania, które zachodzą w metalurgii.

Marco Dorigo zaproponował w 1992 roku algorytm mrowiskowy (z ang. *ant colony optimization*, ACO), który wzorował się na zachowaniu mrówek [62]. Algorytm ten najczęściej wykorzystywany był do rozwiązywania problemów związanych z poszukiwaniem najkrótszych ścieżek w grafach. W algorytmie tym występuje mechanizm komunikacji niebezpośredniej, który jest realizowany poprzez symulację odkładania śladu feromonowego i jego wyparowywania, co umożliwia osobnikom wzajemną komunikację.

W 1995 roku James Kennedy oraz Russel Eberhart opracowli algorytm optymalizacji stadnej cząsteczek (z ang. *particle swarm optimization*, PSO) [94]. Czerpał on inspirację biologiczne ze sztucznego życia i zakładał występowanie stada cząsteczek, które przeszukują przestrzeń rozwiązań, równoważąc wpływ swojej wiedzy na temat tej przestrzeni oraz stada.

W kolejnych latach algorytmy metaheurystyczne opierały się na różnych inspiracjach. Wśród najbardziej popularnych można wyróżnić następujące algorytmy: rojowy (z ang. *artificial bee colony*, ABC) [90], kukułczy (z ang. *cuckoo search*, CS) [187], nietoperzowy (z ang. *bat algorithm*, BA) [185], poszukiwania harmonii (z ang. *harmony search*, HS) [186].

4.2 Algorytmy ewolucyjne w systemach rekomendacyjnych

Analizując literaturę związaną z zastosowaniem algorytmów ewolucyjnych w systemach filtrujących informację można zauważyć, że zagadnienie to cieszy się dużym zainteresowaniem w środowisku naukowym. Powstało wiele prac badawczych powiązanych z tą tematyką, a szczegółowe informację na ten temat można znaleźć w licznych przeglądach literaturowych [49, 137, 122, 1].

W kontekście systemów rekomendacyjnych algorytmy ewolucyjne często wykorzystywane są w celu lepszego spersonalizowania wygenerowanych rekomendacji, ponieważ umożliwiają one stosunkowo łatwą optymalizację parametrów wykorzystywanych w zaprojektowanym modelu. Takie parametry w literaturze są często nazywane *wektorem preferencji* lub *wektorem wag*, ponieważ najczęściej wykorzystywane są do zamodelowania preferencji aktywnego użytkownika. Algorytmy ewolucyjne zmieniają wartości takich wag iteracyjne, optymalizując zadane kryterium, które zazwyczaj oznacza optymalizację pod kątem jakości generowanych rekomendacji [150].

Jednym z najbardziej kompleksowych przeglądów literaturowych w kontekście wykorzystania algorytmów ewolucyjnych w systemach rekomendacji jest przegląd [82], w którym to autorzy uwzględnili ponad 65 publikacji. Opisali oni nie tylko wykorzystywane typy algorytmów ewolucyjnych, ale omówione zostały dokładne metodyki prowadzenia badań, wraz z informacjami na temat wykorzystywanych zbiorów danych. W tym przeglądzie literaturowym, zaproponowano podział algorytmów ewolucyjnych, ze względu na wykorzystywane podejścia:

- Wykorzystujące ważenie cech – w tym podejściu każdy użytkownik lub przedmiot jest reprezentowany przez pewien n-wymiarowy zbiór cech, a zadaniem algorytmu ewolucyjnego jest wyszukanie n-wymiarowego wektora preferencji, gdzie poszczególne elementy tego wektora odpowiadają poszczególnym cechom. Jedną z pierwszych prac w której zaproponowano to podejście była praca [170]. Autorzy wykorzystali w niej GA, który na podstawie wygenerowanych wcześniej profili użytkowników, wyszukiwał wektor wag, który reprezentował wagi w ważonej mierze odległości euklidesowej. Na tej podstawie dobierane było sąsiedztwo w którego skład wchodzili użytkownicy, których zagregowane preferencje w największym stopniu minimalizowały wartość funkcji oceny. Za funkcję oceny autorom posłużyła różnica pomiędzy wygenerowaną oceną przez system rekomendacyjny, a faktyczną oceną znajdującą się w zbiorze treningowym aktywnego użytkownika u_A. Ci sami autorzy w kolejnej pracy [171] rozszerzyli swoje badania i wykorzystali algorytm roju cząstek, który poprawił uzyskane wcześniej rezultaty. Podobne podejście zostało wykorzystane przez innych badaczy w pracach [81, 173] i również przyniosło zadowalające efekty.

- Oparte na grupowaniu – jest to podejście oparte na algorytmach wykorzystywanych przy analizie skupień [178]. Są to algorytmy, których zadaniem jest podzielenie użytkowników na pewne grupy, gdzie w ramach danej grupy użytkownicy są do siebie podobni [102]. W tym podejściu algorytmy ewolucyjne wykorzystywane są często w celu początkowej inicjalizacji centrów grup, ponieważ w algorytmach grupujących takie centra są często inicjalizowane w sposób losowy. Przykładowo w publikacji [96], algorytm genetyczny został wykorzystany do początkowej inicjalizacji algorytmu *k–means* i bazując na wynikach badań, autorzy stwierdzili, że takie podejście poprawia jakość generowanych rekomendacji.

- Oparte na modelach latentnych – w tym podejściu algorytmy ewolucyjne wykorzystywane są do wyszukania optymalnych wag dla ukrytych cech (z ang. *latent features*), które reprezentują użytkowników i przedmioty. Przykładowo w publikacji [145] autorzy wykorzystali do tego celu algorytm genetyczny razem z algorytmem *k-means*. W kolejnej pracy [146] autorzy rozszerzyli swoje podejście, wprowadzając dodatkowo macierz preferencji.

- Oparte na uczeniu się funkcji – w tym podejściu wykorzystuje się algorytmy ewolucyjne w celu wygenerowania funkcji rangującej. Przykładem zastosowania takiego podejścia jest praca [75], w której to autorzy zaproponowali wykorzystanie do tego celu programowania genetycznego i optymalizacji wielokryterialnej.

- Inne – algorytmy ewolucyjne wykorzystywane są często do optymalizacji wielokryterialnej [175, 192]. W kontekście systemów rekomendacyjnych najczęściej odnosi się to

do problemu jednoczesnej optymalizacji miar, które wykorzystywane są do ewaluacji wygenerowanych rekomendacji np. jakości, różnorodności i nowości.

4.3 Algorytm ewolucji różnicowej

Ewolucja różnicowa jest to metaheurystyka, która została opracowana przez K. Price'a i R. Storna w 1997 roku [160]. Znalazła ona zastosowanie w wielu zagadnieniach i problemach optymalizacyjnych, choć zazwyczaj jest wykorzystywana do optymalizacji problemów ciągłych. Przykładem takiego problemu może być *funkcja Michalewicza*, która jako jedna z funkcji testowych jest często wykorzystywana do sprawdzania efektywności działania tego typu algorytmów [120].

Algorytm ten ze względu na prostotę implementacji, szybką zbieżność oraz małą liczbę parametrów kontrolnych jest często wykorzystywany do optymalizacji funkcji wielowymiarowych i nieregularnych, które to funkcje nie mogą być optymalizowane przy użyciu klasycznych algorytmów optymalizacyjnych [69]. Takie cechy czynią ten algorytm idealnym kandydatem do bezpośredniej optymalizacji miar, które wykorzystywane są do ewaluacji systemów rekomendacyjnych.

Należy jednak zaznaczyć, że pomimo dużej popularności i skuteczności tego algorytmu w zastosowaniach praktycznych, istnieje stosunkowo mała liczba publikacji dotyczących jego analizy teoretycznej, a przegląd istniejących prac można znaleźć w publikacji [129]. Dodatkowo w niniejszej rozprawie zaprezentowany został tylko podstawowy wariant tego algorytmu, gdzie w literaturze algorytm ten doczekał się licznych wariantów i modyfikacji [38, 40, 117].

Występuje tutaj pewna populacja, która inicjalizowana jest poprzez losowe rozmieszczenie osobników w przestrzeni rozwiązań. Formalnie populację P będziemy zapisywać jako:

$$P = \{w_1, w_2, \ldots, w_{NP}\}, \tag{4.1}$$

gdzie:

NP – liczba osobników w populacji.

Każdy osobnik w populacji P jest zazwyczaj reprezentowany przez pewien wektor liczb rzeczywistych, który reprezentuje rozwiązanie danego problemu optymalizacyjnego:

$$w_g = [w_{g,1}, w_{g,2}, \ldots, w_{g,b}], \tag{4.2}$$

gdzie:

b – jest to rozmiar wektora.

W algorytmie tym wyróżniane się dwa operatory: mutacja i krzyżowanie. Ich celem jest ciągła zmiana osobników podczas procesu ewolucji, aż do osiągnięcia kryterium stopu. Należy zwrócić uwagę na to, że w odróżnieniu od np. algorytmu genetycznego, mutacja jest tutaj operatorem nadrzędnym i jest wykonywana przed krzyżowaniem. Dodatkowo krok mutacji nie zależy od rozkładu prawdopodobieństwa. W podstawowej wersji algorytmu wariant mutacji różnicowej, który w literaturze nazywany jest wariantem *DE/rand/1*, można wyrazić następującym wzorem:

$$v_i = w_{r_1} + F\left(w_{r_2} - w_{r_3}\right), \tag{4.3}$$

gdzie:

v_i – nowy wektor,

r_1, r_2, r_3 – trzy losowe numery osobników z populacji P, przy czym $r_1 \neq r_2 \neq r_3$,

F – współczynnik wzmocnienia przyjmujący wartość z przedziału $\langle 0, 1\rangle$.

Kolejnym etapem algorytmu jest zastosowanie operatora krzyżowania, który tworzy nowego osobnika z_i, poprzez połączenie genotypów rodzica w_i z populacji rodziców P, oraz osobnika v_i powstałego w wyniku zastosowania operatora mutacji. Podstawowy wariant procesu krzyżowania dwumianowego można wyrazić za pomocą następującego wzoru:

$$z_{i,j} = \begin{cases} v_{i,j} & \text{gdy } (rand(j) \leq CR \text{ lub } i = i_{rand}) \\ w_{i,j} & \text{w przeciwnym wypadku.} \end{cases} \tag{4.4}$$

gdzie CR jest to prawdopodobieństwo krzyżowania, a i_{rand} to losowa liczba ze zbioru $\{1, 2, \ldots, NP\}$. Algorytm 1 przedstawia pseudokod algorytmu DE.

Warto również wspomnieć o tym, że w literaturze zaproponowano wiele różnych strategii mutacji [128].

Algorytm 1 Pseudokod algorytmu ewolucji różnicowej (z ang. *differential evolution*, DE)

Wejście: **NP** – liczba osobników w populacji, **F** – współczynnik wzmocnienia, **CR** – prawdopodobieństwo krzyżowania, **b** – rozmiar wektora

Wyjście: w_{best} – najlepszy osobnik z populacji P

Zainicjuj populację P o rozmiarze **NP**
Zainicjuj populację próbną Z o rozmiarze **NP**

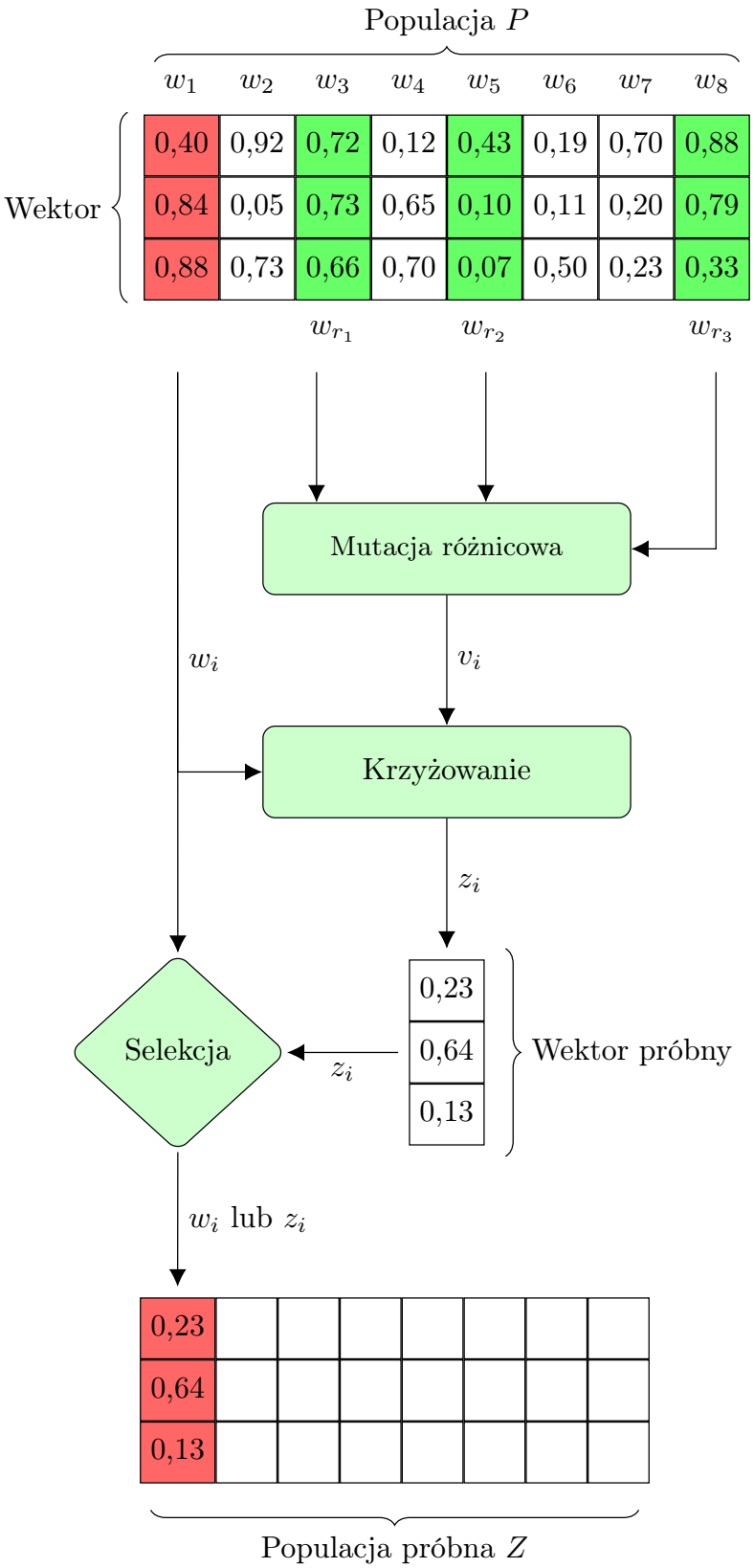
repeat
  for $i := 1$ to **NP do**
    wygeneruj trzy losowe liczby r_1, r_2, $r_3 \in \{1, 2, \ldots, \mathbf{NP}\}$, gdzie $r_1 \neq r_2 \neq r_3 \neq i$
    $v_i = w_{r_1} + \mathbf{F}\,(w_{r_2} - w_{r_3})$ ▷ nowy osobnik v_i

    for $j := 1$ to **b do**
      wygeneruj losową liczbę $j_{rand} \in \{1, 2, \ldots, \mathbf{NP}\}$
      $$z_{i,j} = \begin{cases} v_{i,j} & \text{gdy } (rand(0,1) \leq \mathbf{CR} \text{ lub } j = j_{rand}) \\ w_{i,j} & \text{w przeciwnym wypadku.} \end{cases}$$
    end for

    if fitness(z_i) $\leq$ fitness(w_i) **then** ▷ warunek w zależności od kryterium
      umieść osobnika z_i w populacji próbnej Z
    else
      umieść osobnika w_i w populacji próbnej Z
    end if
  end for

  $P := Z$
until warunek stopu

oceń_wszystkie_osobniki_w_populacji(P)
w_{best} := wybierz_najlepszego_osobnika(P)
return w_{best}



Rysunek 4.1: Schemat algorytmu ewolucji różnicowej
Opracowanie własne

Rozdział 5

Agregacja rang

W tym rozdziale omówiony zostanie problem agregacji rang (z ang. *rank aggregation problem*), w literaturze nazywanym również agregacją preferencji (z ang. *aggregation of preferences*) [42]. W uproszczeniu problem ten odnosi się do sytuacji, w której mając do dyspozycji kilka rankingów, będących uporządkowanymi listami składającymi się z pewnych obiektów (np. przedmiotów), naszym zadaniem jest utworzenie nowego rankingu, który będzie *lepszy* od rankingów bazowych. Podejście to zyskuje obecnie na popularności, ponieważ nawiązuje ono do idei zaczerpniętej z *rodziny klasyfikatorów*.

5.1 Informacje ogólne

Problem agregacji rang jest znanym problemem, szczególnie w kontekście teorii wyboru społecznego, która zajmuje się analizą zbiorowego podejmowania decyzji oraz tym, jak przekształcić preferencje indywidualnych użytkowników w preferencje grupy [112]. Przykładem takiego problemu może być zwyczajne głosowanie, gdzie każda osoba określa swoje preferencje w stosunku do zaprezentowanych kandydatów, a następnie analizując preferencję poszczególnych osób, określa się preferencję grupy [193]. Chociaż za datę powstania współczesnej teorii wyboru społecznego podaje się rok 1951 i monografię [15], napisaną przez Keenetha J. Arrowa, laureata Nagrody Nobla w dziedzinie ekonomii, to początków tej teorii można doszukiwać się już w XVIII wieku, gdzie odkryte przez markiza De Condorceta i Jean-Charles'a de Borda paradoksy, ukazywały niedoskonałości klasycznych metod podejmowania zbiorowych decyzji (np. zwykłej większości) [111].

W kontekście współczesnych systemów filtrujących informację, problem agregacji rang został opisany przez C. Dwork w 2001 roku [63]. Autorka w swojej publikacji przedstawiła podstawy teoretyczne tego problemu, analizując go przez pryzmat systemów wyszukiwania informacji. Zaproponowała ona szereg metod, dzięki którym można taką agregację utworzyć i uzasadniła, że wykorzystanie tego typu algorytmów jest istotne, choćby z uwagi na fakt, że istnieje wiele wyszukiwarek i pomimo upływu lat żadna z nich nie jest uznawana za *najlepszą*. Ma to związek z tym, że nie istnieje jeden, uniwersalny algorytm, który można by było wykorzystać w każdej sytuacji, a potrzeby i preferencje poszczególnych użytkowników w sieci Internet mogą być różne.

Dodatkowo dochodzi problem, który autorka nazywa *spamem*, czyli nierelewantnymi dokumentami pojawiającymi się w wynikach wyszukiwania. Różne systemy wyszukiwania informacji mogą zwracać wyniki, które zawierają różną ilość spamu dla poszczególnych użytkowników. Mniej oczywistym problemem jest również to, że na wyniki działania tego typu systemów mogą wpływać przeróżne firmy, które w ten sposób realizują swoje interesy. Przykładowo wyszukiwarka może wysoko pozycjonować przedmioty, które zostały wcześniej przez taką firmę opłacone. Wydaje się, że w obecnych czasach nadal otwartą kwestią jest to, czy użytkownicy są należycie informowani o płatnych wynikach wyszukiwania, które są im prezentowane w wynikach wyszukiwarek.

W kontekście systemów rekomendacji powstało kilka prac związanych z agregacją rang (patrz tabela 1.2), chociaż niektórzy badacze zwracają uwagę, że nadal jest to tematyka niewystarczająco przebadana [5, s. 417]. W systemach rekomendacji agregację rang wykorzystuje się przykładowo podczas generowania rekomendacji grupowych [16], a algorytmy agregujące wykorzystywane są w celu uogólnienie preferencji pochodzących od różnych użytkowników. Dodatkowo niektóre typy hybrydowych systemów rekomendacyjnych wykorzystują ideę agregacji rang, chociaż autorzy nie definiują tego wprost (np. [141]). Zazwyczaj w systemach rekomendacyjnych, metody agregujące wykorzystuje się w celu: wygenerowania rekomendacji lepszej jakości, usunięcia wad poszczególnych algorytmów rekomendujących, zwiększenia różnorodności generowanych rekomendacji oraz zredukowania wpływu źle spozycjonowanych przedmiotów [126].

Na szczególną uwagę zasługuje publikacja [126], w której to autorzy spróbowali odpowiedzieć na pytanie, czy wykorzystanie metod agregacji rang w systemach rekomendacji może być efektywne. W tym celu przeprowadzili oni systematyczne badania, w których uwzględnili aż 15 algorytmów rekomendacyjnych oraz 19 metod agregujących, a eksperymenty zostały zrealizowane na siedmiu różnych zbiorach danych. Analizując wyniki badań autorzy stwierdzili, że techniki agregujące poprawiły jakość rekomendacji na sześciu testowanych zbiorach danych. Szczególnie interesujące wydaje się być wykorzystanie technik nadzorowanych, ponieważ w ich przypadku rekomendacje zostały poprawione nawet w sytuacji, kiedy w systemie występowały rankingi niskiej jakości.

5.2 Formalna definicja agregacji rang i metody agregujące

Załóżmy, że dysponujemy pewnym zbiorem elementów (obiektów, przedmiotów itp.) $I = \{x_1, x_2, \ldots, x_m\}$. Ranking definiujemy jako uporządkowaną listę tych elementów $\tau = [x_j \geq x_h \geq \cdots \geq x_z]$, gdzie $\geq$ oznacza relację porządku pomiędzy elementami ze zbioru I, a istotność danego elementu jest określona przez jego pozycję. Symbolem $\tau(x_j)$ oznaczana będzie pozycja (lub ranga) przedmiotu x_j w rankingu τ. Dwa przedmioty x_j i x_h można ze sobą porównać, wykorzystując ich pozycje w rankingu τ. Przykładowo możemy powiedzieć, że przedmiot x_j jest na *lepszej* pozycji w rankingu od przedmiotu x_h, co

będzie oznaczane jako $\tau(x_j) < \tau(x_h)$. Dodatkowo pojedynczy algorytm będzie oznaczany jako S^r, a zbiór wszystkich algorytmów będzie oznaczany jako $S = \{S^1,\ S^2,\ ...,\ S^n\}$. Każdy z algorytmów w zbiorze S, generuje ranking τ^r, a zbiór wszystkich rankingów oznaczany będzie jako $\mathcal{T} = \{\tau^1, \tau^2, \ldots, \tau^n\}$, gdzie n oznacza liczbę algorytmów i liczbę wygenerowanych rankingów. Należy zauważyć, że występuje zależność pomiędzy liczbą algorytmów należących do zbioru S, a liczbą rankingów, które znajdują się w zbiorze $\mathcal{T}$.

Celem agregacji rang jest stworzenie nowego rankingu τ^*, który będzie *lepszy* od poszczególnych rankingów w zbiorze $\mathcal{T}$. Oczywiście sformułowanie *lepszy ranking* można interpretować na różne sposoby, a jego znaczenie należy rozpatrywać w kontekście danego problemu, mając na uwadze jego specyfikę. Przykładowo w systemach rekomendacji może to oznaczać taki ranking, który w największym stopniu podnosi jakość rekomendacji, gdzie jakość ta obliczana jest na podstawie miar opisanych w rozdziale 3. W literaturze częściej wykorzystuje się jednak specjalne miary odległości, które umożliwiają określenie stopnia podobieństwa pomiędzy rankingami. Jedną z takich miar jest odległość Tau Kendalla, która zostanie omówiona w podrozdziale 5.3.

Formalnie problem agregacji rang sprowadza się do zdefiniowania funkcji agregującej Ψ, która na podstawie rankingów w zbiorze $\mathcal{T}$, generuje nowy ranking τ^*:

$$\Psi : \{\tau^1, \tau^2, ..., \tau^n\} \rightarrow \tau^*. \tag{5.1}$$

W zależności od dostępnych danych, funkcja agregująca Ψ może zostać utworzona opierając się na różnych metodach. W literaturze podstawowym podziałem tych metod jest podział ze względu na:

- Metody bazujące na wartościach (z ang. *score-based methods*) – w przypadku tych metod, każdy przedmiot w rankingu ma przyporządkowaną pewną wartość, która określa jego pozycję w rankingu. Następnie metody agregujące tworzą nowy ranking τ^*, łącząc ze sobą wartości z rankingów bazowych.

- Metody bazujące na permutacjach (z ang. *permutation-based methods*) – w przypadku tych metod, agregacja jest tworzona poprzez przeszukanie całej przestrzeni możliwych permutacji elementów ze zbioru I. Należy jednak pamiętać, że wykorzystanie tych metod jest często znacznie bardziej kosztowne obliczeniowo, ponieważ sprawdzana jest cała przestrzeń rozwiązań (czyli $|I|!$).

Innym podziałem metod agregujących jest podział ze względu na rodzaj wykorzystywanego algorytmu uczącego:

- Bazujące na uczeniu nadzorowanym – metody te tworzą model rangujący, wykorzystując do tego celu zbiór treningowy. Algorytmy nadzorowane bazujące na wartościach, zazwyczaj wykorzystują podejście nazywane nauką rangowania, które zostało

dokładniej opisane w rozdziale 6. Metody te są często dość złożone i trudne w implementacji. Ponadto mogą posiadać wiele parametrów, które wymagają odpowiedniego dostrojenia, choć zazwyczaj osiągają lepsze wyniki od metod nienadzorowanych [113].

- Bazujące na uczeniu nienadzorowanym – w przypadku tych metod nie dysponujemy zbiorem treningowym, a agregacja najczęściej tworzona jest poprzez wykorzystanie pewnych miar odległości, które umożliwiają porównywanie poszczególnych rankingów ze sobą (np. odległość Tau Kendalla). Metody te są jednak dużo prostsze w zrozumieniu oraz implementacji. Możemy je dodatkowo podzielić na:
  - Metody pozycyjne (z ang. *positional methods*) – metody pozycyjne podczas obliczania funkcji agregującej, uwzględniają pozycję poszczególnych obiektów w rankingu. Na wejściu przyjmują one zbiór rankingów $\mathcal{T}$ i wykorzystują różne wersje funkcji agregujących (np. metodę *Bordy*, *Comb**), które łączą je ze sobą. Ich niewątpliwą zaletą jest duża popularność, szybkość obliczeń oraz prostota implementacji.
  - Metody większościowe (z ang. *majoritarian methods*) – metody te najczęściej bazują na porównaniach pomiędzy parami obiektów. Przykładem takiej metody jest metoda *OutRank* [172] lub metody bazujące na łańcuchach markowa [63].

Jedną z najpopularniejszych metod agregujących jest metoda Bordy (z ang. *Borda fuse*) [35], która każdemu elementowi $x_j \in I$ w rankingu τ^r, przyporządkowuje wartość zgodnie ze wzorem:

$$x_{j\_borda} = \sum_{\tau^r | \forall \tau^r \in \mathcal{T}, x_j \in \tau^r} |\tau^r| - \tau^r(x_j) + 1, \tag{5.2}$$

po czym elementy te są sortowane w kolejności malejącej. Innymi zaproponowanymi w literaturze metodami agregującymi są metody zaliczane do rodziny metod *Comb** [70]. Metody te wykorzystują funkcję:

$$sc(x_j, \tau^r) = 1 - \frac{\tau^r(x_j) - 1}{|\tau^r|}, \tag{5.3}$$

która przyporządkowuje wartość każdemu przedmiotowi $x_j \in I$ w rankingu τ^r. Następnie rankingi są agregowane przy pomocy metod przedstawionych w tabeli 5.1. Przegląd innych metod agregujących można znaleźć w literaturze [107, 126].

Tabela 5.1: Wybrane algorytmy agregujące zaproponowane w [70]

Nazwa metody	Wzór
CombMin	$\min_{\tau^r \in \mathcal{T}} sc(x_j, \tau^r)$
CombMax	$\max_{\tau^r \in \mathcal{T}} sc(x_j, \tau^r)$
CombSum	$\sum_{\tau^r \in \mathcal{T}} sc(x_j, \tau^r)$
CombMed	$\frac{1}{\lvert\mathcal{T}\rvert} CombSum(x_j, \tau^r)$

5.3 Obliczanie podobieństwa pomiędzy rankingami

We wcześniejszym podrozdziale wspomniano, że definiując *lepszy* ranking należy mieć na uwadze specyfikę danego problemu. W literaturze najczęściej do określenia tego, który ranking jest *lepszy*, wykorzystuje się pewne miary odległości. Celem agregacji rang jest wyszukanie takiego rankingu τ^*, który jednocześnie minimalizuje odległość pomiędzy wszystkimi rankingami ze zbioru $\mathcal{T}$. W pierwszej kolejności należy jednak zaznaczyć, że możemy mieć do czynienia z trzema różnymi rodzajami rankingów [63]:

- Rankingi pełne (z ang. *full lists*) – są to rankingi, które zawierają pełne uporządkowanie wszystkich elementów ze zbioru I.
- Rankingi częściowe (z ang. *partial lists*) – są to rankingi, które zawierają tylko częściowe uporządkowanie elementów ze zbioru I, czyli każdy z rankingów może zostać utworzony na podstawie pewnego podzbioru elementów ze zbioru I. Obliczając odległość pomiędzy takimi rankingami, należy uwzględnić przypadek, w którym choć elementy występują w jednym z rankingów np. τ^1, to mogą nie występować w innym rankingu np. τ^2.
- Rankingi zawierające d pierwszych elementów (z ang. *top-d lists*) – są to rankingi, w których pod uwagę bierze się tylko d pierwszych elementów. Zakłada się jednak, że inne elementy mogą istnieć w takim rankingu, ale znajdują się poniżej pierwszych d elementów.

Jedną z najpopularniejszych miar wykorzystywaną do obliczenia odległości pomiędzy dwoma rankingami jest odległość Tau Kendalla. Jest to miara, która powstała na bazie korelacji Tau Kendalla [93] i dla pełnych rankingów można ją zdefiniować w następujący sposób:

$$K_d(\tau^1, \tau^2) = \sum_{\{x_h, x_j\} \in P(\tau^1, \tau^2)} \bar{K}_{x_h, x_j}(\tau^1, \tau^2). \tag{5.4}$$

gdzie:

$$\bar{K}_{x_h,x_j}(\tau^1,\tau^2) = \begin{cases} 0 \text{ gdy } \Big(\tau^1(x_h) > \tau^1(x_j) \wedge \tau^2(x_h) > \tau^2(x_j)\Big) \vee \Big(\tau^1(x_h) < \tau^1(x_j) \wedge \tau^2(x_h) < \tau^2(x_j)\Big) \\ 1 \text{ gdy } \Big(\tau^1(x_h) < \tau^1(x_j) \wedge \tau^2(x_h) > \tau^2(x_j)\Big) \vee \Big(\tau^1(x_h) > \tau^1(x_j) \wedge \tau^2(x_h) < \tau^2(x_j)\Big) \end{cases} \tag{5.5}$$

P jest zbiorem nieuporządkowanych par elementów, a τ^1 oraz τ^2 są to pełne rankingi, utworzone na bazie zbioru I. Jeżeli para elementów $x_h, x_j \in P(\tau^1,\tau^2)$ jest uporządkowana w tej samej kolejności w rankingach τ^1 oraz τ^2, to nie jest nakładana żadna kara, czyli $\bar{K}_{x_h,x_j}(\tau^1,\tau^2) = 0$. W przeciwnym wypadku $\bar{K}_{x_h,x_j}(\tau^1,\tau^2) = 1$.

Intuicyjnie, odległość Tau Kendalla zlicza liczbę niezgodnych par, które występują pomiędzy dwoma rankingami. Dzieląc liczbę tych par przez maksymalną wartość jaką przyjmuje ta odległość, otrzymamy znormalizowaną wersję tej miary. Odległość Tau Kendalla nazywana jest również odległością sortowania bąbelkowego (z ang. *bubble-sort distance*), ponieważ wyznacza ona liczbę potrzebnych zamian, którą musiałby wykonać algorytm sortujący, aby elementy w obydwu rankingach były w tej samej kolejności. W literaturze można również spotkać inne wersje tej odległości, które są dostosowane do rankingów częściowych [66].

Należy pamiętać, że miara korelacji Tau Kendalla K_c, wykorzystywana w statystyce do obliczania monotonicznej zależności dwóch zmiennych losowych, nie jest tożsama z miarą odległości Tau Kendalla K_d. Aby to lepiej zrozumieć, można wskazać dwie podstawowe różnice. Po pierwsze, korelacja K_c zwraca wartości z przedziału $[-1, 1]$, gdzie miara odległości K_d (znormalizowana) wartości $[0, 1]$. Po drugie, dla identycznych rankingów miara korelacji K_c wskaże wartość 1, natomiast miara odległości K_d, wartość 0.

Wykorzystując miarę odległości K_d, możemy również przedstawić problem agregacji rang jako problem optymalizacyjny. Nasza funkcja celu będzie poszukiwać takiej permutacji elementów w rankingu τ^*, która minimalizuje odległość $K_d(\tau^*, \tau^1, \ldots, \tau^n)$. Agregacja uzyskana poprzez optymalizację odległości Tau Kendalla nazywana jest *optymalną agregacją Kemeny'ego* (z ang. *Kemeny optimal aggregation*). Dowiedziono, że jest to problem NP-trudny już dla 4 rankingów, czyli gdy $n = 4$ [63]. Optymalna agregacja Kemeny'ego jest jednak bardzo istotna z puntu widzenia teorii wyboru społecznego, ponieważ wykazano [188], że spełnia ona istotne własności, które są pożądane przez funkcję agregującą, a są nimi: symetryczność, spójność oraz kryterium Condorceta [74].

Rozdział 6

Uczenie się rangowania

W tym rozdziale opisana zostanie technika, która wykorzystywana jest do tworzenia modeli rangujących. W polskiej literaturze podejście to nazywane jest *nauką rangowania* lub *uczeniem się rangowania* (z ang. *learning to rank*) [67, s. 392] i po raz pierwszy technika ta została opisana w kontekście systemów wyszukiwania informacji [73]. W tym rozdziale przedstawione zostaną podstawy teoretyczne tego podejścia, wraz z omówieniem najważniejszych pojęć.

6.1 Informacje ogólne

Podjęcie decyzji o tym, jakiego algorytmu uczenia maszynowego użyć do danego problemu, zależy od wielu czynników i nie ma tutaj jednego, uniwersalnego algorytmu, które byłby efektywny we wszystkich przypadkach. Z tego względu w literaturze mamy do dyspozycji wiele różnych technik, które wybieramy uwzględniając specyfikę danego problemu.

Jednym z typów algorytmów uczenia maszynowego są algorytmy, które bazują na uczeniu nadzorowanym. Najczęściej wykorzystywane są one dla zadań regresji lub klasyfikacji, a ich celem jest predykcja wartości atrybutu decyzyjnego na podstawie wartości pozostałych atrybutów. W problemie regresji atrybut decyzyjny ma wartość rzeczywistą (np. cena, wiek, pensja), natomiast w problemie klasyfikacji wartość dyskretną (np. kobieta lub mężczyzna, spam lub nie spam, prawda lub fałsz).

W literaturze zaproponowano specjalny typ algorytmów, który wykorzystywany jest do przewidzenia optymalnego uporządkowania elementów w rankingu. Podejście to nazywane jest *nauką rangowania* [67, s. 392]. Przedstawienie problemu optymalizacyjnego w ten sposób, szczególnie w kontekście systemów wyszukiwania informacji i rekomendacji jest uzasadnione, ponieważ wyniki działania tych systemów są często prezentowane w formie pewnego rankingu. Ma to związek z tym, że na portalach internetowych mamy ograniczoną ilość miejsca i użytkownik z większym prawdopodobieństwem *skonsumuje* treści, które są mu zaprezentowane w pierwszej kolejności. Z tego względu najlepiej, żeby prezentowany ranking zawierał jak najwięcej relewantnych treści na początkowych pozycjach. Dobrym

wprowadzeniem do tematyki uczenia się rangowania jest publikacja [105], w której to autor na przykładzie systemów wyszukiwania informacji, objaśnia podstawowe zagadnienia związane z tym podejściem.

Nadzorowany proces nauki rangowania można w uproszczeniu podzielić na dwie fazy: treningową i testową. W fazie treningowej dostarczamy algorytmowi uczącemu określony zbiór zapytań, gdzie każdemu zapytaniu odpowiada pewien zbiór dokumentów. Dodatkowo każdy dokument ma przypisaną pewną etykietę (np. określoną przez człowieka), która określa jego relewantność w stosunku do danego zapytania. Celem fazy treningowej jest skonstruowanie modelu rangującego, który generalizuje wiedzę na podstawie dostępnych w zbiorze treningowym zapytań. Do oceny jakości utworzonego rankingu, wykorzystywane są specjalne miary, które zostały opisane w podrozdziale 3.2.

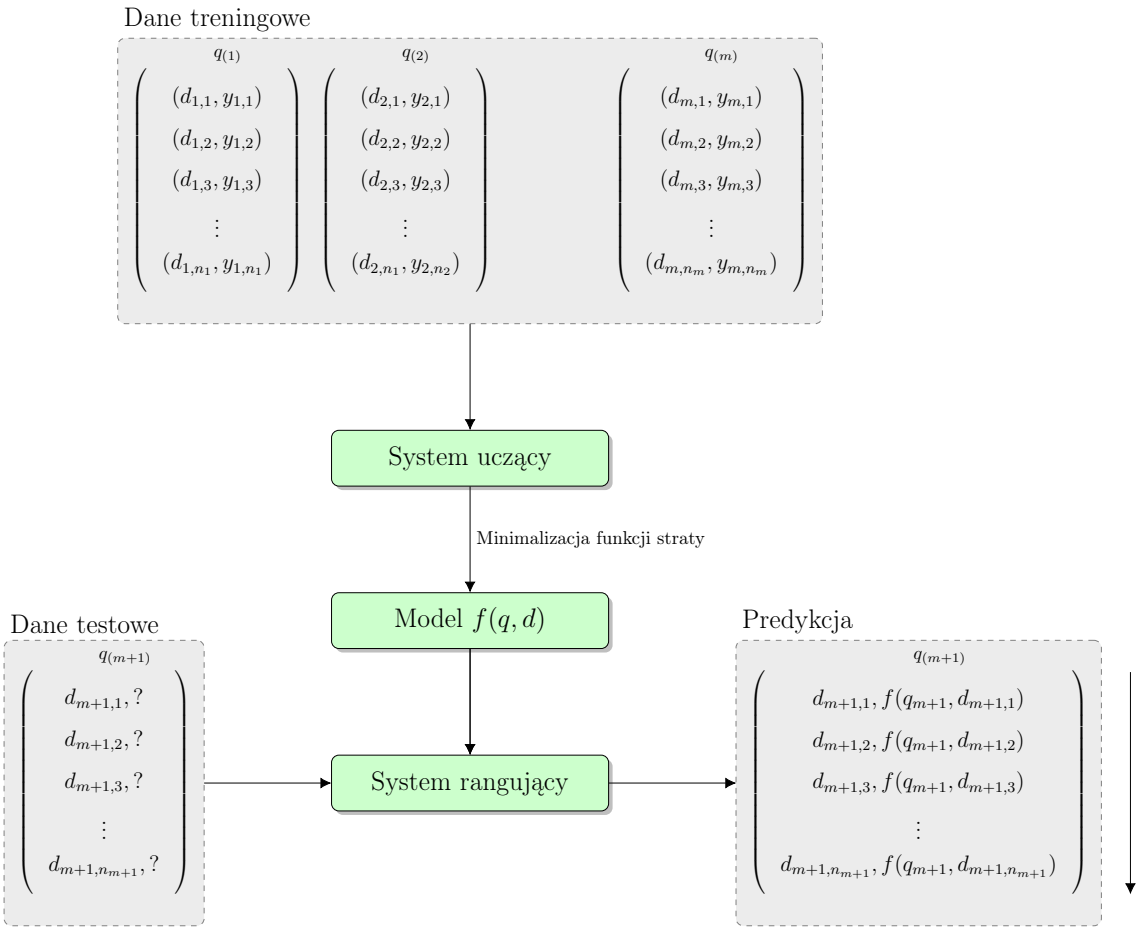
Po fazie treningowej następuje faza testowa, której celem jest sprawdzenie jakości utworzonego modelu. W tym celu wykorzystuje się pewien zbiór zapytań oraz dokumentów, który nie uczestniczył w procesie uczenia się i zwyczajowo nazywany jest zbiorem testowym. Choć przeważnie naukę rangowania wykorzystuje się w kontekście systemów wyszukiwania informacji, to jest to coraz bardziej popularne podejście, wykorzystywane również w kontekście systemów rekomendacji [91].

6.2 Formalna definicja uczenia się rangowania

Formalnie problem uczenia się rangowania możemy zapisać w następujący sposób [33]. Przyjmijmy pewien zbiór zapytań $Q = \{q_1, \ldots, q_{|Q|}\}$ oraz pewien zbiór dokumentów $D = \{d_1, \ldots, d_{|D|}\}$. Zbiór treningowy jest utworzony na podstawie zbioru par zapytanie-dokument $(q_o, d_p) \in Q \times D$. Ponadto para zapytanie-dokument (q_o, d_p) jest reprezentowana przez wektor cech $\phi(q_o, d_p)$. Funkcję rangującą, która jest aproksymatorem liniowym [48, s. 441], zapisujemy w następujący sposób:

$$f(q_o, d_p) = w^T \phi(q_o, d_p), \tag{6.1}$$

gdzie w oznacza wektor, w którym poszczególne elementy tego wektora określają stopień istotności odpowiadającej cechy w wektorze cech $\phi(q_o, d_p)$. W celu utworzenia rankingu dla danego zapytania q_o, obliczamy funkcję $f\,(q_o,\ d_p)$ dla każdego dokumentu d_p, używając do tego wzoru (6.1) i sortujemy te dokumenty w kolejności malejącej. Schemat systemu rangującego został zaprezentowany na rysunku 6.1.



Rysunek 6.1: Schemat systemu rangującego
Opracowanie własne

6.3 Typy algorytmów wykorzystywane w nauce rangowania

Algorytmy wykorzystywane w nauce rangowania można podzielić na 3 typy, co zostało zaprezentowane również na rysunku 6.2:

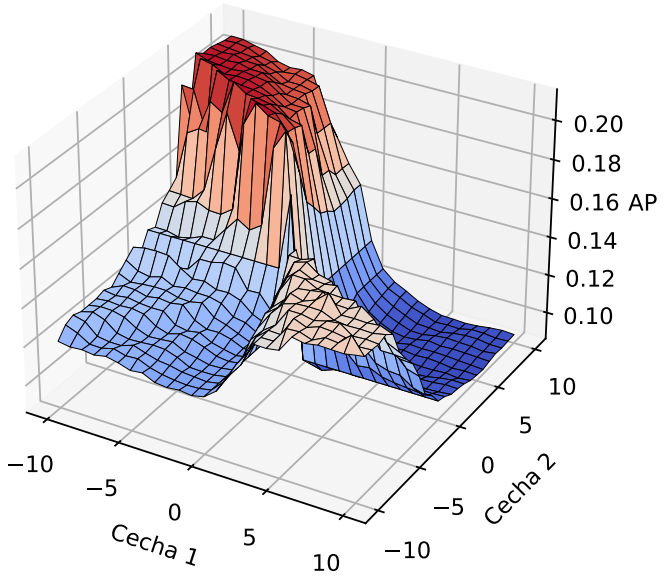
- Punktowe (z ang. *pointwise*) – w tym podejściu każdemu obiektowi w systemie, przyporządkowywana jest pewna wartość, ale podczas jej obliczania nie jest brana pod uwagę kolejność wybranego obiektu, względem innych obiektów znajdujących się w systemie. Algorytmy wykorzystywane w tym podejściu najczęściej bazują na zadaniu regresji, a przykładami takich algorytmów są: McRank [106], Prank [52], OC SVM [154].
- Oparte na parach (z ang. *pairwise*) – w tym podejściu porównywane są ze sobą pary obiektów, co umożliwia określenie kolejności w jakiej powinny zostać zaprezentowane użytkownikowi. Algorytmy wykorzystywane w tym podejściu najczęściej bazują na klasyfikatorze binarnym. Przykładami takich algorytmów są: RankBoost [71], RankNet [43], Lambda Rank [44], LambdaMART [181].

- Oparte na listach (z ang. *listwise*) – zasadniczą wadą opisanych wcześniej technik jest to, że podczas tworzenia rankingu nie rozpatrują one pozycji, na jakich znajdują się wszystkie obiekty w takim rankingu, tylko upraszczają ten problem, analizując go z perspektywy pojedynczego obiektu lub pary obiektów. Możliwe jest jednak zamodelowanie tego problemu w sposób bezpośredni i właśnie w podejściu opartym na listach, w fazie treningowej, rozpatruje się jest pełny ranking obiektów.

  Choć podejście to jest najbardziej zbliżone do rzeczywistego wykorzystania systemów wyszukiwania informacji, to optymalizacja całego rankingu obiektów jest sporym wyzwaniem, ponieważ najczęściej sprowadza się ona do optymalizacji miar służących do oceny jakości rankingu (np. AP, NDCG). Niestety miary te bazują na operacji sortowania [31] (przykładowa wizualizacja miary AP na rys. 6.3). Z tego względu nie mogą one być optymalizowane przy wykorzystaniu klasycznych algorytmów optymalizacyjnych.

  Należy jednak zaznaczyć, że podejście to uzyskuje najlepsze rezultaty w porównaniu do pozostałych metod [46]. Przykładami algorytmów wykorzystujących to podejście są: ListNet [46], ListMLE [182], AdaRank [183].

Rysunek 6.2: Podział podejść wykorzystywanych w nauce rangowania
Opracowanie własne



Rysunek 6.3: Przykładowa wizualizacja krajobrazu funkcji AP, która wykorzystywana jest do oceny jakości rankingu
Opracowanie własne, wykorzystane również w publikacji [37]

Rozdział 7

Zaproponowany algorytm

W tym rozdziale przedstawiony zostanie opis algorytmu, zaproponowanego w niniejszej rozprawie. W pierwszej kolejności omówione zostaną podstawowe informacje na temat podjętej tematyki badawczej i zaprezentowane zostaną argumenty, które uzasadniają wykorzystanie algorytmu metaheurystycznego. Następnie zaprezentowana zostanie ogólna charakterystyka zaproponowanego algorytmu, wraz z pseudokodem i architekturą systemu.

Ponadto wykorzystanie algorytmu bazującego na mechanizmie ewolucji, wiąże się z potrzebą zdefiniowania funkcji przystosowania. W tym rozdziale funkcja ta zostanie omówiona, wraz z zaproponowanym wariantem, który umożliwia sterowanie wpływem, jaki ma zbiór walidacyjny na ogólną jej wartość. Na końcu tego rozdziału zaprezentowana zostanie również autorska modyfikacja, która w procesie agregacji uwzględnia rankingi użytkowników, którzy znajdują się w sąsiedztwie aktywnego użytkownika u_A.

7.1 Informacje ogólne

W kontekście systemów rekomendacyjnych, zaproponowano wiele technik za pomocą których można wygenerować rekomendację i część z nich została zaprezentowana w rozdziale 2. Pomimo tego, że techniki te różnią się między sobą zaproponowanym podejściem oraz stopniem złożoności, to przeważnie efekt ich działania jest taki sam, czyli wygenerowanie rekomendacji, które zostaną zaprezentowane użytkownikowi w formie uporządkowanej listy sugerowanych przedmiotów. Aby wygenerować taką listę (ranking), algorytmy rekomendacyjne przeważnie przyporządkowują pewną wartość (np. liczbę rzeczywistą), każdemu z przedmiotów występujących w systemie i na podstawie tych wartości, określana jest pozycja danego przedmiotu na liście (rankingu).

Okazuje się jednak, że rekomendacje wygenerowane przez poszczególne algorytmy mogą znacząco różnić się od siebie, co stwarza okazję do poprawy końcowego rankingu, poprzez agregację wyników różnych algorytmów. Problem ten przedstawiony jest w literaturze jako problem agregacji rang i został on dokładniej opisany w rozdziale 5. Jeżeli nie mamy dostępu do zbioru treningowego, to przeważnie do generowania agregacji wykorzystuje się proste techniki pozycyjne 5.2. Jednak zasadniczą zaletą pracy z systemami

rekomendacyjnymi jest to, że przeważnie są to systemy spersonalizowane, których głównym zadaniem oprócz samego generowania rekomendacji jest zbieranie i przetwarzanie historycznych interakcji użytkownika z systemem w celu zamodelowania jego preferencji.

Takie historyczne dane można również potraktować jako zbiór treningowy i wykorzystać koncepcje nauki rangowania (opisanej w rozdziale 6) do skonstruowania modelu rangującego, którego zadaniem będzie wygenerowanie rekomendacji dopasowanych do preferencji aktywnego użytkownika u_A. Jednak utworzenie takiego modelu nie jest problemem trywialnym, szczególnie gdy chcemy optymalizować cały ranking przedmiotów, który jest prezentowany użytkownikowi. Jest to związane z tym, że miary wykorzystywane do ewaluacji takich rankingów (np. AP, NDCG) posiadają cechy, które czynią je trudnymi w bezpośredniej optymalizacji (np. nieróżniczkowalność) i z tego względu wykorzystany zostanie algorytm DE do ich bezpośredniej optymalizacji. Optymalizacja całego rankingu przedmiotów w kontekście nauki rangowania nazywana jest podejściem *opartym na listach* i część środowiska badawczego uważa, że zamodelowanie tego problemu w taki sposób daje najlepsze rezultaty [46] (w porównaniu do metod *punktowych* oraz *opartych na parach*).

Ze względu na to, że zaproponowany algorytm będzie bazował na algorytmie DE, należy zwrócić uwagę na to, że obliczanie zaproponowanej funkcji oceny jest procesem bardzo kosztownym. Ma to związek z tym, że w systemach rekomendacyjnych znajduje się wielu użytkowników dla których trzeba wyszukać wektor preferencji, a funkcja oceny musi obliczyć regułę rangującą dla każdego z przedmiotów znajdujących się w systemie. Dodatkowo aby wybrać *TopN* przedmiotów, muszą być one wcześniej posortowane zgodnie z przyporządkowaną wartością. Z tego względu zaproponowano modyfikację, która w celu przyspieszenia obliczeń wykorzystuje reprezentację macierzową populacji algorytmu DE oraz wygenerowanych rankingów. Szczegóły implementacyjne tej modyfikacji można znaleźć w publikacji [36].

7.2 Ogólna charakterystyka zaproponowanego algorytmu

Celem algorytmu *ewolucyjnej agregacji rang* (EAR) zaproponowanego w rozprawie jest poprawa jakości generowanych rekomendacji, poprzez agregację rankingów, które zostały wygenerowane przez poszczególne algorytmy rekomendacyjne w zbiorze S. Takie podejście umożliwia utworzenie nowego rankingu τ_A^*, dopasowanego do preferencji aktywnego użytkownika u_A.

Algorytm EAR poszukuje wektor preferencji w_{u_A} dla aktywnego użytkownika u_A, poprzez optymalizację funkcji przystosowania na zbiorze treningowym TS. Wektor w_{u_A} określa stopień w jakim poszczególne algorytmy uczestniczą w procesie agregacji, ponieważ każdy element tego wektora jest przyporządkowany do jednego algorytmu. Wektor preferencji w_{u_A} wykorzystywany jest w funkcji rangującej wyrażonej wzorem:

$$f(u_A, x_j) = w_{u_A}^T \phi(u_A, x_j), \tag{7.1}$$

gdzie $\phi(u_A, x_j)$ jest to reprezentacja wektorowa przedmiotu $x_j \in I$ dla danego użytkownika u_A, a elementy tego wektora dopowiadają wartościom przyporządkowanym przez poszczególne algorytmy rekomendacyjne ze zbioru S.

Należy pamiętać, że wartości tych cech będą inne dla każdego z użytkowników w systemie, bo dla każdego użytkownika algorytmy rekomendacyjne wygenerują inne rekomendacje. Z tego względu wektor w_{u_A} jest tworzony osobno dla każdego z użytkowników, ponieważ każdy z użytkowników posiada swoje indywidualne preferencje dotyczące rekomendacji.

Ze względu na to, że funkcja rangująca 7.4, będzie optymalizowana poprzez bezpośrednią optymalizację miary AP, to do wyszukania wektora w_{u_A} wykorzystany zostanie algorytm DE, który świetnie nadaje się do rozwiązywania wielowymiarowych problemów optymalizacyjnych. Algorytm ten poprzez operatory mutacji oraz krzyżowania zmienia populację osobników, aby w procesie selekcji premiować osobniki z wyższą wartością funkcji przystosowania. Funkcja przystosowania zostanie opisana w kolejnym podrozdziale 7.3.

Podsumowując, wykorzystanie algorytmu metaheurystycznego jest uzasadnione tym, że algorytmy tego typu umożliwiają bezpośrednią optymalizację miar, wykorzystywanych do ewaluacji całej listy zarekomendowanych przedmiotów (podejście *oparte na listach* opisane w rozdziale 6). Pseudokod zaproponowanego algorytmu znajduje się poniżej 2, a architektura systemu rekomendacyjnego została zaprezentowana na rysunku 7.1 na którym widać, że proces rekomendacji jest podzielony na dwie fazy. W pierwszej fazie algorytmy rekomendacyjne generują rekomendacje w formie rankingów dla aktywnego użytkownika u_A. W drugiej fazie na bazie zbioru treningowego użytkownika u_A, algorytm DE wyszukuje optymalny wektor wag w_{u_A}.

7.3 Funkcja przystosowania

Wykorzystanie algorytmu bazującego na mechanizmie ewolucji, wiąże się z wymogiem zdefiniowania funkcji przystosowania, nazywanej również funkcją oceny. Dzięki niej do kolejnych generacji będą mogły przechodzić osobniki najlepiej przystosowane, a cały algorytm będzie dążyć do znalezienia *optymalnego* rozwiązania. W przypadku algorytmu zaproponowanego w niniejszej rozprawie, głównym elementem funkcji przystosowania będzie miara AP (opisana w rozdziale 3.2.2), która jest obliczana dla aktywnego użytkownika u_A zgodnie z następującym wzorem:

$$Fitness(w_{u_A}, RT, ST) = AP@k(w_{u_A}, RT, ST), \tag{7.2}$$

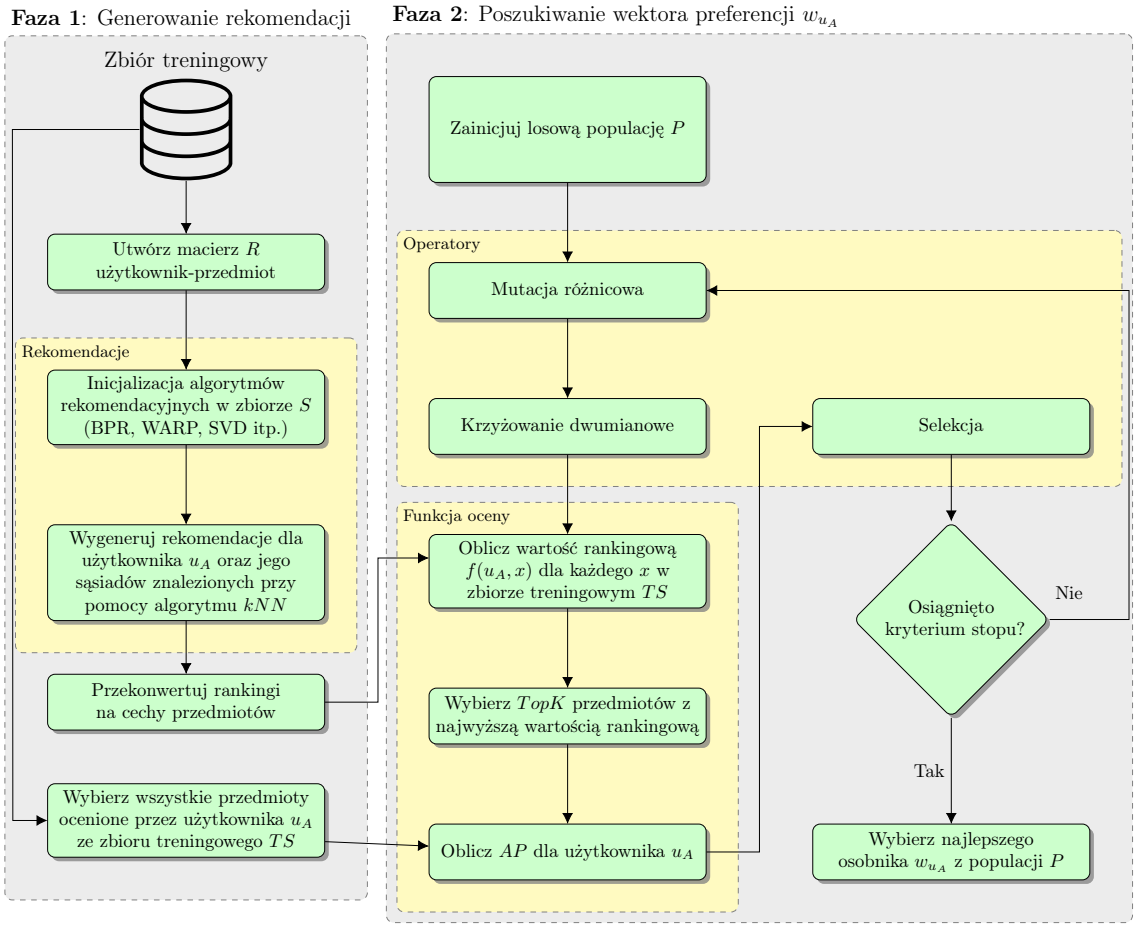
Algorytm 2 Pseudokod algorytmu ewolucyjnej agregacji rang (EAR)

Wejście: **S** – zbiór algorytmów rekomendacyjnych, **I** – zbiór przedmiotów, **TS** – zbiór treningowy, $\boldsymbol{u_A}$ – użytkownik, dla którego generowane będą rekomendacje
Wyjście: τ_A^* - rekomendacja dopasowana do preferencji użytkownika $\boldsymbol{u_A}$

```
1:  T = []                                              ▷ utworzenie kolekcji
2:  τ*_A = []                                           ▷ utworzenie kolekcji
3:  AR := zainicjuj_algorytmy_rekomendacyjne(S)
4:  AR := trenuj_algorytmy_rekomendacyjne(AR, TS)
5:
6:  for ar in AR do
7:      τ = wygeneruj_rekomendacje(u_A, ar)
8:      T.add(τ)
9:  end for
10:
11: C := przekształć_do_postaci_macierzy(T)
12: C := normalizacja(C, "min-max")
13: w_uA := wyszukaj_wektor_preferencji(C, TS, u_A)     ▷ algorytm DE
14:
15: for x in I do
16:     φ(u_A, x) := pobierz_reprezentację_wektorową(C, x)
17:     f := w_uA^T φ(u_A, x)                           ▷ funkcja rangująca
18:     τ*_A.add(x, f)
19: end for
20:
21: τ*_A = sortuj_rosnąco(τ*_A)
22: τ*_A = wybierz_topN_przedmiotów(τ*_A, 10)
23: return τ*
```

gdzie ST jest to zbiór przedmiotów, które użytkownik u_A ocenił w swoim zbiorze treningowym TS, a RT jest to zbiór przedmiotów zarekomendowanych przez system. Formuła 7.2 oznacza średnią precyzję, jaka zostałaby obliczona pomiędzy zbiorami RT oraz ST. Do wygenerowania zbioru RT wykorzystywany jest wektor wag w_{u_A} w funkcji rangującej 7.4, gdzie wektor ten jest pojedynczym osobnikiem w algorytmie DE. Bazując na wynikach eksperymentów, najlepszą wartością K dla miary $AP@K$ podczas ewaluacji funkcji oceny jest liczba przedmiotów, które użytkownik u_A posiada w swoim zbiorze treningowym TS, czyli $k = |ST|$.

Podczas prowadzonych badań zaobserwowano, że wysoka wartość funkcji oceny na zbiorze treningowym TS, nie przekłada się bezpośrednio na wysoką jakość generowanych rekomendacji. Z tego względu zaproponowana została modyfikacja funkcji oceny, która uwzględnia dodatkowo miarę AP obliczaną na zbiorze walidacyjnym VS, który to zbiór został wydzielony ze zbioru treningowego TS. Taka modyfikacja umożliwia lepszą generalizację wektora w_{u_A}, ponieważ algorytm DE optymalizuje go w stosunku do tych dwóch zbiorów jednocześnie. Funkcja oceny w zmodyfikowanej formie wyrażona jest



Rysunek 7.1: Architektura systemu rekomendacyjnego
Rysunek wykorzystano również w publikacji [21]

następującym wzorem:

$$Fitness(w_{u_A}, RT, ST, SV) = AP@K(w_{u_A}, RT, ST) + \lambda\ AP@K(w_{u_A}, RT, SV), \tag{7.3}$$

bazując na wzorze (7.2) do wzoru (7.3) wprowadzono dodatkowy zbiór SV. Zbiór SV jest to zbiór przedmiotów ocenionych przez użytkownika u_A w zbiorze walidacyjnym VS.

Dodatkowo zaproponowano wprowadzenie dodatkowego parametru λ, który określa stopień w jakim wartość miary AP obliczona na zbiorze walidacyjnym, będzie wpływać na ogólną wartość funkcji oceny. Ta modyfikacja jest podyktowana tym, że zbiór walidacyjny jest zazwyczaj mniejszy od zbioru treningowego i aby wyrównać wpływ tego zbioru na ogólną wartość funkcji oceny, zaproponowano ten parametr. Bazując na wynikach badań, sugerowaną początkową wartością tego parametru jest $\lambda = \frac{|ST|}{|SV|}$.

Inspiracją do wykorzystania funkcji oceny w takiej formie była publikacja [33], w której to autorzy wykorzystali algorytm DE do poszukiwania optymalnej funkcji rangującej w kontekście systemów wyszukiwania informacji. W odróżnieniu jednak od algorytmu zaproponowanego w niniejszej rozprawie w funkcji oceny wykorzystywali oni miarę MAP,

ponieważ funkcja rangująca była optymalizowana w stosunku do wszystkich zapytań dostępnych w systemie. Bazując jednak na wynikach eksperymentów w kontekście systemów rekomendacji, bardziej zasadne wydaje się być poszukiwanie wektora preferencji w stosunku do jednego użytkownika. Takie podejście można uzasadnić tym, że każdy z użytkowników będzie posiadał swoje indywidualne preferencje dotyczące generowanych rekomendacji.

Oczywiście zastosowanie w funkcji oceny innej miary niż AP również byłoby uzasadnione (np. NDCG). Miara AP została wybrana ze względu na jej popularność w publikacjach naukowych, gdzie jest często wykorzystywana do oceny jakości generowanych rekomendacji.

7.4 Modyfikacja uwzględniająca rankingi najbliższych sąsiadów

W kontekście systemów rekomendacyjnych, agregacja najczęściej odbywa się na bazie rankingów (rekomendacji), które zostały wygenerowane dla aktywnego użytkownika u_A. Przykładowo jeżeli w systemie mamy dostępne 4 algorytmy generujące rekomendacje, to dla danego użytkownika u_A wygenerują one 4 listy z rekomendacjami (rankingi), które następnie wezmą udział w procesie agregacji. W celu usprawnienia algorytmu *EAR*, zaproponowano modyfikację, która uwzględnia dodatkowe rankingi wygenerowane dla innych użytkowników w systemie. W celu wyszukania takich użytkowników wykorzystany został algorytm *kNN*, który oblicza podobieństwo pomiędzy użytkownikiem u_A, a innymi użytkownikami $v \in U$, wykorzystując do tego celu miarę euklidesową.

Po obliczeniu miary d pomiędzy wszystkimi użytkownikami $v \in U$, wybierane jest k użytkowników, których odległość d jest najmniejsza i na tej podstawie tworzone jest sąsiedztwo $N \subset U$. Następnie rankingi wygenerowane dla użytkowników w zbiorze N są włączane do procesu agregacji. Należy zaznaczyć, że w tej wersji modyfikacji wszystkie rankingi, które zostały wygenerowane dla danego użytkownika przez algorytmy rekomendacyjne są uwzględnianie w tym procesie, aczkolwiek można ten proces zmodyfikować i wybrać tylko pewien ich podzbiór.

Następnie rankingi aktywnego użytkownika u_A oraz jego sąsiadów w zbiorze N są przekształcane do postaci wektorowej (cech przedmiotów), analogicznie jak było to realizowane w podstawowej wersji algorytmu. Funkcja rangująca będzie miała postać:

$$f(u_A, N, x_j) = w_{u_A}^T \phi(u_A, N, x_j), \tag{7.4}$$

gdzie $\phi(u_A, N, x_j)$ jest to reprezentacja wektorowa przedmiotu $x_j \in I$ dla danego użytkownika u_A, która uwzględnia dodatkowe rankingi wygenerowane dla grupy użytowników $N \subset U$, należących do najbliższego sąsiedztwa użytkownika u_A. Należy pamiętać, że uwzględnienie dodatkowych rankingów wymusza rozszerzenie wektora w_{u_A}, aby każdy

element wektora cech $\phi(u_A, N, x_j)$ miał przyporządkowany odpowiadający mu element wektora w_{u_A}.

Na rysunku 7.2 przedstawiono proces agregacji, z wykorzystaniem zaproponowanej modyfikacji algorytmu *EAR*. Proces agregacji obejmuje rekomendacje wygenerowane dla aktywnego użytkownika u_A (obszar zielony) oraz rekomendację utworzone dla najbliższych sąsiadów aktywnego użytkownika (obszar żółty). Dodatkowo pseudokod 3 przedstawia algorytm zaproponowanej modyfikacji.

Główną inspiracją do zaproponowania tej modyfikacji była technika zbiorowej filtracji (opisana w rozdziale 2.5.2), w której to w celu wygenerowania rekomendacji dla aktywnego użytkownika u_A, wyszukuje się innych użytkowników o podobnych preferencjach. Następnie preferencje takiej grupy użytkowników są agregowane w celu wygenerowania rekomendacji dla aktywnego użytkownika u_A. Bazując na tej idei w zaproponowanej modyfikacji również uwzględniono taką dodatkową informację pochodzącą od innych użytkowników, właśnie w postaci wygenerowanych dla nich rekomendacji (rankingów).

Jeszcze jedną wskazówką, która sugerowała, że wykorzystanie takiej modyfikacji może być uzasadnione, były prace badawcze prowadzone w kontekście grupowych systemów rekomendacji [85]. Grupowe systemy rekomendacyjne są to systemy, które w odróżnieniu do ich klasycznych odpowiedników są dopasowane do preferencji całej grupy, a nie tylko do jednego, konkretnego użytkownika [34]. Przykładem zastosowania tego typu systemów może być np. wybór filmu do obejrzenia przez pewną grupę znajomych. System w tym wypadku musi spróbować tak uogólnić wygenerowane rekomendację, żeby były one satysfakcjonujące dla jak największej liczby osób w takiej grupie.

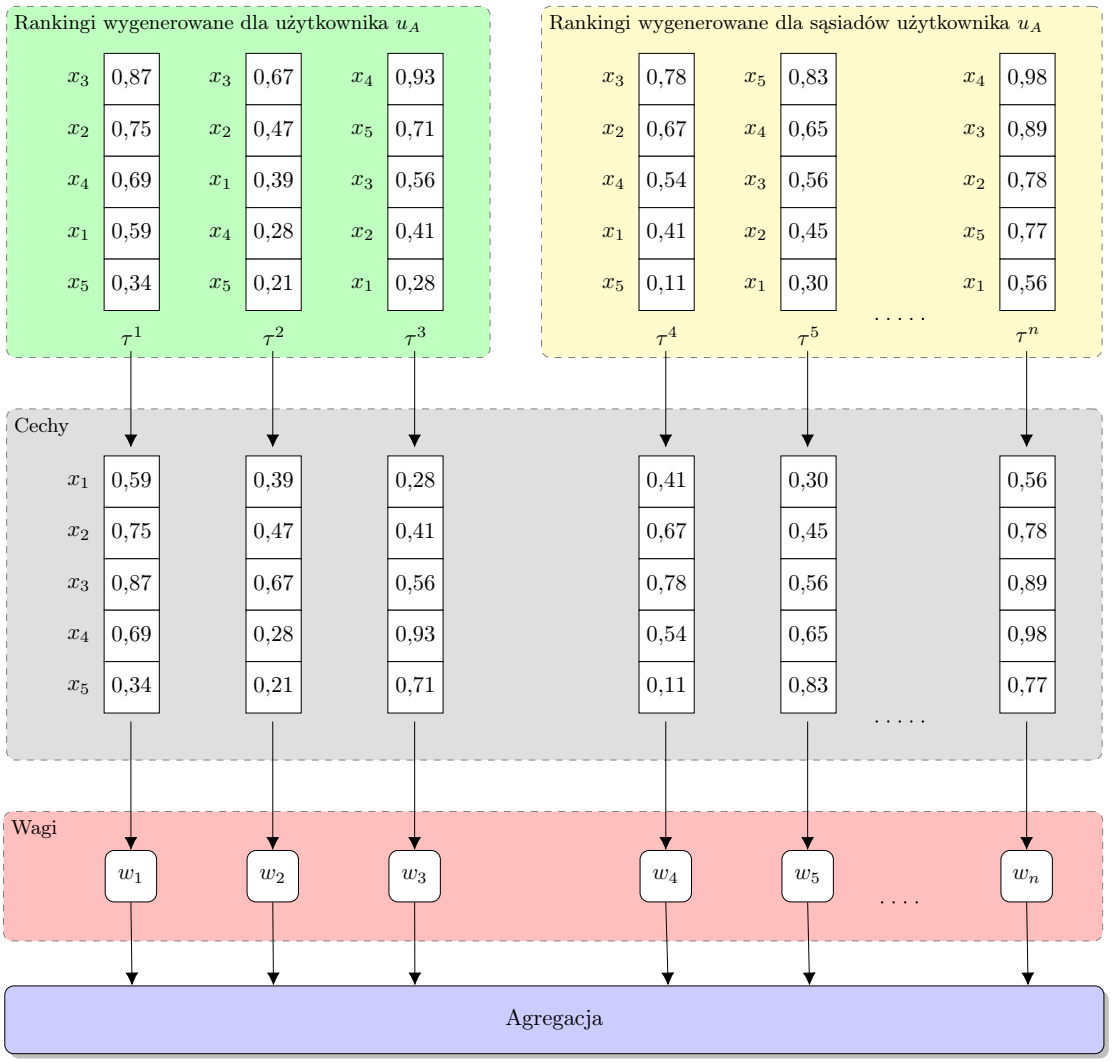
W publikacji [16] przedstawiono ciekawe wyniki badań sugerujące, że rekomendacje wygenerowane dla grupy użytkowników mogą poprawić ogólną jakość rekomendacji dla części użytkowników należących do takiej grupy, w porównaniu do rekomendacji spersonalizowanych. Jest to zaskakująca obserwacja, ponieważ można byłoby się spodziewać, że takie uogólnione rekomendację muszą być gorsze od rekomendacji wygenerowanych indywidualnie dla poszczególnych użytkowników. Autorzy sugerują, że taka sytucja jest prawdopodobnie spowodowana tym, że system generował spersonalizowane rekomendacje niskiej jakości. W innej publikacji [167] autorzy zauważyli, że wyszukując podobnych użytkowników i agregując wyniki wygenerowanych dla nich rekomendacji, zaproponowanym przez siebie algorytmem bazującym na metodzie Bordy, można zwiększyć jakość takich rekomendacji.

Analizując wnioski zaprezentowane w omówionych wcześniej publikacjach można było podejrzewać, że uwzględnienie takich dodatkowych rankingów w procesie agregacji może poprawić ogólną jakość generowanych rekomendacji.

Algorytm 3 Pseudokod modyfikacji algorytmu ewolucyjnej agregacji rang (EAR)

Wejście: **S** – zbiór algorytmów rekomendacyjnych, **U** – zbiór użytkowników, **I** – zbiór przedmiotów, **k** – parametr określający liczbę sąsiadów, **TS** – zbiór treningowy, $\boldsymbol{u_A}$ – użytkownik, dla którego generowane będą rekomendacje
Wyjście: τ_A^* - rekomendacja dopasowana do preferencji użytkownika $\boldsymbol{u_A}$

```
1: T = []                                                  ▷ utworzenie kolekcji
2: τ*_A = []                                               ▷ utworzenie kolekcji
3: AR := zainicjuj_algorytmy_rekomendacyjne(S)
4: AR := trenuj_algorytmy_rekomendacyjne(AR, TS)
5: U_NN := kNN(u_A, U, k)                                  ▷ sąsiedztwo użytkownika u_A
6:
7: for ar in AR do
8:     τ = wygeneruj_rekomendacje(u_A, ar)
9:     T.add(τ)
10:
11:    for u in U_NN do
12:        τ = wygeneruj_rekomendacje(u, ar)
13:        T.add(τ)                                        ▷ rekomendacje sąsiadów
14:    end for
15: end for
16:
17: C := przekształć_do_postaci_macierzy(T)
18: C := normalizacja(C, "min-max")
19: w_uA := wyszukaj_wektor_preferencji(C, TS, u_A)        ▷ algorytm DE
20:
21: for x in I do
22:    φ(u_A, x) := pobierz_reprezentację_wektorową(C, x)
23:    f := w_uA^T φ(u_A, x)                               ▷ funkcja rangująca
24:    τ*_A.add(x, f)
25: end for
26:
27: τ*_A = sortuj_rosnąco(τ*_A)
28: τ*_A = wybierz_topN_przedmiotów(τ*_A, 10)
29: return τ*
```



Rysunek 7.2: Modyfikacja algorytmu *EAR*
Opracowanie własne, wykorzystane również w publikacji [22]

Rozdział 8

Przygotowanie środowiska badawczego

W tym rozdziale przedstawione zostanie środowisko eksperymentalne, które zostało stworzone na potrzeby przeprowadzonych badań. W pierwszej kolejności omówiony zostanie zbiór danych *MovieLens 100k*. Następnie przedstawione zostaną szczegóły dotyczące środowiska eksperymentalnego oraz objaśniony zostanie proces dostrajania parametrów algorytmów rekomendacyjnych. Na końcu przedstawiona zostanie metodyka przeprowadzonych eksperymentów.

8.1 Zbiór danych MovieLens 100k

W niniejszej rozprawie eksperymenty zostały przeprowadzone na zbiorze danych *MovieLens 100k* [77]. Był to jeden z pierwszych ogólnodostępnych zbiorów danych, który na przestrzeni lat z powodzeniem był wykorzystywany przez licznych badaczy do prowadzenia badań nad systemami rekomendacyjnymi. Utworzony został on w ramach projektu badawczego grupy *GroupLens* na Uniwersytecie Minnesoty w 1998 roku. Dane zostały pozyskane za pośrednictwem strony internetowej, która była spersonalizowanym i niekomercyjnym systemem rekomendacyjnym.

Aby móc korzystać z tego systemu, użytkownicy w pierwszej kolejności musieli założyć w nim konto oraz uzupełnić podstawowe informacje, dotyczących swojej osoby (np. wiek, wykonywany zawód itp.). Następnie użytkownicy mogli oceniać różne filmy w skali od 1 do 5. Wybory użytkowników były zapisywane wraz ze znacznikiem czasu i wykorzystywane przez system rekomendacyjny, który tworzył na ich podstawie pewien profil preferencji aktywnego użytkownika i generował spersonalizowane rekomendacje. System był rozwijany przez lata i obecnie (od 2005 roku) umożliwia wprowadzenie dodatkowych informacji, np. tagów, czyli słów kluczowych dzięki którym użytkownicy opisują własnymi słowami wrażenie, jakie wywarł na nich dany film np.: *nudny*, *brutalny*, *emocjonujący* itp.

Dane, które znajdują się w tym zbiorze zostały zebrane pomiędzy 19 września 1997 roku, a 22 kwietnia 1998 roku. Przed opublikowaniem zostały wcześniej odpowiednio przygotowane i oczyszczone, a użytkownicy, którzy ocenili mniej niż 20 filmów lub nie wprowadzili

pełnych informacji demograficznych zostali usunięci. Należy zaznaczyć, że istnieją również nowsze wersje tego zbioru danych np. *MovieLens 1M, MovieLens 10M, MovieLens 25M* [77]. Nazwy kolejnych wersji oznaczają liczbę wprowadzonych przez użytkowników ocen. Tak naprawdę główną różnicą pomiędzy tymi zbiorami jest liczba użytkowników, filmów oraz ocen, choć niektóre z nich mogą uwzględniać dodatkowe informację np. wprowadzone przez użytkowników tagi.

Do przeprowadzenia badań wykorzystany został jeden z mniejszych zbiorów danych, w którym znajdowało się tylko 100 tysięcy interakcji użytkowników z filmami. Wybór ten był jednak podyktowany tym, że wraz ze wzrostem liczby dostępnych interakcji, wzrastał też czas potrzebny do wytrenowania poszczególnych modeli algorytmów rekomendacyjnych. W efekcie wydłużało to również proces dostrajania ich parametrów. Dodatkowo algorytm DE oblicza funkcję rangującą dla każdego z filmów w systemie, co przy dużej liczbie filmów znacznie wydłużałoby proces obliczania funkcji oceny.

Należy zaznaczyć, że zastosowanie mniejszej wersji zbioru danych wydaje się nie wpływać negatywnie na istotność uzyskanych wyników, co sugeruje aktualna literatura. Starsze i mniej liczne zbiory danych są nadal popularne i chętnie wykorzystywane przez badaczy do przeprowadzania eksperymentów (np. *MovieLens 1M*) [163]. Autorzy w publikacji [12, s. 4] również zwrócili uwagę na ten fakt zaznaczając, że obecnie w publikacjach naukowych zazwyczaj stosuje się mniejsze zbiory danych w stosunku do tych, które były wykorzystywane choćby w 2006 roku podczas konkursu *Netflix Prize* (zawierał on aż 100 milionów interakcji [26]). Taką tendencję autorzy tłumaczą tym, że obecnie algorytmy rekomendacyjne są dużo bardziej złożone, co oczywiście często przekłada się na ich wydłużony czas treningu, a porównywanie się do licznych algorytmów, wiąże się z czasochłonnym procesem dostrajania ich parametrów.

Zbiór danych *MovieLens 100k* jest dostępny w formie skompresowanego folderu w którym znajduje się sześć plików:

- *u.data* – plik zwierający 100 000 ocen, które zostały wystawione przez 943 użytkowników dla 1682 filmów. Na podstawie tego pliku można utworzyć macierz R, która reprezentuje wszystkie interakcje pomiędzy użytkownikami i filmami. Pierwsze pięć rekordów z tego pliku zostało zaprezentowane w tabeli 8.1.

- *u.user* – plik zawierający informacje demograficzne dotyczące użytkowników. Każdy użytkownik reprezentowany jest przez unikalny identyfikator, który wykorzystywany jest w pliku *u.data*. Pierwsze pięć rekordów z tego pliku zostało zaprezentowane w tabeli 8.2.

- *u.item* – plik zawierający informacje dotyczącą filmów, takie jak: tytuł, rok produkcji i gatunki. Każdy film posiada swój unikalny identyfikator, który wykorzystywany jest w pliku *u.data*. Pierwsze pięć rekordów z tego pliku zostało zaprezentowane w tabeli 8.3.

- *u.info* – plik zawierający podstawowe informacje dotyczące zbioru danych (liczba użytkowników, liczba filmów itp.).
- *u.genre* – plik zawierający listę wszystkich gatunków.
- *u.occupation* - plik zawierający listę wszystkich zawodów.

Tabela 8.1: Pięć pierwszych rekordów z pliku *u.data*

Id użytkownika	Id filmu	Ocena	Znacznik czasu
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

Tabela 8.2: Pięć pierwszych rekordów z pliku *u.user*

Id użytkownika	Wiek	Płeć	Zawód	Kod pocztowy
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

Tabela 8.3: Pięć pierwszych rekordów z pliku *u.item*. Usunięto kolumnę *Link*, która zawierała odnośnik do internetowej bazy filmów IMDb (dla zachowania przejrzystości)

Id filmu	Tytuł filmu	Rok produkcji	Przyporządkowane gatunki
1	Toy Story (1995)	01-Jan-1995	|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0|0
2	GoldenEye (1995)	01-Jan-1995	|0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3	Four Rooms (1995)	01-Jan-1995	|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4	Get Shorty (1995)	01-Jan-1995	|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5	Copycat (1995)	01-Jan-1995	|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0

Tabela 8.4: Podstawowe statystyki zbioru danych *MovieLens 100k*

Nazwa statystyki	Wartość
Liczba wystawionych ocen	100 000
Liczba użytkowników	943
Liczba filmów	1682
Możliwe wartości ocen	$\{1, 2, 3, 4, 5\}$
Średnia ocen	3,53
Mediana ocen	4,0
Rzadkość danych	93,695%

W tabeli 8.4 zaprezentowano podstawowe statystyki dotyczące wykorzystywanego zbioru danych *MovieLens 100k*. Należy tutaj zaznaczyć, że charakteryzują go dwie cechy, które odróżniają go od jego większych wersji (np. *MovieLens 10M*, *MovieLens 25M*). Po pierwsze, liczba użytkowników jest mniejsza od liczby filmów, co przy większych zbiorach danych zazwyczaj nie ma miejsca i to użytkowników jest dużo więcej od filmów. Po drugie, warto zwrócić uwagę na rzadkość danych, która w przypadku tego zbioru jest stosunkowo niewielka i wynosi 93,695% (w przypadku większych wersji wynosi ona ponad 99%).

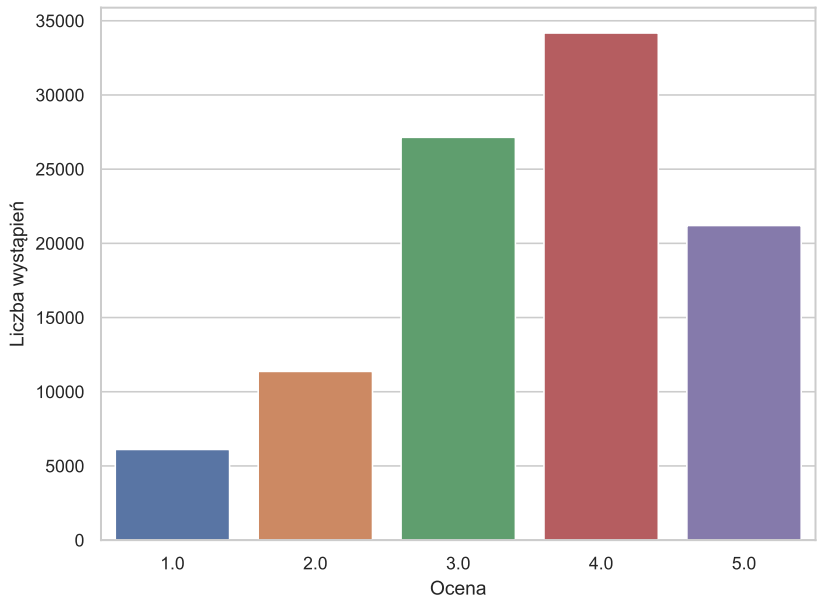
Tabela 8.5: Podstawowe statystyki zbioru danych *MovieLens 100k* po zastosowaniu filtra, który usunął wszystkie filmy, które zostały ocenione mniej niż 50 razy

Nazwa statystyki	Wartość
Liczba wystawionych ocen	83 715
Liczba użytkowników	943
Liczba filmów	603
Możliwe wartości ocen	$\{1, 2, 3, 4, 5\}$
Średnia ocen	3,63
Mediana ocen	4,0
Rzadkość danych	85,278 %

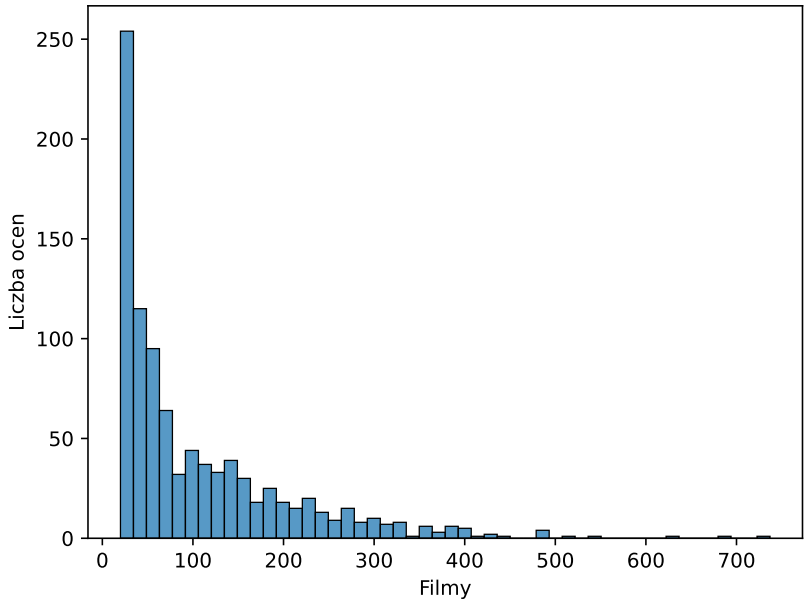
Ze względu na to, że algorytm zaproponowany w niniejszej rozprawie jest algorytmem mającym na celu personalizację generowanych rekomendacji, to popularnym procesem wstępnego przetworzenia danych w kontekście spersonalizowanych systemów rekomendacyjnych jest usunięcie użytkowników i filmów, które rzadko wchodząc ze sobą w interakcje [12, s. 4]. Zbiór danych *MovieLens 100k* był już wcześniej oczyszczony przez jego autorów z użytkowników, którzy ocenili małą liczbę filmów (mniej niż 20). Z tego względu aby

dodatkowo zredukować wpływ filmów, które były rzadko oceniane przez użytkowników, usunięto również filmy, które były ocenione mniej niż 50 razy. Efekt zastosowania tego filtra został zaprezentowany w tabeli 8.5, gdzie widać, że liczba wystawionych przez użytkowników ocen w tym zbiorze spadła o 16 285 (z 100 000 do 83 715), a liczba dostępnych filmów spadła o 1079 (z 1682 do 603). Spowodowało to również zmniejszenie rzadkości danych z 93,695 % do 85,278 %.

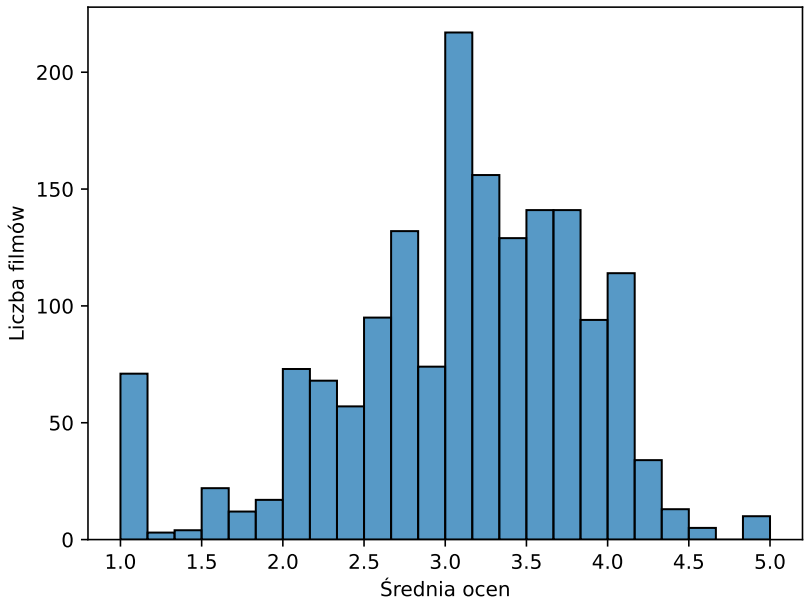
W ramach analizy wykorzystywanego zbioru danych, poniżej zaprezentowano kilka histogramów, które pozwalają na jego krótką charakterystykę. Na wykresie 8.1 zaprezentowany został rozkład wystawionych przez użytkowników ocen. Widać, że najczęściej wystawianą oceną była ocena 4, a najrzadziej 1. Przyjmując, że za pozytywną ocenę uznajemy ocenę od wartości 3, to analizując ten wykres widać ogólne przesunięcie wartości w stronę ocen bardziej pozytywnych. Może to świadczyć o tym, że użytkownicy byli bardziej skłonni wystawiać oceny pozytywne od negatywnych, choć może to również oznaczać, że filmy które oceniali negatywnie były oceniane przez nich rzadziej. Dodatkowo na wykresie 8.2 można zauważyć charakterystyczny dla systemów rekomendacyjnych problem *długiego ogona*, który był opisany w podrozdziale 2.6.1, a na wykresach 8.3 oraz 8.4 zaprezentowano rozkład średniej ocen dla filmów i dla użytkowników.



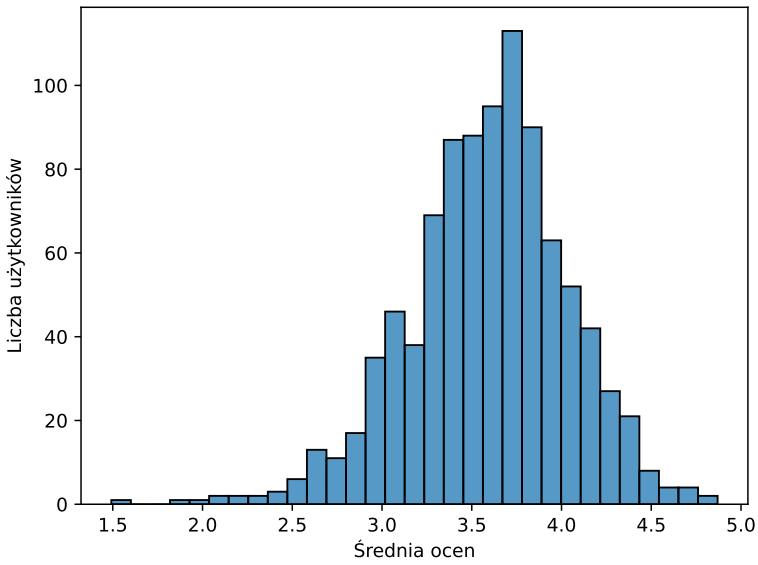
Rysunek 8.1: Histogram prezentujący rozkład ocen w zbiorze *MovieLens 100k*



Rysunek 8.2: Histogram prezentujący rozkład ocen wystawionych dla poszczególnych filmów w zbiorze *MovieLens 100k*



Rysunek 8.3: Histogram prezentujący rozkład średniej ocen filmów w zbiorze *MovieLens 100k*



Rysunek 8.4: Histogram prezentujący rozkład średniej ocen użytkowników w zbiorze *MovieLens 100k*

8.2 Implementacja środowiska badawczego

Do poprawnego działania systemy rekomendacyjne potrzebują licznych funkcjonalności, których implementacja może być bardzo czasochłonna. Na szczęście na przestrzeni lat powstało kilka bibliotek programistycznych, które znacznie ułatwiają proces przygotowania środowiska badawczego oraz redukują szansę wystąpienia ewentualnych błędów. Autor niniejszej rozprawy zdecydował się na wykorzystanie języka Python oraz bibliotek, które zostały wyszczególnione w tabeli 8.6.

Podstawą środowiska badawczego jest biblioteka *LensKit* [64], która umożliwia: wczytywanie i podział danych na odpowiednie zbiory (treningowy, walidacyjny oraz testowy), oczyszczenie zbioru danych, utworzenie macierzy interakcji R oraz ewaluację wygenerowanych rekomendacji za pomocą licznych miar jakości. Implementacja algorytmu DE została zaczerpnięta z biblioteki *Pymoo* [30], która jest popularną biblioteką napisaną w języku Python, wykorzystywaną do optymalizacji wielokryterialnej. Dodatkowo ze względu na to, że zaproponowany algorytm jest algorytmem agregującym, to zostanie on porównany z innymi technikami wykorzystywanymi do tego celu. Zostały one opisane w rozdziale 5, a ich kod źródłowy jest dostępny w bibliotece *Ranx* [17].

W tabeli 8.7 znajduje się zestawienie wszystkich algorytmów, które zostały wykorzystane do generowania rekomendacji w fazie eksperymentalnej, ze wskazaniem źródła na podstawie którego utworzona została dana implementacja. Eksperymenty przeprowadzono na komputerze Intel Core i5-7600 (3,50 GHz) z 16GB RAM.

Tabela 8.6: Podsumowanie głównych bibliotek, które zostały wykorzystane podczas implementowania środowiska badawczego

Nazwa biblioteki	Wersja	Zastosowanie biblioteki	Język	Referencja
LensKit	0.14.2	System rekomendacyjny	Python	[64]
Pymoo	0.6	Algorytm DE	Python	[30]
Rankx	0.3	Techniki agregujące	Python	[17]
Optuna	3.0.3	Dostrajanie parametrów	Python	[9]

Tabela 8.7: Algorytmy rekomendacyjne wykorzystane w eksperymentach

Algorytm	Bazująca na	Referencja
UserUser	Sąsiedztwie	[64]
ItemItem	Sąsiedztwie	[58]
PureSVD	Faktoryzacji macierzy	[76]
ImplicitMF	Faktoryzacji macierzy	[83]
BPR	Faktoryzacji macierzy	[139]
MostPopular	Najpopularniejszych przedmiotach	[64]
Random	Losowych przedmiotach	Autorska implementacja

8.3 Dostrajanie parametrów algorytmów rekomendacyjnych

Przed przystąpieniem do generowania rekomendacji, parametry algorytmów rekomendacyjnych muszą zostać odpowiednio dostrojone. Celem tego procesu jest znalezienie takiego zestawu parametrów, który maksymalizuje jakość generowanych rekomendacji (wyrażonej przy pomocy miary MAP) na zbiorze walidacyjnym. Krok ten jest niezwykle istotny, ponieważ algorytmy rekomendacyjne posiadają dużą liczbę parametrów, które bez odpowiedniego dostrojenia mogą działać nieefektywnie.

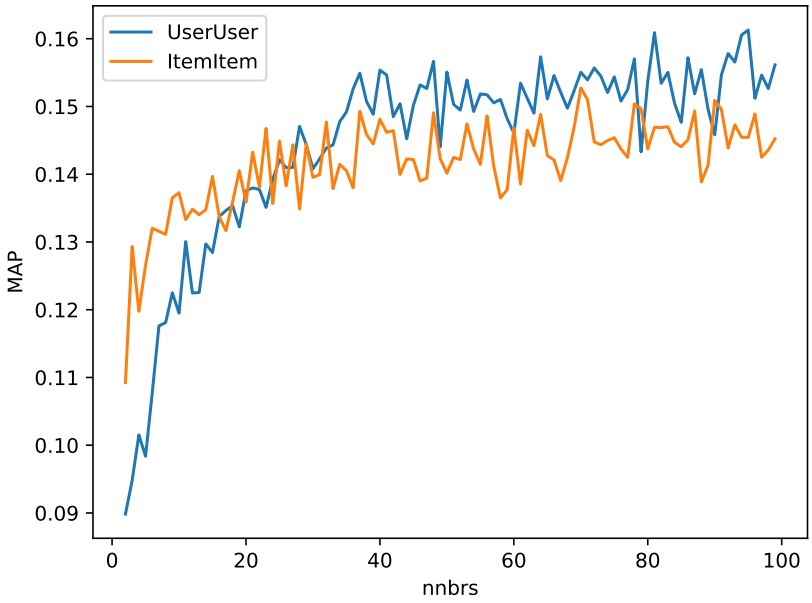
Proces dostrajania został zrealizowany przy pomocy biblioteki *Optuna* [9], która jest biblioteką umożliwiającą zautomatyzowanie tego procesu (podobnie jak popularna biblioteka *HyperOpt* [28]). Algorytmem dostrajającym był *Tree-structured Parzen Estimator* [29], który na bazie historii wcześniejszych wartości parametrów, tworzy model probabilistyczny, a następnie wykorzystuje go do sugerowania kolejnych wartości parametrów. Aby proces dostrajania nie był zbyt długi, można wcześniej określić pewien limit prób, który na potrzeby tej rozprawy wynosił 100.

Tabela 8.8: Parametry algorytmów rekomendacyjnych wraz z typem i zakresem wartości, które zostały wykorzystane podczas procesu dostrajania parametrów

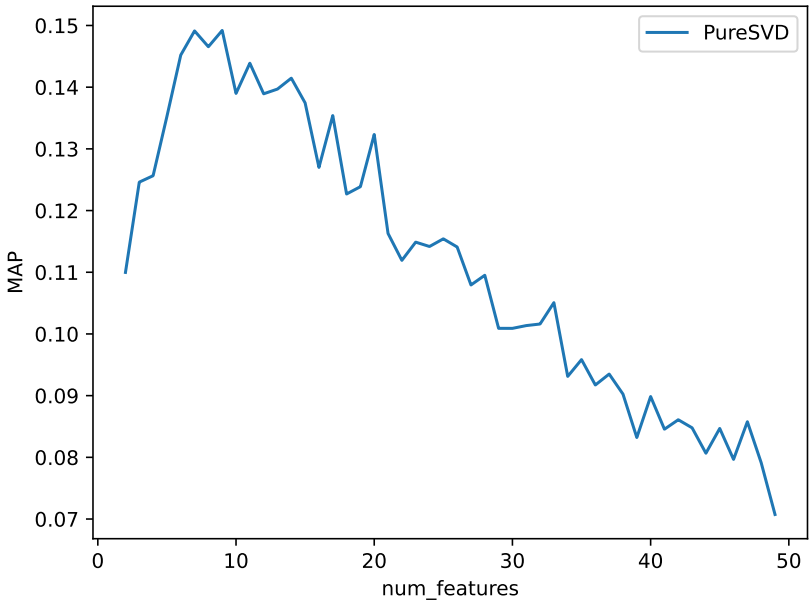
Algorytm	Parametr	Zakres wartości	Typ danych	Najlepsza wartość
UserUser	nnbrs	[2-100]	int	95
ItemItem				70
PureSVD	num factors	[2-50]	int	7
BPR	num factors	[2-50]	int	49
	reg	[0-1]	double	0,02
	neg count	[0-20]	int	4
ImplicitMF	num factors	[2-50]	int	11
	reg	[0-1]	double	0,72
	weight	[0-100]	int	2

W tabeli 8.8 zaprezentowano parametry algorytmów rekomendacyjnych wraz z ich typem, zakresem wartości oraz najlepszą znalezioną wartością. Aby nie było wątpliwości, który z parametrów był dostrajany, wykorzystano angielskie nazwy parametrów z biblioteki *Lenskit*.

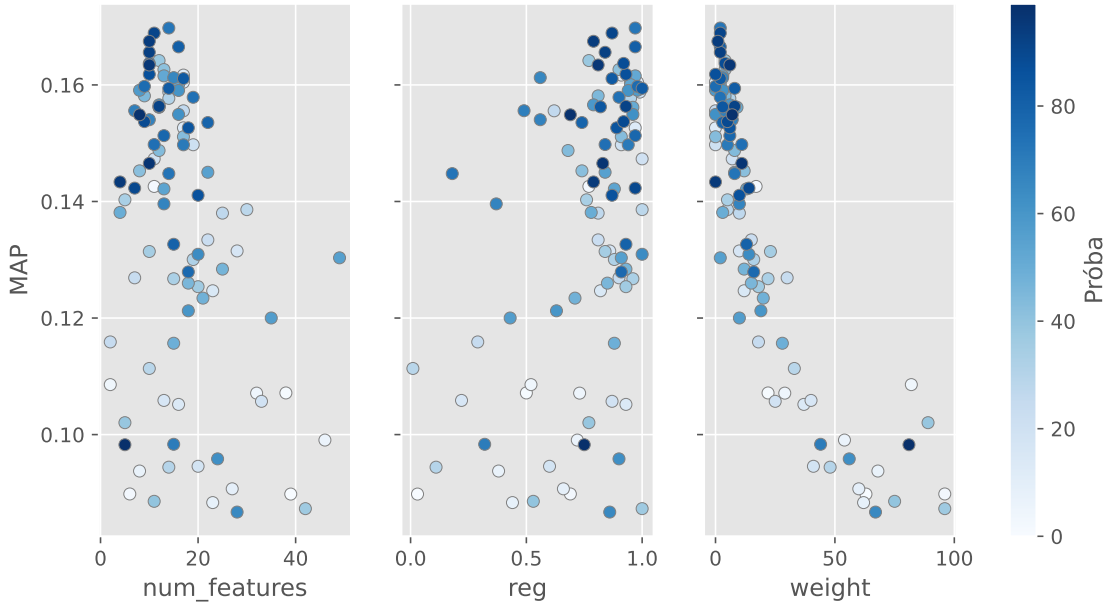
Wyniki procesu dostrajania parametrów algorytmów rekomendacyjnych, zostały zaprezentowane poniżej w formie wykresów. Na wykresie 8.5 przedstawiono dostrajanie parametru *nnbrs* dla algorytmów *UserUser* oraz *ItemItem*. Można zauważyć, że generalnie wraz ze wzrostem wartości tego parametru, zwiększa się jakość generowanych rekomendacji, choć nie jest ona zbyt stabilna i nieznacznie zmienia się w zależności od konkretnej wartości. Na kolejnym wykresie 8.6 można zauważyć, jak zmiana liczby cech ukrytych dla algorytmu *PureSVD*, wpływa na jakość generowanych rekomendacji. Algorytm ten generuje rekomendację wyższej jakości dla stosunkowo niewielkiej liczby cech ukrytych (od 6 do 11). Jednak wraz ze wzrostem liczby tych cech, jakość generowanych rekomendacji systematycznie spada. Na wykresach 8.7, 8.8 oraz 8.9, 8.10 przedstawiono proces dostrajania parametrów dla algorytmów *ImplicitMF* i *BPR*. Na przykładzie tych algorytmów można zauważyć, że choć obydwa algorytmy posiadają takie same parametry (*num_features* oraz *reg*), to optymalne wartości tych parametrów dla każdego z nich mogą być różne. Przykładowo algorytm *ImplicitMF* generuje rekomendację wyższej jakości przy stosunkowo niewielkiej liczbie cech (wymiarów), natomiast algorytm *BPR* do generowania rekomendacji wyższej jakości, potrzebuje większej liczby cech.



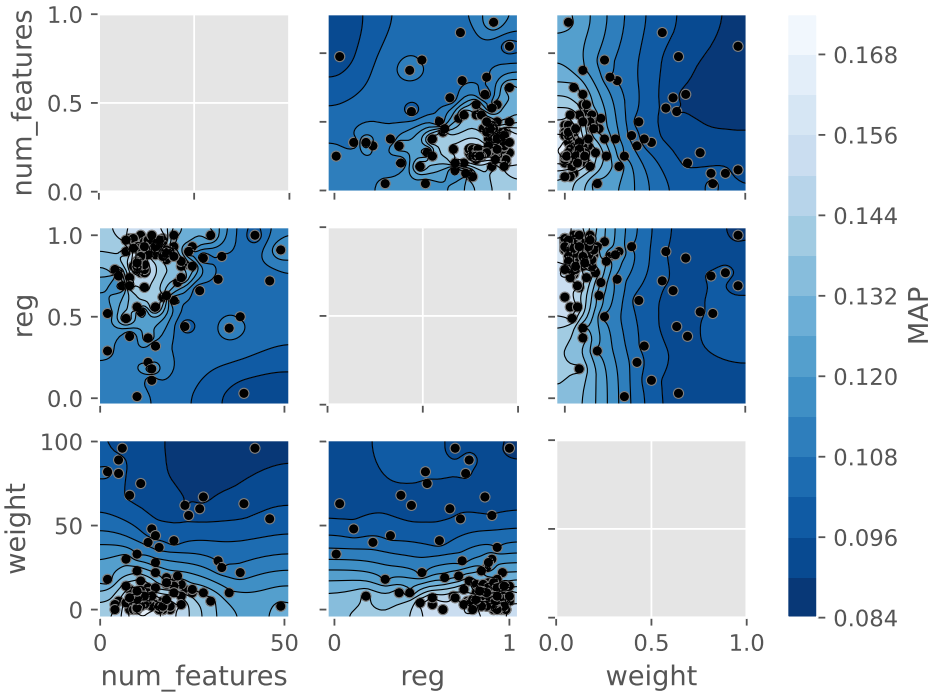
Rysunek 8.5: Dostrajanie parametru *nnbrs* dla algorytmów *UserUser* oraz *ItemItem*



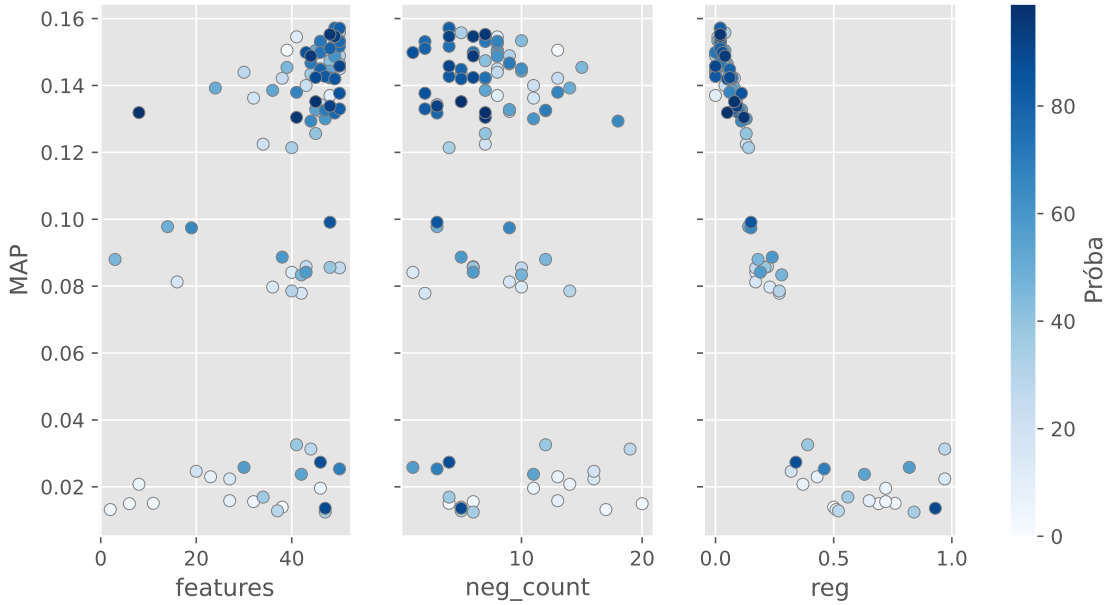
Rysunek 8.6: Dostrajanie parametru *num_features* dla algorytmu *PureSVD*



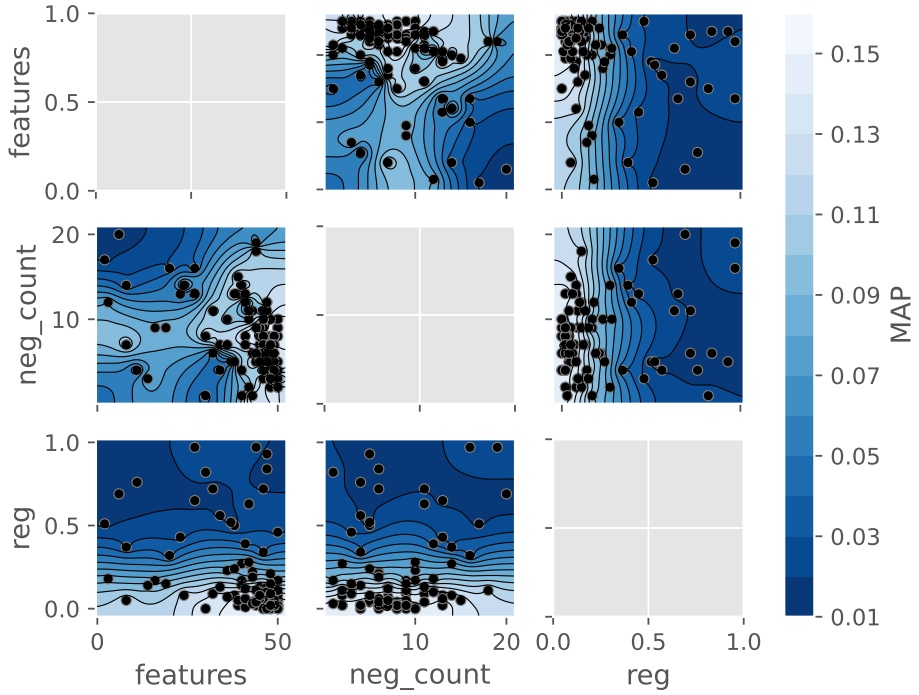
Rysunek 8.7: Dostrajanie parametrów *num_features*, *reg* oraz *weight* dla algorytmu *ImplicitMF*



Rysunek 8.8: Dostrajanie parametrów *num_features*, *reg* oraz *weight* dla algorytmu *ImplicitMF*



Rysunek 8.9: Dostrajanie parametrów *num_features*, *neg_count* oraz *reg* dla algorytmu *BPR*



Rysunek 8.10: Dostrajanie parametrów *num_features*, *neg_count* oraz *reg* dla algorytmu *BPR*

8.4 Metodyka przeprowadzonych eksperymentów

Przeprowadzając badania należy pamiętać o tym, że mamy tutaj do czynienia z systemem rekomendacyjnym, który wykorzystuje algorytm metaheurystyczny, w celu poprawienia jakości generowanych rekomendacji. Z tego względu przeprowadzając eksperymenty należy przemyśleć ich realizację z dwóch perspektyw, które zostaną opisane w dwóch kolejnych podrozdziałach.

8.4.1 Algorytm metaheurystyczny

Do wyszukiwania wektora preferencji algorytm *EAR* wykorzystuje algorytm DE. Ze względu na niedeterministyczny charakter algorytmów ewolucyjnych, niezbędne jest wykonanie wielu niezależnych uruchomień algorytmu DE [13, s. 112]. Z tego względu dla każdego z użytkowników, algorytm DE został uruchomiony 30 razy, a wyniki zostały uśrednione. Domyślne parametry algorytmu DE zostały zaprezentowane w tabeli 8.9. Domyślne parametry oznaczają, że jeżeli w opisie eksperymentu nie wskazano inaczej, to wykorzystywane są parametry domyślne.

Tabela 8.9: Domyślne parametry algorytmu DE na podstawie publikacji [21]

Nazwa parametru	Wartość
Populacja	50
Liczba iteracji	500
Prawdopodobieństwo krzyżowania (CR)	0,9
Współczynnik wzmocnienia (F)	0,5
Mutacja	DE/rand/1

8.4.2 System rekomendacyjny

Ze względu na to, że proces generowania rekomendacji jest złożony i wieloaspektowy, to w pierwszej kolejności podsumowane zostaną informacje, które są niezbędne do przeprowadzenia eksperymentów. Zbiór danych wykorzystany do przeprowadzenia eksperymentów został zaprezentowany w podrozdziale 8.1, a implementacja środowiska badawczego została opisana w podrozdziale 8.2. Dodatkowo przed przystąpieniem do generowania rekomendacji, parametry algorytmów rekomendacyjnych muszą zostać odpowiednio dostrojone, co zostało opisane w podrozdziale 8.3.

Aby przygotować zbiór danych do przeprowadzenia eksperymentów w pierwszej kolejności oceny użytkowników zostały posortowane według znacznika czasu. Takie podejście jest uzasadnione, ponieważ naszym zadaniem jest przewidzenie przyszłych wyborów użytkowników na podstawie ich wcześniejszej aktywności. Dodatkowo na zbiorze danych

MovieLens 100k został zastosowany specjalny filtr, który usunął wszystkie filmy ocenione mniej niż 50 razy, a którego efekt działania został zaprezentowany w tabeli 8.5.

Następnie zbiór ocenionych filmów dla każdego z użytkowników wykorzystanych w eksperymentach, został podzielony na trzy podzbiory:

- Treningowy (60%) – w tym zbiorze znajdują się oceny aktywnego użytkownika, które wykorzystane zostaną do poszukiwania wektora preferencji przez algorytm *EAR*. Na podstawie tego zbioru przeprowadzono również proces dostrajania parametrów algorytmów rekomendacyjnych w podrozdziale 8.3.
- Walidacyjny (20%) – w tym zbiorze znajdują się oceny aktywnego użytkownika, które zostaną wykorzystane do poszukiwania wektora preferencji przez algorytm *EAR*. Jednak zbiór ten został również wykorzystany, podczas dostrajania parametrów algorytmów rekomendacyjnych.
- Testowy (20%) – ten zbiór ocen aktywnego użytkownika wykorzystany zostanie do sprawdzenia jakości zaproponowanych rekomendacji. Na podstawie tego zbioru obliczane są miary jakości, które będą prezentowane w wynikach eksperymentów.

W celu sprawdzenia efektywności zaproponowanego algorytmu, zastosowano protokół ewaluacyjny *wszystkich ocen* opisany w [155]. Oznacza to, że w celu sprawdzenia jakości wygenerowanych rekomendacji, algorytm rekomendacyjny musiał przewidzieć $TopN$ filmów, które znajdują się w systemie rekomendacyjnym, a których użytkownik jeszcze nie ocenił, ponieważ rekomendowane są tylko nieocenione przez użytkownika filmy.

Tak, jak zaznaczyli to autorzy w [155, s. 7], takie podejście jest zbliżone do rzeczywistego zastosowania systemów rekomendacyjnych i należy oczekiwać, że uzyskiwane wyniki wyrażone przy pomocy miar jakości takich jak MAP i NDCG, będą niższe, w stosunku do wyników otrzymywanych przy pomocy innych protokołów ewaluacyjnych. Przykładowo, jeśli w systemie rekomendacyjnym znajdują się 603 filmy, a użytkownik ocenił 100 z nich, to przy założeniu, że $TopN = 10$, zadaniem systemu rekomendacyjnego jest zasugerowanie użytkownikowi 10 filmów z 503 pozostałych. Do określenia jakości wygenerowanych rekomendacji dla danego użytkownika, wykorzystane zostały specjalne miary opisane w podrozdziale 3.2. Miary te porównują $TopN$ rekomendacji zasugerowanych przez algorytm rekomendacyjny z filmami, które dany użytkownik posiada w swoim zbiorze testowym. Ze względu na czasochłonny proces uczenia się funkcji rangującej przez algorytm *EAR*, w części przeprowadzonych eksperymentów wybrano losowo pewien podzbiór użytkowników ze zbioru U dla których wygenerowano rekomendacje, a następnie wyniki uśredniono.

Dodatkowo w celu wykazania istotności statystycznej zaprezentowanych wyników, w niektórych eksperymentach przeprowadzono również test statystyczny Wilcoxona [179]. Wybór tego testu jest zgodny z sugestiami dotyczącymi procesu ewaluacji algorytmów rekomendacyjnych, które można znaleźć w literaturze [153, s. 269].

Rozdział 9

Badania eksperymentalne

Ten rozdział poświęcony zostanie badaniom eksperymentalnym. W pierwszej kolejności podsumowane zostaną najważniejsze informacje, związane z wykorzystanym środowiskiem badawczym i omówiony zostanie ogólny plan eksperymentów. Następnie zaprezentowane zostaną szczegółowe wyniki badań, wraz ze stosownym komentarzem.

9.1 Informacje ogólne

Eksperymenty zostały przeprowadzone na zbiorze danych *MovieLens 100k*, który został opisany w podrozdziale 8.1. Rekomendacje generowane były na podstawie sześciu algorytmów rekomendacyjnych, które przedstawiono w tabeli 8.7. Każdy z algorytmów rekomendacyjnych generował rekomendacje w postaci dziesięciu przedmiotów, które posortowano zgodnie z poziomem istotności. Parametry algorytmów rekomendacyjnych zostały dostrojone zgodnie z procesem dostrajania parametrów, który został opisany w 8.3.

Ze względu na to, że zaproponowany algorytm bazuje na idei agregacji, zostanie on porównany z pięcioma innymi metodami agregującymi, które omówiono w podrozdziale 5.2. Do ewaluacji zaproponowanego podejścia wykorzystane zostaną specjalne miary, które służą do określenia jakości rankingu. Miary te porównują rekomendacje, które zostały wygenerowane przez poszczególne algorytmy rekomendacyjne z przedmiotami, które znajdują się w zbiorze testowym danego użytkownika. Zostały one szczegółowo opisane w podrozdziale 3.2 i będą obliczane dla następujących wartości odcięcia @ (ang. cut-off treshold): *AP@10*, *NDCG@10*, *NDCG@5*, *Precision@1* oraz *Precision@10*.

Eksperymenty zostały przeprowadzone zgodnie z przedstawionym w podrozdziale 8.4 protokołem. Plan eksperymentów wygląda następująco:

1. Badania podstawowe – w pierwszej kolejności w podrozdziale 9.2 przedstawione zostaną badania podstawowe, których celem jest zaprezentowanie przykładowych wyników działania algorytmów rekomendacyjnych oraz metod agregujących na rzeczywistym zbiorze danych *MovieLens 100k*. Do ich realizacji wybrano losowo 25 użytkowników.

2. Badania eksperymentalne zaproponowanego algorytmu – w podrozdziale 9.3 przedstawione zostaną wyniki badań eksperymentalnych zaproponowanego algorytmu. W pierwszej kolejności przebadane zostaną różne warianty funkcji oceny. Następnie przeprowadzona zostanie analiza populacji algorytmu DE oraz analiza wybranych osobników. Dodatkowo w celu sprawdzenia, czy zaproponowany algorytm potrafi wykryć rankingi niskiej jakości, przeprowadzone zostaną badania uwzględniające w procesie agregacji rankingi, które zostały wygenerowane w sposób losowy. Ponadto w tym podrozdziale przeprowadzone zostaną badania związane z modyfikacją zaproponowanego algorytmu, która została opisana w podrozdziale 7.4.

3. Badania końcowe – na końcu tego rozdziału (podrozdział 9.4), przedstawione zostaną wyniki badań, które zostały przeprowadzone na wszystkich użytkownikach dostępnych w zbiorze U. Ma to na celu ostateczne potwierdzenie skuteczności zaproponowanego algorytmu. Dodatkowo w tym podrozdziale przeprowadzony zostanie test statystyczny Wilcoxona [179], którego celem będzie wykazanie istotności statystycznej prezentowanych wyników.

9.2 Badania podstawowe

Wynik eksperymentów przedstawione w podrozdziałach 9.3 oraz 9.4 będą zazwyczaj uśrednione dla większej grupy użytkowników. Aby jednak lepiej zrozumieć powody dla których zastosowanie metod agregujących w systemach rekomendacyjnych jest uzasadnione, warto w pierwszej kolejności zapoznać się z przykładowymi wynikami działania tych algorytmów na mniejszej grupie użytkowników. Zostanie to zaprezentowane w podrozdziałach 9.2.1 oraz 9.2.2.

9.2.1 Przykładowe rezultaty

W tym podrozdziale przedstawione zostaną przykładowe wyniki działania algorytmów rekomendacyjnych oraz metod agregujących, które zostały przeprowadzone na niewielkiej grupie użytkowników. Ma to na celu zaprezentowanie powodów dla których zastosowanie metod agregujących w systemach rekomendacyjnych jest uzasadnione. W tym celu z całego zbioru użytkowników wybrano losowo 25 użytkowników i następnie każdy z algorytmów rekomendacyjnych wygenerował dla nich rekomendacje, których jakość została obliczona na podstawie miary $AP@10$.

Analizując wyniki zaprezentowane w tabeli 9.1 można zauważyć, że algorytmy rekomendacyjne generowały rekomendacje o zróżnicowanej jakości i żaden z nich nie był zdecydowanie lepszy od innych. Przykładowo algorytm *BPR* wygenerował najlepsze rekomendacje dla sześciu użytkowników, algorytm *ImplicitMF* dla czterech użytkowników,

Tabela 9.1: Jakość rekomendacji, wygenerowana przez poszczególne algorytmy rekomendacyjne i wyrażona przy pomocy miary *AP@10* (dla 25 losowo wybranych użytkowników). Pogrubiono najlepszy wynik dla każdego z użytkowników

Id	BPR	ImplicitMF	ItemItem	SVD	UserUser	MostPopular
1	0,23	**0,56**	0,23	0,36	0,47	0,29
2	**0,33**	0,25	0,22	0,04	0,08	0,04
3	0,28	0,21	**0,37**	0,21	0,30	0,03
4	0,05	0,03	0,03	0,02	0,13	0,01
5	**0,17**	0,14	0,08	0,03	0,04	0,06
6	0,00	0,02	**0,10**	0,01	0,05	0,01
7	0,05	0,15	**0,16**	0,15	0,11	0,00
8	0,05	0,19	**0,69**	0,20	0,60	0,49
9	0,00	0,01	**0,02**	0,01	0,01	0,00
10	0,33	0,51	0,43	0,56	**0,66**	0,05
11	0,00	0,00	0,00	0,00	0,00	0,00
12	**0,67**	0,15	0,01	0,16	0,05	0,00
13	0,38	0,11	0,03	**0,45**	0,08	0,10
14	0,00	0,24	0,28	**0,30**	0,30	0,20
15	0,10	0,71	0,53	0,58	**0,76**	0,38
16	0,10	0,01	0,16	**0,27**	0,22	0,01
17	0,00	0,01	0,02	0,03	0,01	0,02
18	**0,15**	0,03	0,00	0,08	0,00	0,10
19	0,01	0,10	0,12	**0,20**	0,15	0,05
20	**0,05**	0,01	0,01	0,04	0,03	0,02
21	0,00	0,00	0,00	0,00	0,00	0,00
22	**0,15**	0,03	0,00	0,07	0,01	0,00
23	0,30	0,32	0,31	0,24	**0,39**	0,31
24	0,23	0,08	**0,32**	0,19	0,24	0,00
25	0,08	0,00	**0,13**	0,05	0,03	0,01
średnia	0,15	0,15	0,17	0,17	0,19	0,09

algorytm *ItemItem* dla sześciu użytkowników i algorytm *UserUser* dla czterech użytkowników. Świadczy to o tym, że nie istnieje tutaj jeden, uniwersalny algorytm, który generuje rekomendacje wysokiej jakości we wszystkich przypadkach. Choć po uśrednieniu algorytmy te generują rekomendacje o zbliżonej jakości, to analizując ich efektywność w kontekście poszczególnych użytkowników można zauważyć, że jest ona mocno zróżnicowana.

Dla niektórych użytkowników algorytmy rekomendacyjne generowały rekomendacje, które okazały się kompletnie chybione (np. użytkownicy nr 11 i 21), a różnica w jakości generowanych rekomendacji, pomiędzy poszczególnymi algorytmami, była dosyć znaczna. Przykładowo dla użytkownika nr 12, algorytm *BPR* wygenerował rekomendacje, które okazały się dużo lepsze od rekomendacji, które zostały zarekomendowane przez inne algorytmy rekomendacyjne. Natomiast dla użytkownika nr 3 algorytmem, który generował rekomendacje wyraźnie wyższej jakości, był algorytm *ItemItem*.

Wyniki działania algorytmów agregujących zaprezentowane zostały w tabeli 9.2. Analizując przedstawione wyniki można zauważyć, że różne algorytmy agregujące poprawiły jakość rekomendacji w różnym stopniu i żaden z tych algorytmów nie uzyskał najlepszego wyniku w przypadku wszystkich 25 użytkowników. Warto przy tym zwrócić uwagę na uśrednioną wartość jakości rekomendacji. Algorytmy *CombMax*, *CombMed* oraz *CombMin* miały identyczną, średnią wartość jakości rekomendacji, która wyniosła 0,16. Natomiast algorytmy *BordaFuse* oraz *CombSum*, uzyskały średnią wartość jakości rekomendacji, wynoszącą 0,18. Choć po uśrednieniu niektóre z metod agregujących okazały się nieznacznie lepsze od innych, to analizując jakość rekomendacji w kontekście konkretnych użytkowników można zaobserwować, że występuje tutaj podobny problem, który został już wcześniej omówiony w przypadku algorytmów rekomendacyjnych. Nie istnieje tutaj uniwersalna metoda agregująca, dzięki której uzyskamy dobre wyniki we wszystkich przypadkach (dla każdego z użytkowników).

Należy również zauważyć, że w niektórych przypadkach algorytmy agregujące uzyskały jakość rekomendacji równą 0 (przykładowo dla użytkownika nr 11). Taki wynik nie jest w przypadku tego użytkownika zaskoczeniem, ponieważ świadczy to o tym, że jeżeli algorytmy rekomendacyjne wygenerowały rekomendacje, których jakość jest równa 0, to utworzona na ich podstawie agregacja również będzie takiej jakości. Można jednak zaobserwować, że w przypadku niektórych użytkowników utworzona agregacja jest wyjątkowo dobra. Przykładowo analizując agregację utworzoną dla użytkownika nr 8 (tabela 9.2), można zauważyć, że algorytm agregujący *CombMax* uzyskał wyniki dużo lepsze, w stosunku do innych metod agregujących.

Jedną z ciekawszych obserwacji, jaką można zauważyć analizując wyniki przedstawione w tabeli 9.2 jest to, że dla niektórych użytkowników (np. 1 i 23), jakość rekomendacji, która została wygenerowana przez algorytmy agregujące była lepsza od wszystkich algorytmów rekomendacyjnych. Świadczy to o tym, że algorytmy agregujące w niektórych przypadkach mogą poprawić ogólną jakość rekomendacji.

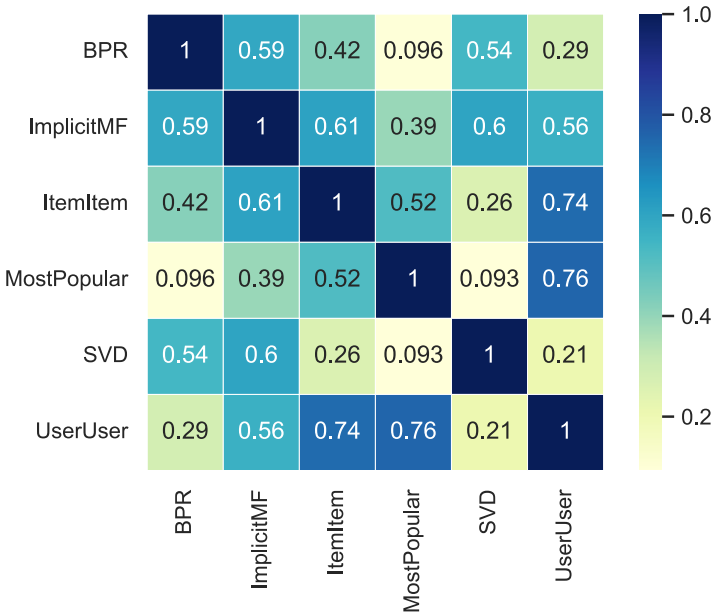
Tabela 9.2: Jakość rekomendacji, wygenerowana przez poszczególne metody agregujące i wyrażona przy pomocy miary *AP@10* (dla 25 losowych użytkowników)

Id	BordaFuse	CombMax	CombMed	CombMin	CombSum
1	0,57	0,51	**0,59**	0,41	0,58
2	0,16	**0,17**	0,10	0,11	0,10
3	**0,30**	0,23	0,23	0,28	0,29
4	0,07	0,02	0,03	0,03	**0,06**
5	0,10	0,03	0,08	0,05	**0,10**
6	0,00	**0,05**	0,01	0,01	0,00
7	0,09	**0,23**	0,12	0,08	0,17
8	0,19	**0,60**	0,22	0,22	0,29
9	0,01	0,00	0,01	**0,02**	0,01
10	**0,61**	0,38	0,58	0,53	0,60
11	0,00	0,00	0,00	0,00	0,00
12	**0,23**	0,01	0,09	0,15	0,12
13	0,27	0,05	0,09	**0,44**	0,15
14	0,17	0,25	0,25	**0,28**	0,28
15	0,57	0,52	**0,73**	0,53	0,68
16	0,24	0,07	0,20	0,18	**0,26**
17	0,00	0,01	**0,01**	0,00	0,01
18	0,01	0,00	0,00	**0,02**	0,00
19	0,14	0,07	0,14	0,16	**0,24**
20	**0,13**	0,02	0,05	0,03	0,05
21	0,00	0,00	0,00	0,00	0,00
22	0,01	0,00	**0,03**	0,00	0,02
23	0,31	**0,53**	0,34	0,26	0,37
24	**0,27**	0,23	0,20	0,18	0,23
25	0,00	**0,02**	0,00	0,01	0,00
średnia	0,18	0,16	0,16	0,16	0,18

9.2.2 Analiza generowanych rekomendacji (rankingów)

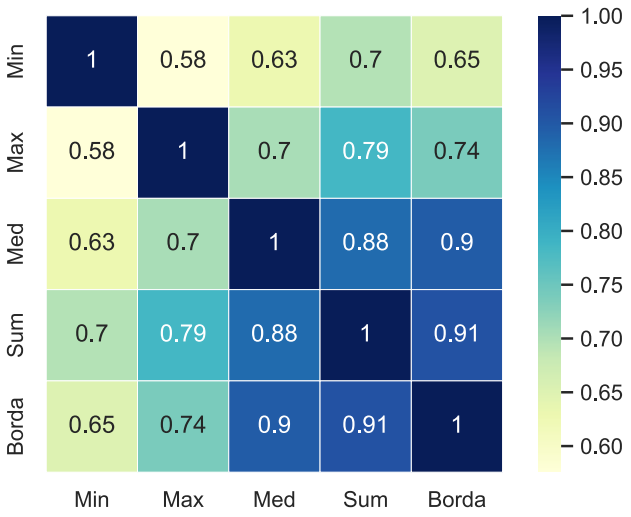
W niniejszym podrozdziale zaprezentowana zostanie analiza generowanych rekomendacji (rankingów). W tym celu obliczona zostanie korelacja pomiędzy rankingami utworzonymi przez poszczególne algorytmy rekomendacyjne i metody agregujące. Wyniki tych obliczeń zostaną zaprezentowane w postaci macierzy korelacji. Celem tej macierzy jest przedstawienie, jakie występuje podobieństwo pomiędzy rekomendacjami (rankingami), które zostały wygenerowane przez poszczególne algorytmy rekomendacyjne i metody agregujące. Macierz ta została obliczona tylko dla jednego użytkownika, dlatego na jej podstawie nie można wyciągnąć ogólnych wniosków, dotyczących działania całego systemu rekomendacyjnego. Wykorzystanie jej w tym podrozdziale ma na celu tylko przedstawienie przykładowego zastosowania tego typu macierzy, w analizie działania systemów rekomendacyjnych.

Analizując macierz korelacji zaprezentowaną na rysunku 9.1 można zauważyć, że stopień korelacji pomiędzy generowanymi rekomendacjami jest zróżnicowany. Przykładowo rekomendacje wygenerowane przez algorytmy bazujące na sąsiedztwie (*UserUser* oraz *ItemItem*), mają stosunkowo wysoki współczynnik korelacji, wynoszący 0,74. Najmniejsza korelacja występuje pomiędzy algorytmami *MostPopular*, *SVD* oraz *BPR*. Można jednak zaobserwować, że korelacja pomiędzy algorytmami bazującymi na faktoryzacji (*SVD* i *BPR*), a algorytmami bazującymi na sąsiedztwie (*UserUser* oraz *ItemItem*), również nie była zbyt wysoka. Świadczy to o tym, że stopień korelacji pomiędzy generowanymi rankingami, może zależeć od typu wykorzystywanego algorytmu.



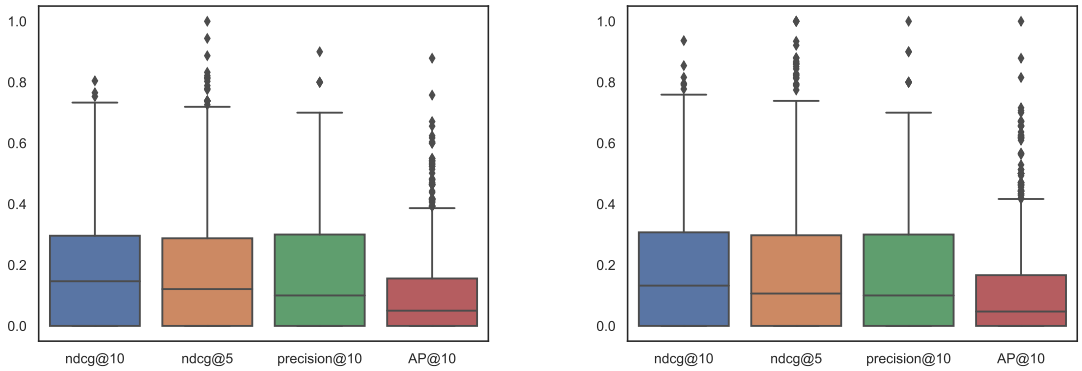
Rysunek 9.1: Macierz korelacji algorytmów rekomendacyjnych (dla jednego użytkownika), gdzie korelacja została obliczona przy pomocy korelacji Tau Kendalla

Na rysunku 9.2 przedstawiono macierz korelacji dla rankingów utworzonych przez metody agregujące. Rekomendacje wygenerowane przez te metody są ze sobą dużo bardziej skorelowane, w porównaniu do stopnia korelacji algorytmów rekomendacyjnych (rysunek 9.1). Metodami, które generują rankingi najbardziej zbliżone do siebie są metody: *BordaFuse*, *CombSum* oraz *CombMed*. Metoda *CombMin* generuje rekomendacje, które są najmniej skorelowane z rekomendacjami utworzonymi przez inne metody.

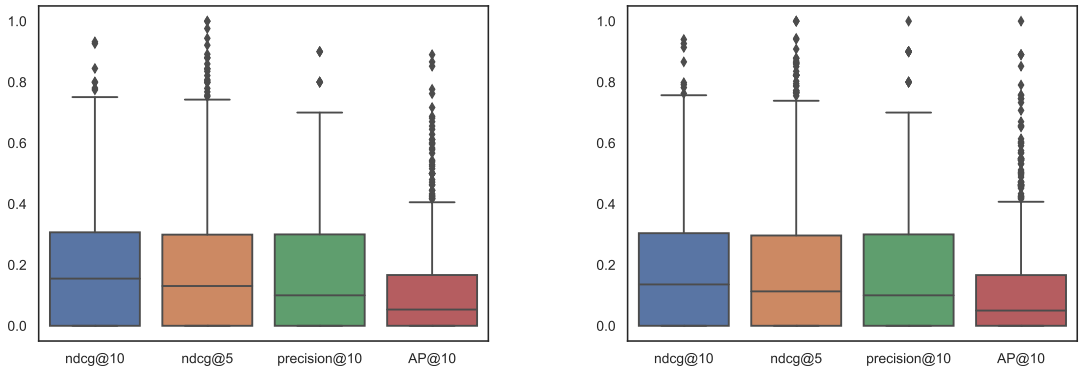


Rysunek 9.2: Macierz korelacji metod agregujących (dla jednego użytkownika), gdzie korelacja została obliczona przy pomocy korelacji Tau Kendalla

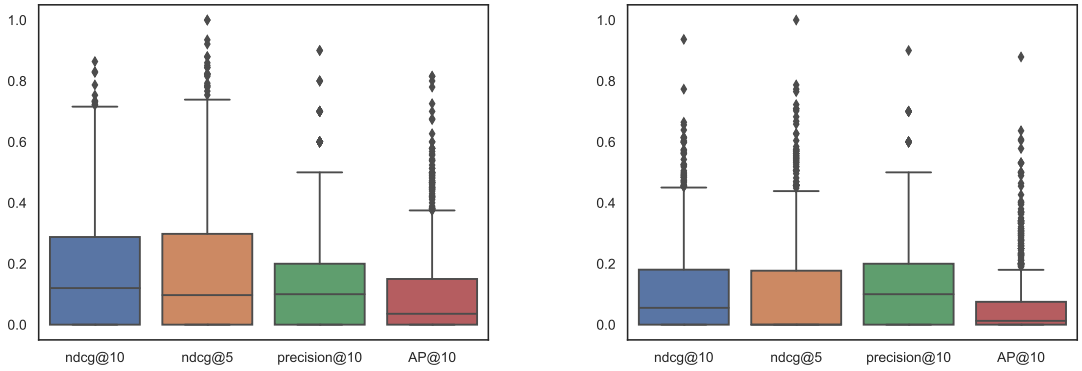
Dodatkowo w celu zaprezentowania rozkładu wartości jakości generowanych rekomendacji, na rysunkach 9.3, 9.4 oraz 9.5 zostały zaprezentowane wykresy pudełkowe, które przedstawiają jakość generowanych rekomendacji w zależności od wybranych miar jakości, które to rekomendacje zostały wygenerowane przez poszczególne algorytmy rekomendacyjne. Analizując te wykresy można zauważyć, że praktycznie we wszystkich przypadkach mediana jakości generowanych rekomendacji była stosunkowo niska. Należy jednak zwrócić uwagę, że dla każdej miary występuje dosyć duża liczba odstańców, co oznacza, że w systemie występują użytkownicy, dla których generowane rekomendacje są wyjątkowo dobrej jakości w porównaniu do reszty użytkowników.



Rysunek 9.3: Wykresy pudełkowe, przedstawiające rozkład wartości jakości rekomendacji, wyrażonej przy pomocy miar: *NDCG@10*, *NDCG@5*, *Precision@10*, *AP@10*, które zostały wygenerowane przez algorytmy: *BPR* (lewo) oraz *ImplicitMF* (prawo)



Rysunek 9.4: Wykresy pudełkowe, przedstawiające rozkład wartości jakości rekomendacji, wyrażonej przy pomocy miar: *NDCG@10*, *NDCG@5*, *Precision@10*, *AP@10*, które zostały wygenerowane przez algorytmy: *ItemItem* (lewo) oraz *UserUser* (prawo)



Rysunek 9.5: Wykresy pudełkowe, przedstawiające rozkład wartości jakości rekomendacji, wyrażonej przy pomocy miar: *NDCG@10*, *NDCG@5*, *Precision@10*, *AP@10*, które zostały wygenerowane przez algorytmy: *SVD* (lewo) oraz *MostPopular* (prawo)

9.3 Badania eksperymentalne zaproponowanego algorytmu

W tym podrozdziale zaprezentowane zostaną badania eksperymentalne zaproponowanego algorytmu *EAR*, który został omówiony w rozdziale 7. Badania te będą skoncentrowane głównie na eksperymentach z zastosowaniem różnych wariantów funkcji oceny i przeanalizowaniem ich wpływu na jakość tworzonej agregacji (rekomendacji).

9.3.1 Sprawdzenie różnych wariantów funkcji oceny

W podrozdziale 7.3 opisane zostały różne warianty funkcji oceny, które mogą zostać wykorzystane przez algorytm *EAR* do poszukiwania wektora preferencji w_{u_A}. Aby przetestować ich wpływ na jakość tworzonej agregacji, w tym podrozdziale przeprowadzone zostaną badania na następujących wariantach funkcji oceny:

- $EAR_{\mathrm{DE_{F1}}} = AP@K(w_{u_A}, RT, ST)$ – podczas obliczania tego wariantu funkcji oceny, porównywane są ze sobą przedmioty, które znajdują się w zbiorze treningowym aktywnego użytkownika u_A z przedmiotami, które zostały zarekomendowane przez system.
- $EAR_{\mathrm{DE_{F2}}} = AP@K(w_{u_A}, RT, ST) + AP@K(w_{u_A}, RT, SV)$ – w tym wariancie funkcji oceny, ze zbioru treningowego wydzielony został zbiór walidacyjny, co umożliwia optymalizację wektora w_{u_A}, w stosunku do tych dwóch zbiorów jednocześnie. Takie podejście umożliwia lepszą generalizację wektora w_{u_A}, a tym samym zapobiega zbytniemu dopasowaniu wektora preferencji do zbioru treningowego.
- $EAR_{\mathrm{DE_{F3}}} = AP@K(w_{u_A}, RT, ST) + \lambda\ AP@K(w_{u_A}, RT, SV)$ – ten wariant funkcji oceny, podobny jest do wariantu $\mathrm{EAR_{DE_{F2}}}$, ale wprowadza się tutaj dodatkowo parametr λ, który umożliwia sterowanie wpływem miary AP obliczonej na zbiorze walidacyjnym, na ogólną wartość funkcji oceny.

Analizując wyniki zaprezentowane w tabeli 9.3 można zauważyć, że algorytmem rekomendacyjnym, który generował rekomendacje najwyższej jakości był algorytm *UserUser*. Algorytmem, który generował rekomendacje najniższej jakości, był niespersonalizowany algorytm *MostPopular*. Generalnie wszystkie spersonalizowane algorytmy rekomendacyjne, generowały rekomendacje o zbliżonej jakości.

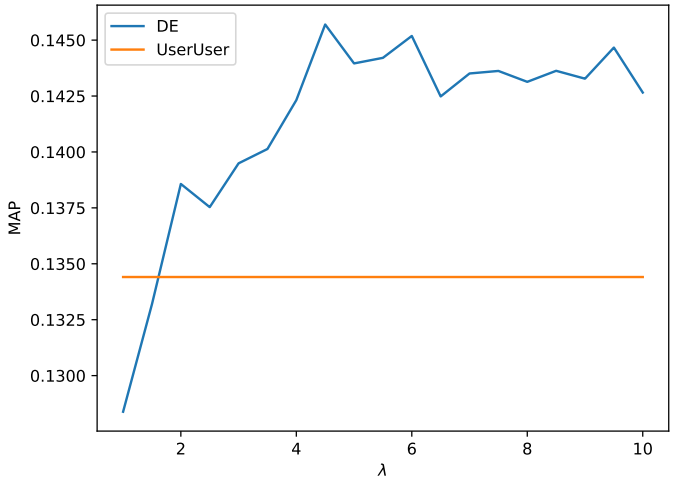
Natomiast analizując wyniki uzyskane przez metody agregujące można zauważyć, że agregacja wysokiej jakości została utworzona przy wykorzystaniu metod: *CombMed*, *BordaFuse* oraz *CombSum*. Metodami, które utworzyły agregację wyraźnie gorszą, były metody: *CombMax* oraz *CombMin*. Warto zaznaczyć, że metoda agregująca *CombSum* uzyskała najlepsze wyniki dla miary *Precision@1*, spośród wszystkich metod wykorzystanych w badaniach. Oznacza to, że najczęściej rekomendowała ona relewantne przedmioty, które znajdowały się na pierwszej pozycji w rekomendowanej liście.

Tabela 9.3: Wyniki eksperymentów przeprowadzonych na 300 losowo wybranych użytkownikach dla różnych wariantów funkcji oceny

Algorytm	**AP@10**	**NDCG@10**	**NDCG@5**	**Precision@1**	**Precision@10**
Algorytmy rekomendacyjne					
BPR	0,1209	0,2027	0,2090	0,2867	0,2157
ImplicitMF	0,1272	0,2114	0,2129	0,2533	0,2263
ItemItem	0,1170	0,1964	0,2092	0,2933	0,2020
SVD	0,1239	0,2112	0,2246	0,3033	0,2153
UserUser	0,1344	0,2177	0,2349	0,3233	0,2173
MostPopular	0,0715	0,1307	0,1269	0,1733	0,1490
Metody agregujące					
BordaFuse	0,1370	0,2276	0,2361	0,3133	0,2343
CombMax	0,1159	0,1948	0,2056	0,2867	0,1987
CombMed	0,1380	0,2244	0,2358	0,2967	0,2307
CombMin	0,1133	0,2003	0,2152	0,2700	0,2050
CombSum	0,1393	0,2279	0,2376	**0,3367**	0,2340
Zaproponowany algorytm					
$EAR_{DE_{F1}}$	0,1258	0,2119	0,2221	0,2967	0,2230
$EAR_{DE_{F2}}$	0,1284	0,2135	0,2193	0,2967	0,2273
$EAR_{DE_{F3}}$	**0,1414**	**0,2296**	**0,2396**	0,3233	**0,2427**

W tabeli 9.3 zaprezentowane zostały również wyniki dla różnych wariantów funkcji oceny, które zostały wykorzystane w algorytmie *EAR*. Warianty $EAR_{DE_{F1}}$ oraz $EAR_{DE_{F2}}$ uzyskały wyniki bardzo zbliżone do siebie, biorąc pod uwagę wszystkie miary jakości. Natomiast porównując je w stosunku do wyników uzyskanych przez inne algorytmy rekomendacyjne można zauważyć, że warianty te, były lepsze od większości z nich i tylko algorytm *UserUser* wygenerował rekomendacje wyższej jakości. Ponadto porównując warianty $EAR_{DE_{F1}}$ i $EAR_{DE_{F2}}$ z metodami agregującymi, to rekomendację wyższej jakości wygenerowały metody: *BordaFuse*, *CombMed* oraz *CombSum*. Na szczególną uwagę zasługuje wariant $EAR_{DE_{F3}}$ z parametrem $\lambda = 5$, który uzyskał najlepsze wyniki praktycznie w odniesieniu do wszystkich miar jakości. Aby ostatecznie potwierdzić skuteczność tej wersji funkcji oceny, w podrozdziale 9.4 przeprowadzone zostaną eksperymenty na pełnym zbiorze użytkowników.

Dodatkowo na wykresie 9.6 zaprezentowane zostały wyniki, przedstawiające wpływ różnych wartości parametru λ na jakość generowanych rekomendacji. Parametr ten, wykorzystywany jest w funkcji oceny $\text{EAR}_{\text{DE}_{\text{F3}}}$ i steruje on wpływem miary AP, która obliczana jest na zbiorze walidacyjnym na ogólną wartość funkcji oceny. Generalnie można zauważyć, że wraz ze wzrostem wartości parametru λ, rośnie jakość generowanych rekomendacji (wyrażona przy pomocy miary MAP). Jest to ciekawa obserwacja, ponieważ oznacza ona, że zwiększając istotność zbioru walidacyjnego podczas obliczania funkcji oceny, możemy zwiększyć jakość generowanych rekomendacji. Powodem takiej sytuacji może być fakt, że zbiór treningowy i walidacyjny nie jest tak samo liczny. Bez tego parametru z większym prawdopodobieństwem jakość rekomendacji jest optymalizowana pod kątem jakości rekomendacji na zbiorze treningowym, ponieważ znajduje się w nim więcej przedmiotów i wektor preferencji w_{u_A}, z większym prawdopodobieństwem będzie dopasowany do niego.



Rysunek 9.6: Wpływ zmiany parametru λ (wariant funkcji oceny $\text{EAR}_{\text{DE}_{\text{F3}}}$) na jakość rekomendacji, wygenerowanej przez algorytm *EAR*

9.3.2 Analiza populacji algorytmu DE dla wybranych użytkowników

Analiza populacji, która została przedstawiona w tym podrozdziale, będzie zrealizowana tylko dla pewnej grupy wybranych użytkowników (podobnie jak w podrozdziale 9.2.1). Jest to związane między innymi z tym, że dla algorytmu EAR, każdy z użytkowników jest osobnym problemem optymalizacyjnym, ponieważ każdy z nich posiada swój spersonalizowany wektor preferencji. Chociaż można analizować uśrednione wartości biorąc pod uwagę wszystkich użytkowników w systemie, to ciekawszym podejściem wydaje się być analiza działania zaproponowanego algorytmu z perspektywy poszczególnych użytkowników.

Tabela 9.4: Wartości funkcji oceny (wariant $EAR_{DE_{F1}}$, $EAR_{DE_{F2}}$ i $EAR_{DE_{F3}}$) dla 25 losowo wybranych użytkowników

	$EAR_{DE_{F1}}$				$EAR_{DE_{F2}}$				$EAR_{DE_{F3}}$			
Id	**Min**	**Med**	**Avg**	**Max**	**Min**	**Med**	**Avg**	**Max**	**Min**	**Med**	**Avg**	**Max**
1	0,58	0,77	0,76	0,77	0,65	0,84	0,83	0,85	0,86	1,10	1,08	1,10
2	0,42	0,53	0,53	0,54	0,44	0,54	0,54	0,55	0,52	0,70	0,69	0,71
3	0,48	0,74	0,73	0,76	0,50	0,75	0,74	0,76	0,58	0,79	0,78	0,79
4	0,44	0,61	0,60	0,62	0,46	0,62	0,61	0,63	0,50	0,66	0,65	0,67
5	0,17	0,46	0,43	0,47	0,19	0,48	0,45	0,491	0,22	0,56	0,52	0,57
6	0,28	0,470	0,46	0,49	0,35	0,59	0,57	0,61	0,57	0,91	0,89	0,94
7	0,36	0,68	0,64	0,69	0,43	0,75	0,72	0,76	0,65	0,98	0,93	0,98
8	0,23	0,52	0,50	0,54	0,40	0,62	0,60	0,63	0,79	1,07	1,05	1,08
9	0,29	0,60	0,58	0,61	0,41	0,70	0,68	0,71	0,67	1,08	1,04	1,08
10	0,26	0,40	0,39	0,40	0,36	0,62	0,60	0,65	0,65	1,53	1,40	1,54
11	0,31	0,40	0,39	0,41	0,40	0,47	0,46	0,47	0,60	0,67	0,67	0,69
12	0,15	0,44	0,41	0,450	0,28	0,67	0,64	0,67	0,67	1,83	1,78	1,88
13	0,36	0,52	0,51	0,52	0,42	0,64	0,62	0,65	0,61	1,02	0,98	1,06
14	0,14	0,38	0,37	0,39	0,17	0,46	0,43	0,47	0,27	0,78	0,72	0,79
15	0,39	0,47	0,46	0,48	0,41	0,49	0,48	0,49	0,44	0,61	0,57	0,61
16	0,07	0,41	0,39	0,42	0,19	0,55	0,52	0,56	0,53	1,10	1,05	1,18
17	0,32	0,54	0,52	0,55	0,35	0,74	0,71	0,74	0,45	1,30	1,21	1,31
18	0,19	0,49	0,48	0,50	0,21	0,55	0,53	0,56	0,28	0,72	0,70	0,73
19	0,13	0,32	0,31	0,33	0,23	0,46	0,44	0,46	0,52	0,98	0,95	1,01
20	0,17	0,42	0,41	0,43	0,17	0,42	0,42	0,44	0,17	0,55	0,52	0,56
21	0,51	0,62	0,62	0,62	0,53	0,65	0,65	0,66	0,60	0,77	0,76	0,78
22	0,36	0,65	0,64	0,67	0,40	0,82	0,79	0,82	0,53	1,34	1,29	1,35
23	0,41	0,58	0,57	0,59	0,47	0,69	0,67	0,69	0,66	1,05	1,02	1,06
24	0,21	0,32	0,31	0,32	0,25	0,39	0,40	0,44	0,30	0,72	0,70	0,75
25	0,51	0,64	0,63	0,64	0,52	0,65	0,65	0,66	0,54	0,72	0,70	0,74

W tabeli 9.4 zaprezentowano podstawowe statystyki dotyczące funkcji oceny, które zostały uzyskane podczas optymalizacji funkcji rangującej dla poszczególnych użytkowników. Uwzględnione zostały następujące wartości: minimalna (*Min*), mediana (*Med*), średnia (*Avg*) i maksymalna (*Max*). Obliczone one zostały dla losowo wybranych 25 użytkowników i dla 3 wariantów funkcji oceny: $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F1}}}$, $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F2}}}$, $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F3}}}$.

Warto zauważyć, że dla poszczególnych użytkowników, wartość funkcji oceny osiągnęła zróżnicowaną maksymalną wartość (*Max*). Świadczy to o tym, że nie dla każdego użytkownika dostrajanie wektora preferencji przebiegało tak samo i algorytm EAR nie zawsze mógł uzyskać wysoką wartość miary AP. Przykładowo w wariancie $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F1}}}$ dla użytkownika nr 24, maksymalna wartość funkcji oceny (*Max*) wynosiła 0,32, a dla użytkownika nr 1 wyniosła ona 0,77.

Warto również zauważyć, że dla żadnego użytkownika wartość funkcji oceny dla wariantu $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F1}}}$ nie zbliżyła się do wartości 1, czyli do maksymalnej wartości, jaką funkcja ta może uzyskać. Choć w badaniach jest uwzględnionych tylko 25 użytkowników, to autor niniejszej rozprawy przetestował ten wariant funkcji oceny również na większej grupie użytkowników i nigdy jej wartość nie zbliżyła się do jedności.

Prawdopodobnie jest to spowodowane specyficznym ustawieniem parametru K w $AP@K$ (opis w podrozdziale 7.3). Gdy wartość tego parametru wynosi maksymalną liczbę przedmiotów, którą użytkowników posiada w swoim zbiorze treningowym, to jest to bardzo mało prawdopodobne, że algorytm EAR utworzy funkcję rangującą, która uporządkuje wszystkie przedmioty znajdujące się w systemie w takiej kolejności, aby wartość miary $AP@K$ wynosiła 1.

Oczywiście można zmniejszyć wartość tego parametru, ale bazując na wynikach eksperymentów, które autor niniejszej rozprawy przeprowadził w swoich wcześniejszych publikacjach [37], pogarsza to generalizację i powoduje nadmierne dopasowanie wektora preferencji do pewnego podzbioru przedmiotów znajdujących się w zbiorze treningowym.

Tabela 9.5: Uśrednione wartości wag (w wektorze preferencji), przyporządkowane przez algorytm *EAR*

Algorytm	$\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F1}}}$	$\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F2}}}$	$\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F3}}}$
BPR	0,62	0,63	0,59
ImplicitMF	0,08	0,07	0,06
ItemItem	0,04	0,05	0,10
MostPopular	0,10	0,07	0,05
SVD	0,08	0,08	0,09
UserUser	0,08	0,10	0,11

W tabeli 9.5 zaprezentowano uśrednione wartości wag, które zostały przyporządkowane każdemu z algorytmów rekomendacyjnych przez algorytm *EAR*, przy wykorzystaniu różnych wariantów funkcji oceny. Generalnie najwyższy poziom istotności uzyskał algorytm *BPR*, a reszta algorytmów miała dużo niższy poziom istotności.

W dodatku A na wykresie A.2 zaprezentowano jak z kolejnymi iteracjami zmieniały się wartości wag, przyporządkowane poszczególnym algorytmom (dla sześciu wybranych użytkowników), a na wykresach pudełkowych (rys. A.3) zaprezentowany został rozkład wartości tych wag. Dodatkowo na rysunku A.1 zaprezentowano sześć wykresów zbieżności, które zostały wygenerowane dla sześciu wybranych użytkowników. Widać na nich jak w kolejnych iteracjach kształtowała się wartość funkcji oceny, która na początku przyjmowała mniejsze wartości i rosła wraz z kolejnymi iteracjami. Oznacza to, że algorytm DE optymalizował wektor preferencji, aby jak najlepiej odwzorować preferencje aktywnego użytkownika, zwiększając tym samych wartość miary AP na zbiorze treningowym.

9.3.3 Uwzględnienie w procesie agregacji rekomendacji niskiej jakości

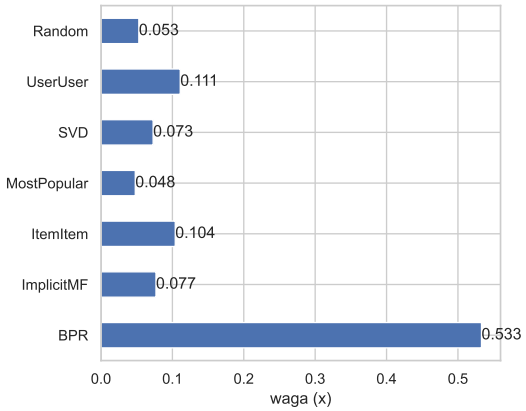
W tym podrozdziale przedstawione zostaną wyniki badań, których celem było zaprezentowanie, jaki wpływ na metody agregujące mają algorytmy, które generują rekomendacje bardzo niskiej jakości. Aby to sprawdzić w procesie agregacji uwzględnione zostały algorytmy, który generowały rekomendacje w sposób losowy (oznaczone jako *Random*).

Porównując wyniki zaprezentowane w tabeli 9.3, z wynikami przedstawionymi po uwzględnieniu rekomendacji wygenerowanych w sposób losowy (tabela 9.6) można zauważyć, że dla wszystkich metod agregujących jakość rekomendacji wyraźnie spadła, choć dla poszczególnych metod w różnym stopniu. Przykładowo analizując wyniki uzyskane przez metodę *BordaFuse* przed uwzględnieniem w procesie agregacji algorytmu *Random_1* można zaobserwować, że uzyskała ona jakość rekomendacji na poziomie 0,1370 (w stosunku do miary *AP@10*). Po wprowadzeniu do procesu agregacji rekomendacji wygenerowanych przez algorytm *Random_1*, jakość rekomendacji zmalała do wartości 0,1210. Podobna sytuacja wystąpiła dla tego algorytmu w stosunku do miary *NDCG@10*, gdzie jakość rekomendacji spadła z wartości 0,2276 do wartości 0,2057. Metodą, która utworzyła agregację najwyższej jakości była metoda *CombSum*, gdzie jakość rekomendacji dla tej metody wyrażoną przy pomocy miary *AP@10* spadła z wartości 0,1393 do wartości 0,1282. Natomiast metodą, która utworzyła agregację najniższej jakości po wprowadzeniu losowych rekomendacji była metoda *CombMax*, której jakość rekomendacji wyrażonej przy pomocy miary *AP@10* spadła z wartości 0,116 do wartości 0,049.

Analizując wyniki, które zostały uzyskane przez zaproponowany algorytm *EAR* z wariantem funkcji oceny $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F3}}}$ można zauważyć, że jakość rekomendacji spadła w nieznacznym stopniu. Przykładowo w odniesieniu do miary *AP@10* jakość rekomendacji obniżyła się z wartości 0,1414 do wartości 0,1382, a w stosunku do miary *NDCG@10* z wartości 0,2296 do wartości 0,2265.

Tabela 9.6: Wyniki eksperymentów przeprowadzonych na 300 losowo wybranych użytkownikach

Algorytm	AP@10	NDCG@10	NDCG@5	Precision@1	Precision@10
Algorytmy rekomendacyjne					
BPR	0,1209	0,2027	0,209	0,2867	0,2157
ImplicitMF	0,1272	0,2114	0,2129	0,2533	0,2263
ItemItem	0,1170	0,1964	0,2092	0,2933	0,2020
SVD	0,1239	0,2112	0,2246	0,3033	0,2153
UserUser	0,1344	0,2177	0,2349	0,3233	0,2173
MostPopular	0,0715	0,1307	0,1269	0,1733	0,1490
Random_1	0,0007	0,0013	0,0022	0,0002	0,0011
Metody agregujące					
BordaFuse	0,1210	0,2057	0,2134	0,2700	0,2150
CombMax	0,0487	0,0971	0,1014	0,1667	0,1047
CombMed	0,1224	0,2055	0,2082	0,2967	0,2123
CombMin	0,0926	0,1681	0,1809	0,2233	0,1737
CombSum	0,1282	0,2134	0,2236	0,2800	0,2207
Zaproponowany algorytm					
$EAR_{DE_{F3}}$	**0,1382**	**0,2265**	**0,2326**	**0,3167**	**0,2417**



Rysunek 9.7: Wartości wag (w wektorze preferencji), przyporządkowane przez algorytm *EAR*

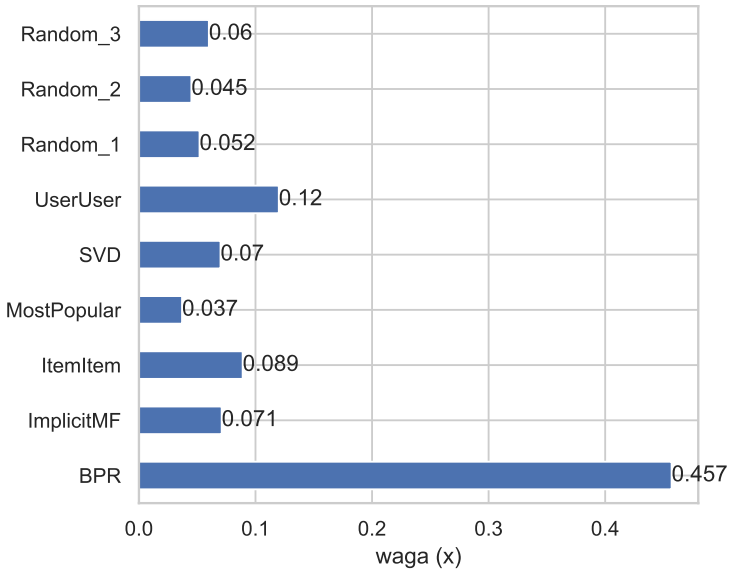
Na rysunku 9.7 przedstawione zostały średnie wartości wag, które zostały przyporządkowane poszczególnym algorytmom przez algorytm *EAR*. Widać na nich, że losowemu algorytmowi przyporządkowano niewielką wagę, która wynosiła 0,053. Aby dodatkowo przetestować wpływ wygenerowanych losowo rekomendacji na metody agregujące, przeprowadzono badania, w których uwzględniono aż 3 algorytmy generujące rekomendacje w sposób losowy, a wyniki tych eksperymentów zostały zaprezentowane w tabeli 9.7.

Tabela 9.7: Wyniki eksperymentów przeprowadzonych na 300 losowo wybranych użytkownikach

Algorytm	**AP@10**	**NDCG@10**	**NDCG@5**	**Precision@1**	**Precision@10**
Algorytmy rekomendacyjne					
BPR	0,1209	0,2027	0,209	0,2867	0,2157
ImplicitMF	0,1272	0,2114	0,2129	0,2533	0,2263
ItemItem	0,1170	0,1964	0,2092	0,2933	0,2020
SVD	0,1239	0,2112	0,2246	0,3033	0,2153
UserUser	0,1344	0,2177	0,2349	0,3233	0,2173
MostPopular	0,0715	0,1307	0,1269	0,1733	0,1490
Random_1	0,0007	0,0013	0,0022	0,0039	0,0021
Random_2	0,0004	0,0009	0,0014	0,0021	0,0011
Random_3	0,0033	0,0028	0,0025	0,0071	0,0023
Metody agregujące					
BordaFuse	0,1133	0,1952	0,1944	0,2867	0,2107
CombMax	0,0371	0,0784	0,0859	0,1167	0,086
CombMed	0,1168	0,2025	0,2055	0,2600	0,2137
CombMin	0,0766	0,1462	0,1577	0,2467	0,1517
CombSum	0,1149	0,2007	0,2064	**0,3000**	0,2080
Zaproponowany algorytm					
$EAR_{DE_{F3}}$	**0,133**	**0,2209**	**0,2325**	0,2967	**0,2317**

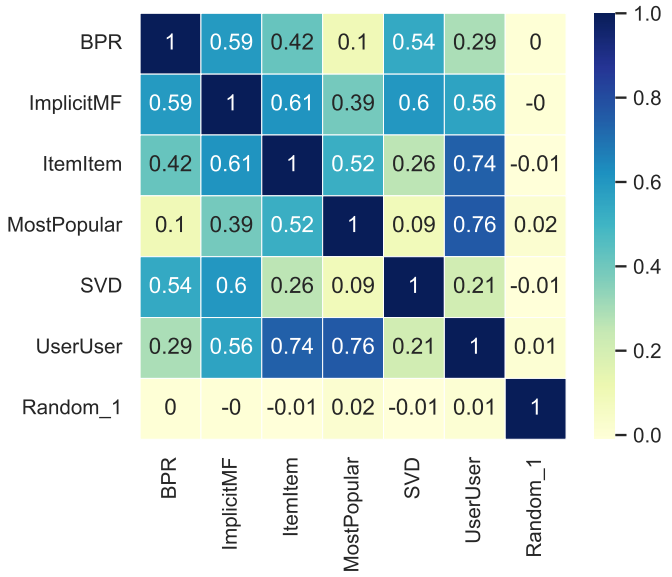
Po analizie porównawczej z eksperymentami przeprowadzonymi z jednym losowym algorytmem (tabela 9.6), można zauważyć, że metody agregujące wygenerowały rekomendacje, które były niższej jakości, w stosunku do wszystkich uwzględnionych w badaniach miar jakości. Taki wynik nie jest oczywiście zaskoczeniem, ponieważ większa liczba algorytmów generujących rekomendacje w sposób losowy, wprowadza do procesu agregacji większą ilość szumu, co musi odbić się negatywnie na jakości tworzonej agregacji. Najbardziej odpornym na ten problem wydaje się być znowu algorytm *EAR* z wariantem funkcji oceny $\text{EAR}_{\text{DE}_{\text{F3}}}$, ponieważ w przypadku tego algorytmu jakość rekomendacji spadła w najmniejszym stopniu (w porównaniu do innych metod agregujących).

Dodatkowo na rysunku 9.8 przedstawione zostały średnie wartości wag, które zostały przyporządkowane poszczególnym algorytmom przez algorytm *EAR*. Widać na nich, że algorytmom, które generowały rekomendacje w sposób losowy, algorytm *EAR* przyporządkował stosunkowo niewielką wagę.



Rysunek 9.8: Wartości wag (w wektorze preferencji), które zostały przyporządkowane przez algorytm *EAR*

Dodatkowo w celu zaprezentowania, że rekomendacje wygenerowane w sposób losowy nie były skorelowane z innymi algorytmami, poniżej zaprezentowane zostały macierze korelacji. Na rysunku 9.9 przedstawiona została macierz korelacji z jednym algorytmem *Random_1*, a w dodatku A na rysunku A.6, przedstawiona została macierz korelacji z trzema algorytmami: *Random_1*, *Random_2*, *Random_3*. Na przykładzie tych dwóch macierzy można zauważyć, że algorytmy generujące rekomendacje w sposób losowy, miały z innymi algorytmami rekomendacyjnymi oraz metodami agregującymi korelację bliską wartości 0.



Rysunek 9.9: Macierz korelacji Tau Kendalla dla metod agregujących po uwzględnieniu algorytmu *Random_1*

Podsumowując, po analizie przeprowadzonych eksperymentów można zauważyć, że algorytmy agregujące w różnym stopniu radzą sobie z algorytmami, które generują rekomendacje niskiej jakości. Nienadzorowanymi metodami agregującymi, które generują agregację stosunkowo dobrej jakości są to metody: *BordaFuse*, *CombMed*, *CombSum*.

Natomiast na podstawie przeprowadzonych badań, najlepiej z szumem informacyjnym, wprowadzanym do procesu agregacji przez algorytmy niskiej jakości, radzi sobie algorytm *EAR*. Dzięki wykorzystaniu danych historycznych, tworzy on spersonalizowany wektor preferencji dla każdego z aktywnych użytkowników i w którym algorytmy generujące rekomendacje niskiej jakości, otrzymują odpowiednio niską wagę. Dogłębniejsza analiza wpływu rankingów niskiej jakości, które są uwzględniane w procesie agregacji na jakość rekomendacji uzyskiwaną przez poszczególne metody agregujące jest bez wątpienia ciekawym kierunkiem przyszłych prac badawczych.

9.3.4 Uwzględnienie w procesie agregacji rankingów innych użytkowników

W tym podrozdziale przetestowana zostanie modyfikacja podstawowej wersji algorytmu, uwzględniająca w procesie agregacji rekomendacje, które zostały wygenerowane dla innych użytkowników w systemie. Do wyznaczenia takich użytkowników wykorzystany został algorytm *kNN*, który po obliczeniu odległości euklidesowej pomiędzy użytkownikami w systemie, wybiera k użytkowników, którzy są najbardziej podobni do aktywnego użytkownika u_A. Następnie rekomendacje wygenerowane dla tych użytkowników są uwzględniane w procesie agregacji. Bardziej szczegółowy opis tej modyfikacji, można znaleźć w podrozdziale 7.4.

Tabela 9.8: Wyniki eksperymentów przeprowadzonych na 100 losowo wybranych użytkownikach

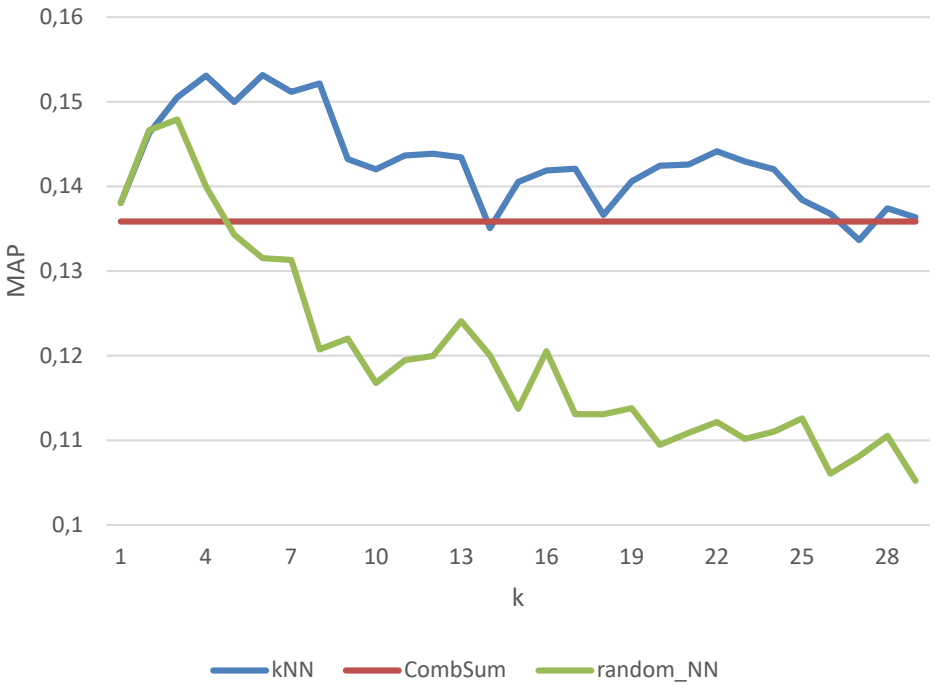
Algorytm	**AP@10**	**NDCG@10**	**NDCG@5**	**Precision@1**	**Precision@10**
Algorytmy rekomendacyjne					
BPR	0,1209	0,2027	0,2090	0,2867	0,2157
ImplicitMF	0,1272	0,2114	0,2129	0,2533	0,2263
ItemItem	0,1170	0,1964	0,2092	0,2933	0,2020
SVD	0,1239	0,2112	0,2246	0,3033	0,2153
UserUser	0,1344	0,2177	0,2349	0,3233	0,2173
MostPopular	0,0715	0,1307	0,1269	0,1733	0,1490
Metody agregujące					
BordaFuse	0,1370	0,2276	0,2361	0,3133	0,2343
CombMax	0,1159	0,1948	0,2056	0,2867	0,1987
CombMed	0,1380	0,2244	0,2358	0,2967	0,2307
CombMin	0,1133	0,2003	0,2152	0,2700	0,2050
CombSum	0,1393	0,2279	0,2376	**0,3367**	0,2340
Zaproponowany algorytm					
$\text{EAR}_{\text{DE}_{\text{F3}}}$	0,1258	0,2119	0,2221	0,2967	0,2230
$\text{EAR}_{\text{DE}_{\text{F3(5)}}}$	0,1284	0,2135	0,2193	0,2967	0,2273
$\text{EAR}_{\text{DE}_{\text{F3(10)}}}$	**0,1414**	**0,2296**	**0,2396**	0,3233	**0,2427**
$\text{EAR}_{\text{DE}_{\text{F3(15)}}}$	0,1321	0,2140	0,2204	0,3033	0,2320

W tabeli 9.8 zaprezentowane zostały wyniki badań, które przedstawiają jakość rekomendacji uzyskaną przez zaproponowany algorytm EAR. Rekomendacje wygenerowane przez ten algorytm, powstały w wyniku agregacji rekomendacji wygenerowanych dla aktywnego użytkownika u_A (oznaczenie: $\text{EAR}_{\text{DE}_{\text{F3}}}$) oraz poprzez uwzględnienie w tym procesie również rekomendacji, które zostały wygenerowane dla: 5 najbliższych sąsiadów (oznaczenie: $\text{EAR}_{\text{DE}_{\text{F3(5)}}}$), 10 najbliższych sąsiadów (oznaczenie: $\text{EAR}_{\text{DE}_{\text{F3(10)}}}$) oraz 15 najbliższych sąsiadów $\text{EAR}_{\text{DE}_{\text{F3(15)}}}$).

Analizując wyniki eksperymentów zaprezentowane w tabeli 9.8 można zauważyć, że po wprowadzeniu dodatkowych rekomendacji pochodzących od użytkowników będących w najbliższym sąsiedztwie aktywnego użytkownika u_A, jakość rekomendacji uległa poprawie. Najlepsza agregacja w stosunku do czterech miar jakości (*AP@10*, *NDCG@10*,

NDCG@5, *Precision@10*), została utworzona przy pomocy wariantu $\text{EAR}_{\text{DE}_{\text{F3(10)}}}$, czyli kiedy wielkość sąsiedztwa wynosiła 10.

Aby lepiej zaobserwować, jak zmiana wartości parametru *k*, wpływa na jakość tworzonej agregacji na wykresie 9.10 przedstawiono jakość rekomendacji dla różnych wartości tego parametru, wyrażoną przy pomocy miary *MAP*.



Rysunek 9.10: Wpływ wartości parametru k w algorytmie *kNN*, na jakość tworzonej agregacji (wyrażonej przy pomocy miary MAP)

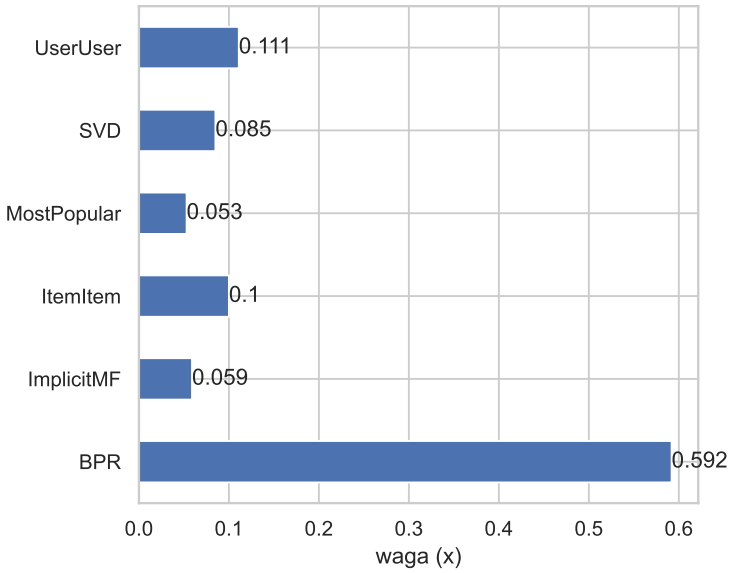
Analizując wyniki tego eksperymentu można zauważyć, że agregacja utworzona na podstawie rekomendacji pochodzących od użytkowników z losowego sąsiedztwa (oznaczenie: *random_NN*), generuje rekomendacje niższej jakości, niż te, które zostały utworzone na podstawie sąsiedztwa wybranego przy pomocy algorytmu *kNN* (oznaczenie: *kNN*). Sugeruje to, że uwzględnienie w procesie agregacji rekomendacji, które zostały wygenerowane dla innych użytkowników może podnieść jakość rekomendacji dla aktywnego użytkownika u_A, ale tylko w przypadku, gdy sąsiedztwo to zostanie dobrane w odpowiedni sposób. Oczywiście istnieją różne sposoby obliczania miary podobieństwa pomiędzy użytkownikami i można by tutaj zastosować inne miary odległości. Jest to ciekawy kierunek przyszłych prac badawczych.

Dodatkowo na wykresie 9.10 można zaobserwować, jaki wpływ na jakość generowanych rekomendacji ma parametr *k*. Dla algorytmu *kNN* jakość rekomendacji była wyższa w sytuacji, gdy sąsiedztwo wynosiło od 4 do 8 użytkowników. Gdy parametr k przyjmował wartości powyżej 8, nastąpił zauważalny spadek jakości generowanych rekomendacji. Takie wyniki nie są jednak zaskoczeniem, ponieważ można przypuszczać, że im użytkownicy

będą mniej podobni do aktywnego użytkownika u_A, tym będzie to wprowadzać więcej szumu informacyjnego do procesu agregacji.

Natomiast analizując wyniki uzyskane przez algorytm *random_NN* można zauważyć, że powyżej 3 sąsiadów jakość rekomendacji zaczęła maleć. Początkowy wzrost wartości *MAP* w przypadku tego wariantu można wyjaśnić tym, że mała liczba losowych użytkowników wprowadzała stosunkowo niewielką ilość szumu informacyjnego do procesu agregacji i algorytm *EAR*, dzięki fazie treningowej, potrafił przyporządkować odpowiednio niskie wagi algorytmom, które generowały rekomendacje niskiej jakości. Jednak wraz ze wzrostem liczby użytkowników (parametr k) można zauważyć, że jakość rekomendacji systematycznie spadała.

Aby przedstawić uśredniony rozkład wartości wag, które zostały przyporządkowane poszczególnym algorytmom przez algorytm *EAR* w wektorze preferencji, poniżej zaprezentowany został stosowny wykres.



Rysunek 9.11: Wartości wag (w wekorze preferencji), które zostały przyporządkowane przez algorytm *EAR* (wariant $\text{EAR}_{\text{DE}_{\text{F3}}}$)

Wykres 9.11 pokazuje, jak rozkładały się wartości wag dla wariantu $\text{EAR}_{\text{DE}_{\text{F3}}}$ (bez najbliższych sąsiadów). W przypadku tego wariantu najwyższą wagę uzyskał algorytm *BPR*. Reszta algorytmów uzyskała stosunkowo niewielki poziom istotności. Taki rozkład wartości wag w wektorze preferencji oznacza, że najbardziej istotnym algorytmem w procesie agregacji dla większości użytkowników był algorytm *BPR*.

W dodatku A znajduje się wykres A.4, reprezentujący uśrednione wagi wektora preferencji, które zostały przyporządkowane przez wariant $\text{EAR}_{\text{DE}_{\text{F3(5)}}}$ poszczególnym algorytmom. Można zauważyć, że istotność algorytmu *BPR*, który wygenerował rekomendacje dla aktywnego użytkownika zmalała na rzecz innych algorytmów wchodzących w skład agregacji. Generalnie jednak można zaobserwować, że w wektorze preferencji, większą

wartość wagi w stosunku do innych algorytmów uzyskał algorytm *BPR*, a najniższą algorytm *MostPopular*. Jest to ciekawa obserwacja, ponieważ może oznaczać ona, że analizując wektory preferencji można uzyskać interesujące informacje na temat tego, które algorytmy powinny zostać uwzględnione w procesie agregacji. Podobne wnioski można wyciągnąć analizując wyniki eksperymentów, które zostały zaprezentowane na wykresie A.5. Przedstawiono na nim uśredniony wektor preferencji dla wariantu $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F3(10)}}}$.

Wprowadzenie zaproponowanej przez autora modyfikacji, uwzględniającej w procesie agregacji rekomendacje wygenerowane dla sąsiadów aktywnego użytkownika u_A, może zwiększyć jakość generowanych rekomendacji. Użytkownicy byli wyznaczani przy pomocy algorytmu *kNN*. Niestety liczba sąsiadów k, którzy uczestniczą w procesie agregacji również ma istotne znaczenie i jest kolejnym parametrem, który trzeba odpowiednio dostroić. Uwzględnianie wszystkich algorytmów w procesie agregacji jest również prawdopodobnie niewłaściwe, co pokazała analiza wag wektora preferencji. Wydaje się, że ciekawą modyfikacją zaproponowanego algorytmu byłoby również wprowadzenie parametru, który uwzględniałby pewien próg, dopiero po przekroczeniu którego dany algorytm uczestniczyłby w procesie agregacji.

9.4 Badania końcowe

Ze względu na to, że poszukiwanie wektora preferencji przez algorytm EAR jest czasochłonne, to przeprowadzone wcześniej badania, zazwyczaj były realizowane na mniejszej grupie użytkowników. Aby jednak ostatecznie potwierdzić skuteczność zaproponowanego podejścia, badania końcowe zostaną zrealizowane na pełnym zbiorze użytkowników, którzy byli dostępni w zbiorze U, czyli 943 użytkownikach. Dodatkowo w celu wykazania istotności statystycznej prezentowanych wyników, przeprowadzony zostanie test statystyczny Wilcoxona. W badaniach ponadto uwzględnione zostaną rankingi, które zostały wygenerowane dla użytkowników wybranych przez algorytm *kNN*, zgodnie z modyfikacją opisaną w podrozdziale 7.4.

Analizując wyniki przedstawione w tabeli 9.9 można zauważyć, że algorytmem rekomendacyjnym, który generował rekomendacje najwyższej jakości był algorytm *UserUser*. Natomiast algorytmem, który generował rekomendacje najniższej jakości, był niespersonalizowany algorytm *MostPopular*. Generalnie wszystkie spersonalizowane algorytmy rekomendacyjne generowały rekomendacje zbliżonej jakości. Ponadto analizując wyniki uzyskane przez algorytmy agregujące można zauważyć, że agregacja najwyższej jakości została utworzona przy wykorzystaniu metod: *CombMed*, *BordaFuse* oraz *CombSum*. Metodami, które utworzyły agregację wyraźnie gorszą, były metody: *CombMax* oraz *CombMin*.

Tabela 9.9: Jakość rekomendacji na pełnym zbiorze użytkowników. Najlepsze wyniki dla danej miary zostały pogrubione

Algorytm	**MAP@10**	**NDCG@10**	**NDCG@5**	**Precision@1**	**Precision@10**
Algorytmy rekomendacyjne					
BPR	0,1025	0,1796	0,1720	0,2354	0,1671
ImplicitMF	0,1110	0,1879	0,1764	0,2322	0,1754
ItemItem	0,1123	0,1936	0,1878	0,2566	0,1726
SVD	0,1023	0,1783	0,1763	0,2365	0,1604
UserUser	0,1156	0,1946	0,1901	0,2503	0,1764
MostPopular	0,0583	0,1097	0,1013	0,1432	0,1102
Metody agregujące					
BordaFuse	0,1200	0,2057	0,1967	0,2418	0,1888
CombMax	0,0927	0,1633	0,1621	0,2174	0,1499
CombMed	0,1207	0,2025	0,1948	0,2545	0,1857
CombMin	0,1043	0,1833	0,1748	0,2238	0,1680
CombSum	0,1197	0,2040	0,1950	0,2545	0,1874
Zaproponowany algorytm					
$EAR_{DE_{F3}}$	0,1196	0,2042	0,1971	0,2556	0,1867
$EAR_{DE_{F3(5)}}$	0,1292	0,2152	**0,2086**	**0,2609**	0,1973
$EAR_{DE_{F3(10)}}$	**0,1299**	**0,2156**	0,2065	0,2545	**0,1985**
$EAR_{DE_{F3(15)}}$	0,1283	0,2130	0,2059	0,2577	0,1964

W tabeli 9.9 przedstawione zostały również wyniki, które zostały uzyskane przez zaproponowany w rozprawie algorytm. W podstawowej wersji algorytmu (oznaczonej jako $EAR_{DE_{F3}}$), algorytm ten uzyskał wyniki, które były lepsze od wszystkich algorytmów rekomendacyjnych. Niestety był on gorszy od algorytmów agregujących *BordaFuse*, *CombMed* oraz *CombSum* (dla miary $AP@10$). Analizując jednak wyniki, jakie zostały uzyskane przez zaproponowany algorytm z modyfikacją uwzględniającą najbliższe sąsiedztwo (oznaczonych odpowiednio: $EAR_{DE_{F3(5)}}$, $EAR_{DE_{F3(10)}}$, $EAR_{DE_{F3(15)}}$), można zauważyć, że modyfikacja ta uzyskała wyniki wyraźnie lepsze od podstawowej wersji algorytmu $EAR_{DE_{F3}}$. Zostało to dodatkowo potwierdzone przez testy statystyczne, które zostały przedstawione w tabeli 9.10. Świadczy to o tym, że modyfikacja ta poprawia jakość generowanej agregacji, a uwzględnienie rankingów innych użytkowników w tym procesie, może pozytywnie

wpłynąć na jakość generowanych rekomendacji.

Należy dodać, że liczba użytkowników (sąsiadów), którzy są uwzględniani w procesie agregacji również ma znaczenie. Analizując wyniki przedstawione w tabeli 9.9 można zauważyć, że uwzględniając rankingi pięciu najbliższych sąsiadów użytkownika u_A, algorytm ten uzyskał najlepsze wyniki dla miar *NDCG@5* oraz *Precision@1*. Świadczy to o tym, że taka liczba podobnych użytkowników wpływała lepiej na rekomendacje, które były prezentowane na początkowych pozycjach w rankingu. Natomiast uwzględnienie w procesie agregacji rankingów dziesięciu najbliższych sąsiadów, zwiększało jakość rekomendacji, która była wyrażona przy pomocy miar: *AP@10*, *NDCG@10* oraz *Precision@10*. Świadczy to o tym, że algorytm ten uzyskiwał lepsze wyniki dla pierwszych dziesięciu przedmiotów, które były prezentowane użytkownikowi.

Należy jednak zauważyć, że zwiększanie liczby sąsiadów niekoniecznie musi wiązać się ze zwiększeniem jakości generowanych rekomendacji. Przykładowo przy uwzględnieniu rankingów wygenerowanych dla 15 najbliższych sąsiadów aktywnego użytkownika u_A, jakość generowanej rekomendacji nie zwiększyła się. Świadczy to o tym, że liczba użytkowników, którzy są uwzględniani w procesie agregacji nie może być zbyt duża, ponieważ zbyt oddaleni użytkownicy wprowadzają za dużo szumu informacyjnego do tworzonej agregacji, co w następstwie powoduje spadek jakości rekomendacji.

Tabela 9.10: Test statystyczny Wilcoxona, obliczony pomiędzy algorytmami rekomendacyjnymi i metodami agregującymi, a zaproponowanym algorytmem (dla miary *AP@10*)

Typ	Algorytm	$EAR_{DE_{F3}}$	$EAR_{DE_{F3(5)}}$	$EAR_{DE_{F3(10)}}$	$EAR_{DE_{F3(15)}}$
Algorytmy rekomendacyjne	BPR	$1,32 \cdot 10^{-9}$	$5,20 \cdot 10^{-14}$	$2,27 \cdot 10^{-10}$	$2,67 \cdot 10^{-9}$
	ImplicitMF	$7,28 \cdot 10^{-6}$	$7,59 \cdot 10^{-10}$	$6,96 \cdot 10^{-9}$	$8,955 \cdot 10^{-8}$
	ItemItem	$1,14 \cdot 10^{-1}$	$4,01 \cdot 10^{-4}$	$1,71 \cdot 10^{-4}$	$1,12 \cdot 10^{-2}$
	UserUser	$1,34 \cdot 10^{-1}$	$5,10 \cdot 10^{-5}$	$3,10 \cdot 10^{-4}$	$2,50 \cdot 10^{-3}$
	SVD	$3,17 \cdot 10^{-9}$	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-15}$
	MostPopular	$3,32 \cdot 10^{-15}$	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-15}$
Algorytmy agregujące	BordaFuse	$3,49 \cdot 10^{-1}$	$2,40 \cdot 10^{-2}$	$1,47 \cdot 10^{-1}$	$2,11 \cdot 10^{-1}$
	CombMax	$1,49 \cdot 10^{-13}$	$4,00 \cdot 10^{-2}$	$< 10^{-15}$	$< 10^{-15}$
	CombMed	$6,17 \cdot 10^{-1}$	$6,67 \cdot 10^{-3}$	$2,44 \cdot 10^{-2}$	$1,43 \cdot 10^{-1}$
	CombMin	$7,12 \cdot 10^{-6}$	$1,2 \cdot 10^{-13}$	$1,19 \cdot 10^{-13}$	$5,17 \cdot 10^{-15}$
	CombSum	$8,53 \cdot 10^{-1}$	$5,10 \cdot 10^{-3}$	$3,50 \cdot 10^{-2}$	$2,81 \cdot 10^{-2}$

Rozdział 10

Podsumowanie

W rozprawie zaproponowano algorytm agregujący *ewolucyjnej agregacji rang* (EAR), który wykorzystując historyczne interakcje użytkownika z systemem rekomendacyjnym, modeluje jego preferencje. Preferencje te, reprezentowane są przez wektor liczb rzeczywistych, gdzie poszczególne elementy tego wektora określają w jakim stopniu rekomendacja wygenerowana przez dany algorytm, będzie wpływać na końcową agregację. Do wyszukania tego wektora, wykorzystany został algorytm ewolucji różnicowej, który umożliwia bezpośrednią optymalizację funkcji rangującej w stosunku do miar, służących do oceny jakości wygenerowanych rekomendacji. Takie podejście umożliwiło stworzenie nadzorowanego algorytmu agregującego, którego efektywność została zweryfikowana przez liczne eksperymenty przeprowadzone na popularnym i ogólnodostępnym zbiorze danych *MovieLens 100k*. Dodatkowo wyniki badań zostały potwierdzone przez testy statystyczne.

Na wybór tematyki niniejszej rozprawy, niewątpliwie miała wpływ rosnąca popularność prac badawczych, prowadzonych w kontekście *rodzin klasyfikatorów*. Ich wykorzystanie okazuje się szczególnie efektywne w sytuacjach, w których mamy do czynienia z danymi złożonymi, wielowymiarowymi i zaszumionymi [61]. Dodatkowo analizując aktualną literaturę związaną z systemami rekomendacyjnymi, można zauważyć, że niektórzy badacze sugerują przedstawienie problemu fuzji algorytmów rekomendacyjnych, jako problemu agregacji rang. Jednak szczegółowe motywacje uzasadniające podjęcie tej tematyki, zostały już zaprezentowane w podrozdziale 1.1.

Aby przeprowadzenie badań było możliwe, niezbędne okazało się przygotowanie specjalnego środowiska badawczego, ponieważ żadne dostępne oprogramowanie nie spełniało wszystkich wymagań. W tym celu wykorzystany został język Python oraz biblioteki, które omówione zostały w podrozdziale 8.2. Wykorzystanie bibliotek programistycznych zminimalizowało prawdopodobieństwo wystąpienia potencjalnych błędów, podczas samego procesu implementacji. Należy również zauważyć, że implementacja zaproponowanego algorytmu nie była trywialna, ponieważ wymagała ona połączenia ze sobą zagadnień z czterech obszarów tematycznych: systemów rekomendacyjnych, agregacji rang, uczenie się rankingu oraz algorytmów metaheurystycznych.

10.1 Wnioski

Na podstawie wyników eksperymentów, które szczegółowo zostały zaprezentowane w rozdziale 9 można wyciągnąć następujące wnioski:

1. Przeprowadzona w podrozdziale 9.2 wstępna analiza, która została zrealizowana na 25 losowo wybranych użytkownikach wykazała, że algorytmy rekomendacyjne generują rekomendacje różnej jakości. Z tego względu zasadne wydaje się być wykorzystanie metod agregujących. Po ich zastosowaniu jakość rekomendacji w przypadku większości użytkowników uległa poprawie, choć dla poszczególnych metod w różnym stopniu.

2. Na bazie analizy wyników eksperymentów zaprezentowanych w podrozdziałach 9.3.1 oraz 9.4 można stwierdzić, że niektóre z klasycznych metod agregujących poprawiły jakość generowanych rekomendacji, w stosunku do wszystkich uwzględnionych w badaniach algorytmów rekomendacyjnych. Metody agregujące, które uzyskały najlepsze wyniki to: *BordaFuse*, *CombMed*, *CombSum*.

3. Badania przeprowadzone na zaproponowanym algorytmie w podrozdziale 9.3.1 pokazały, że z powodzeniem można wykorzystać algorytm DE do bezpośredniej optymalizacji miary AP. Algorytm ten wykorzystując zaproponowane warianty funkcji oceny, przeważnie uzyskiwał wyniki lepsze od algorytmów rekomendacyjnych oraz klasycznych metod agregujących, szczególnie dla wariantu funkcji oceny $\mathrm{EAR}_{\mathrm{DE}_{\mathrm{F3}}}$.

4. Uwzględnienie rekomendacji wygenerowanych dla innych użytkowników, może poprawić jakość generowanych rekomendacji. Badania zaprezentowane w podrozdziale 9.3.4 pokazują jednak, że wybór użytkowników jest tutaj bardzo istotny. Jeżeli użytkownicy zostali wybrani przez algorytm *kNN*, jakość rekomendacji wzrastała. Jeżeli jednak byli oni wybierani w sposób losowy, to jakość rekomendacji wyraźnie malała.

5. Uwzględnienie rekomendacji niskiej jakości w procesie agregacji wyraźnie pogarszało jej jakość (podrozdział 9.3.3), szczególnie dla klasycznych metod agregujących. W przypadku algorytmu *EAR* jakość rekomendacji również zmalała, choć w dużo mniejszym stopniu. Można to wytłumaczyć tym, że ze względu na to, że występuje tutaj faza treningowa, umożliwia ona wykrycie algorytmów, które generują rekomendacje niskiej jakości.

6. Dodatkowo badania zaprezentowane w podrozdziale 8.3 wykazały, że należy pamiętać o odpowiednim dostrojeniu parametrów algorytmów rekomendacyjnych, ponieważ wpływ poszczególnych parametrów na jakość uzyskiwanych wyników jest bardzo zróżnicowany.

7. Zasadniczą zaletą zaproponowanego podejścia jest łatwość jego wdrożenia do istniejącego systemu rekomendacyjnego. Ze względu na to, że agregacja odbywa się tylko na podstawie wygenerowanych rekomendacji (rankingów), nie występuje tutaj potrzeba ingerencji w implementacje samych algorytmów rekomendacyjnych.

10.2 Nawiązanie do tezy rozprawy oraz celów dodatkowych

Bazując na wynikach eksperymentów przedstawionych w rozdziale 9 oraz wyciągniętych na ich podstawie wnioskach (podrozdział 10.1), autor niniejszej rozprawy stwierdza, że postawiona w podrozdziale 1.2 teza rozprawy *Zaproponowany algorytm ewolucyjnej agregacji rang poprawia jakość generowanej agregacji, w porównaniu z wybranymi metodami zaproponowanymi w literaturze* została potwierdzona. Główny cel pracy, jakim było opracowanie algorytmu agregującego, bazującego na algorytmie ewolucji różnicowej, został zrealizowany. Dodatkowo w rozprawie zrealizowano następujące cele dodatkowe:

1. **Przegląd literatury związanej z tematyką niniejszej rozprawy** – w pracy zaprezentowany został przegląd literaturowy, nawiązujący do podstawowych informacji związanych z tematyką rozprawy, czyli: systemów rekomendacyjnych (rozdział 2), ewaluacji systemów rekomendacyjnych (rozdział 3), algorytmu ewolucji różnicowej (rozdział 4), agregacji rang (rozdział 5) oraz nauki rangowania (rozdział 6).

2. **Zaprezentowanie zbioru danych wykorzystanego do przeprowadzenia eksperymentów** – w podrozdziale 8.1 przedstawiona została analiza zbioru danych *MovieLens 100k*, który został wykorzystany do przeprowadzenia badań w części eksperymentalnej.

3. **Dostrojenie parametrów poszczególnych algorytmów wchodzących w skład agregacji** – w podrozdziale 8.3 zaprezentowany został proces dostrajania parametrów algorytmów rekomendacyjnych, które wykorzystane zostały do generowania rekomendacji i na podstawie których tworzona była agregacja.

4. **Włączenie do procesu agregacji rankingów innych użytkowników** – w podrozdziale 9.3.4 zaprezentowane zostały wyniki badań, przedstawiające wpływ rankingów innych użytkowników na jakość tworzonej agregacji, gdzie użytkownicy zostali wyznaczeni przy pomocy algorytmu *kNN*.

5. **Sprawdzenie wpływu różnych wariantów funkcji oceny na jakość agregacji** – w podrozdziale 9.3.1 zaprezentowane zostały wyniki badań, w których przeprowadzono analizę 3 wariantów funkcji oceny algorytmu DE oraz wpływu parametru λ na jakość tworzonej agregacji.

10.3 Prace na przyszłość

Badania zrealizowane w niniejszej rozprawie oraz przedstawiona modyfikacja zaproponowanego algorytmu są tylko wstępem do dalszych prac badawczych nad nadzorowanymi algorytmami agregującymi, szczególnie w kontekście systemów rekomendacyjnych. Zaproponowany algorytm agregujący można spróbować udoskonalić, implementując następujące modyfikacje:

1. Utworzenie globalnego wektora preferencji w_{global} – taki wektor można utworzyć poprzez modyfikację funkcji oceny w algorytmie DE i optymalizowanie wektora preferencji nie w stosunku do miary AP, tylko do miary MAP, uwzględniając w tym procesie wszystkich użytkowników w systemie rekomendacyjnym. Umożliwi to wyszukanie takiego wektora preferencji, który będzie generalizował preferencję w stosunku do wszystkich użytkowników w systemie. Następnie wektor ten można wykorzystać w sytuacji, w której do systemu loguje się nowy użytkownik, czyli taki, który nie posiada żadnej historii aktywności.

2. Utworzenie lokalnego (grupowego) wektora preferencji w_{local} – zaproponowane w punkcie poprzednim rozwiązanie można zmodyfikować i spróbować uzyskać od użytkownika pewną informację zwrotną, przykładowo prosząc go o wypełnienie krótkiej ankiety. Następnie można wyszukać innych użytkowników, którzy podobnie taką ankietę wypełnili i optymalizować miarę MAP w stosunku do takiej grupy użytkowników.

Prace badawcze nad zaproponowanym algorytmem można dodatkowo rozszerzyć o:

1. Inne nadzorowane algorytmy agregujące – zalecanym byłoby uwzględnienie w raportowanych wynikach algorytmów agregujących, które również bazują na uczeniu nadzorowanym. Należy jednak pamiętać o tym, że często ich implementacja jest dużo bardziej skomplikowana oraz mogą wymagać czasochłonnego procesu dostrajania parametrów.

2. Sprawdzenie wpływu innych wariantów mutacji na jakość tworzonej agregacji – w niniejszej rozprawie wykorzystano tylko podstawowy i jeden z najbardziej popularnych wariantów mutacji algorytmu DE. Zasadnym byłoby sprawdzenie, jak inne warianty mutacji, wpływają na jakość tworzonej agregacji.

3. Uwzględnienie w badaniach większej liczby zbiorów danych – badania zaprezentowane w niniejszej rozprawie były realizowane na jednym i stosunkowo małym zbiorze danych. W przyszłości badania powinny być przeprowadzane na większych i bardziej aktualnych danych.

4. Sprawdzenie innych metaheurystyk – w niniejszej rozprawie do poszukiwania wektora preferencji w_{u_A}, wykorzystywany został algorytm DE. Jednak ciekawym kierunkiem przyszłych prac badawczych, byłoby sprawdzenie efektywności innych metaheurystyk.

5. Dostrojenie parametrów algorytmu DE – w rozprawie pominięto proces dostrajania parametrów algorytmu DE, ze względu na znaczną czasochłonność tego procesu. Autor rozprawy wykorzystał parametry pochodzące z jego wcześniejszych prac [21]. W przyszłości należy jednak dokładniej przetestować wpływ poszczególnych parametrów na uzyskiwane wyniki.

6. Remisy w rankingach – niektóre przedmioty w rankingu mogą mieć przyporządkowaną identyczną wartość, co utrudnia ustalenie ich kolejności. Należy dokładniej przeanalizować jak taka sytuacja, wpływa na jakość uzyskiwanych wyników.

Innymi ciekawymi kierunkami badań są zagadnienia związane z:

1. Wykorzystaniem technik uczenia się metryki – uczenie się metryki (z ang. *metric learning* [25]) to technika, w której nie wykorzystujemy jednej konkretnej miary jakości (np. AP), tylko stosujemy algorytmy, które same taką miarę definiują. Niektórzy badacze wykorzystali ją już z powodzeniem w systemach rekomendacji (np. [169]).

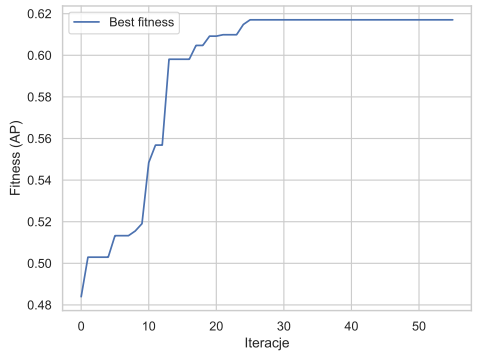
2. Wykorzystaniem wektorów preferencji w innych obszarach – ciekawym kierunkiem przyszłych prac badawczych jest wykorzystanie zaproponowanego podejścia w innych obszarach nauki, gdzie wektory preferencji również znajdują zastosowanie (np. w klasyfikacji [100]).

Słownik symboli

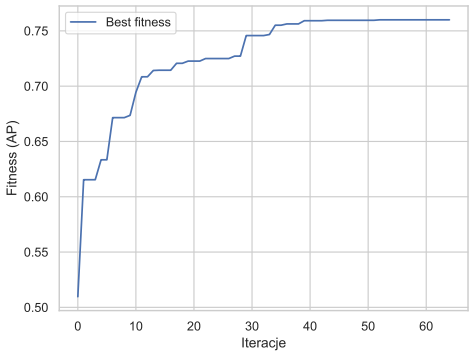
Symbol	Opis
U	zbiór wszystkich użytkowników
I	zbiór wszystkich przedmiotów
R	macierz preferencji (ocen) użytkownik-przedmiot o rozmiarze $\|U\| \times \|I\|$
S	zbiór n algorytmów rekomendacyjnych $S = \{S^1,\ S^2,\ ...,\ S^n\}$
S^r	algorytm r znajdujący się w zbiorze S
u	ogólny użytkownik systemu rekomendacyjnego
u_i	konkretny użytkownik systemu rekomendacyjnego
u_A	aktywny użytkownik systemu rekomendacyjnego, czyli taki, dla którego generowane są rekomendacje
x	ogólny *przedmiot* znajdujący się w systemie rekomendacyjnym
x_j	konkretny *przedmiot* znajdujący się w systemie rekomendacyjnym
τ	ogólny ranking
τ_A^r	ranking zarekomendowany użytkownikowi u_A przez algorytm S^r
$\tau\left(x_j\right)$	pozycja przedmiotu x_j w rankingu τ
Ψ	funkcja agregująca
τ_A^*	ranking utworzony po procesie agregacji dla użytkownika u_A
$\mathcal{T}$	zbiór n rankingów $\mathcal{T} = \{\tau^1, \tau^2, \ldots, \tau^n\}$
$\mathcal{T}_A$	zbiór rankingów, które zostały zarekomendowane aktywnemu użytkownikowi u_A, przez algorytmy w zbiorze S
N_A	zbiór użytkowników, wyznaczonych jako sąsiedztwo aktywnego użytkownika u_A, przy pomocy algorytmu kNN
$\mathcal{T}_{N_A}$	zbiór rankingów, które zostały zarekomendowane przez algorytmy w zbiorze S, użytkownikom w zbiorze N_A
w	wektor, który jest również osobnikiem w algorytmie DE
w_{u_A}	wektor preferencji aktywnego użytkownika u_A
P	populacja algorytmu DE
$\phi(\mathcal{T}_A, \mathcal{T}_{N_A}, I)$	jest to reprezentacja wektorowa przedmiotów $x_j \in I$, która powstała na bazie wartości, które przedmioty te mają przyporządkowane w rankingach znajdujących się w zbiorach $\mathcal{T}_A$ oraz $\mathcal{T}_{N_A}$
$\phi(U)$	jest to reprezentacja wektorowa wszystkich użytkowników w zbiorze U

Dodatek A

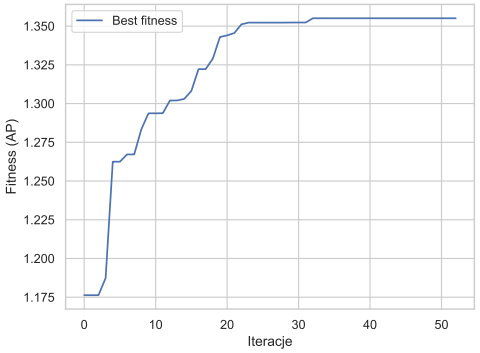
Dodatek A



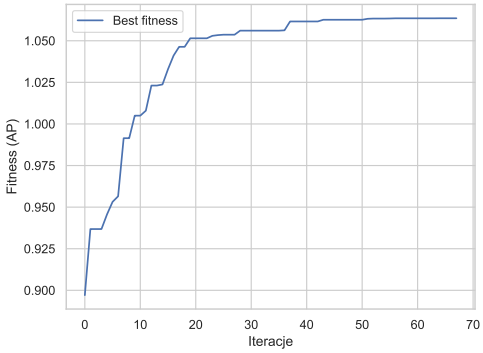
(A) Użytkownik 1



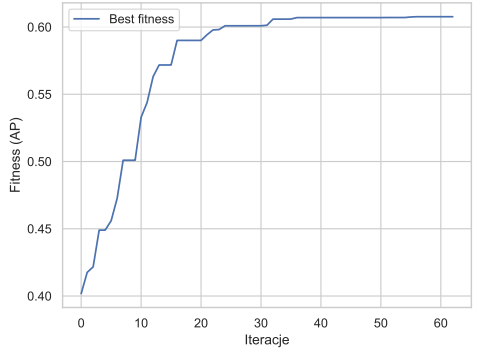
(B) Użytkownik 2



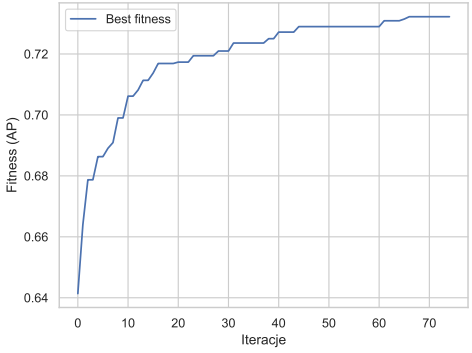
(C) Użytkownik 3



(D) Użytkownik 4

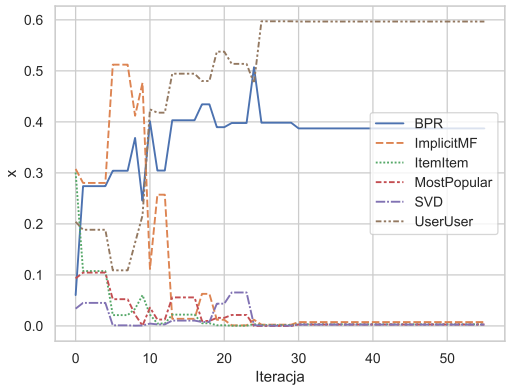


(E) Użytkownik 5

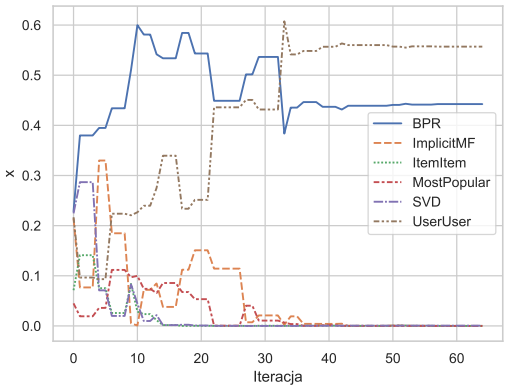


(F) Użytkownik 6

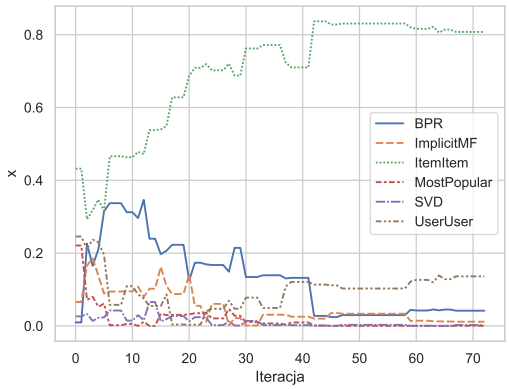
Rysunek A.1: Wykresy zbieżności, przedstawiające proces poszukiwania wektora preferencji przez algorytm *EAR* dla 6 wybranych użytkowników



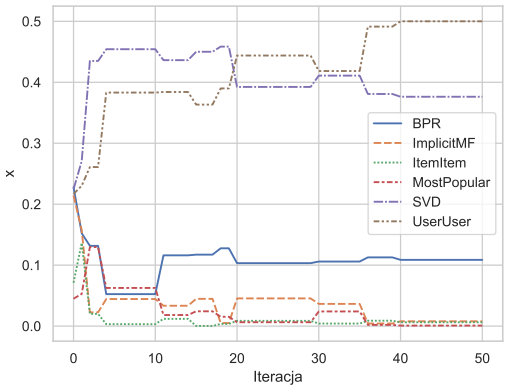
(A) Użytkownik 1



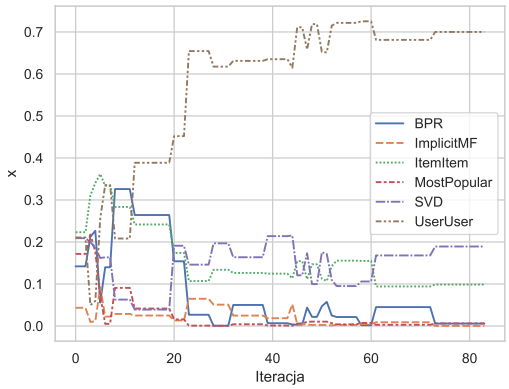
(B) Użytkownik 2



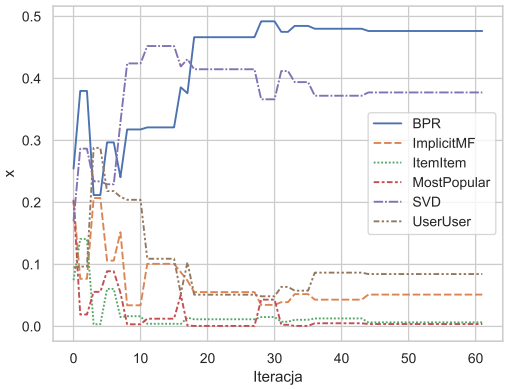
(C) Użytkownik 3



(D) Użytkownik 4

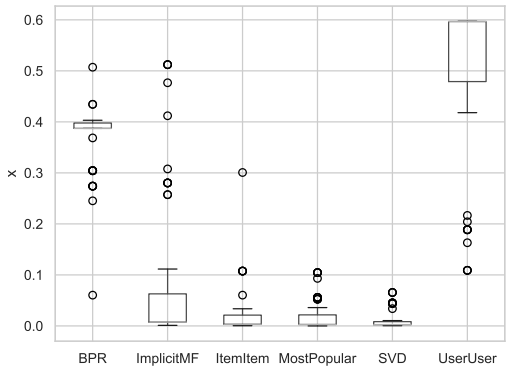


(E) Użytkownik 5

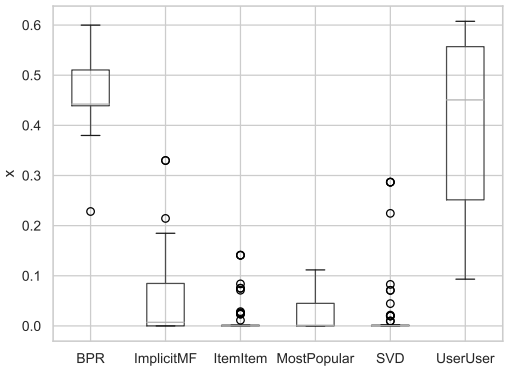


(F) Użytkownik 6

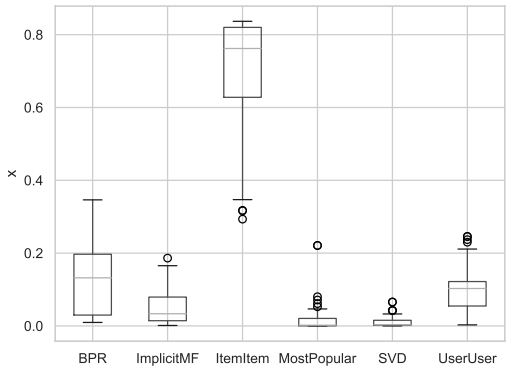
Rysunek A.2: Zmiana wartości wag w wektorze preferencji, podczas procesu jego poszukiwania przez algorytm *EAR* dla 6 wybranych użytkowników



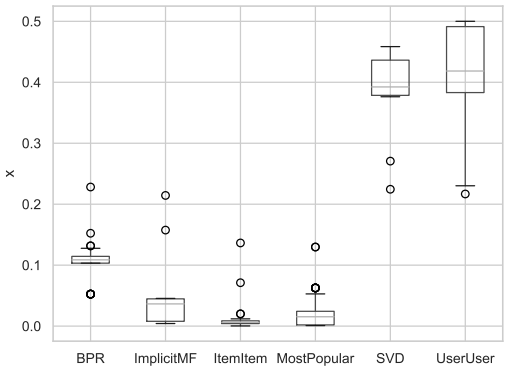
(A) Użytkownik 1



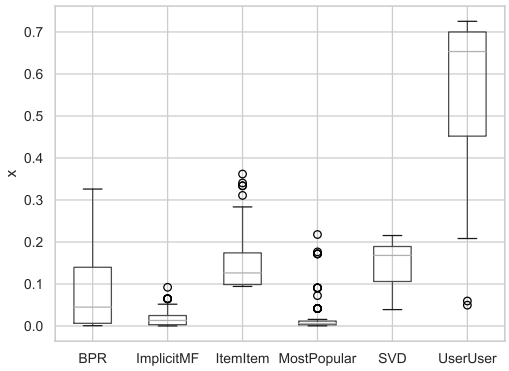
(B) Użytkownik 2



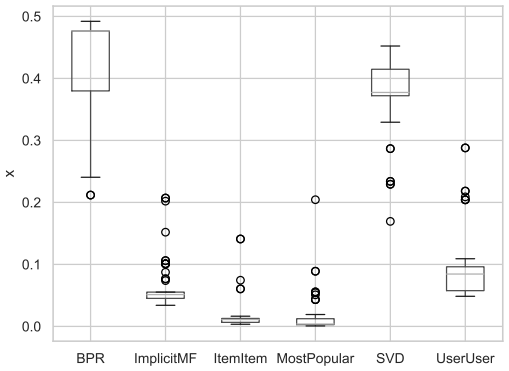
(C) Użytkownik 3



(D) Użytkownik 4

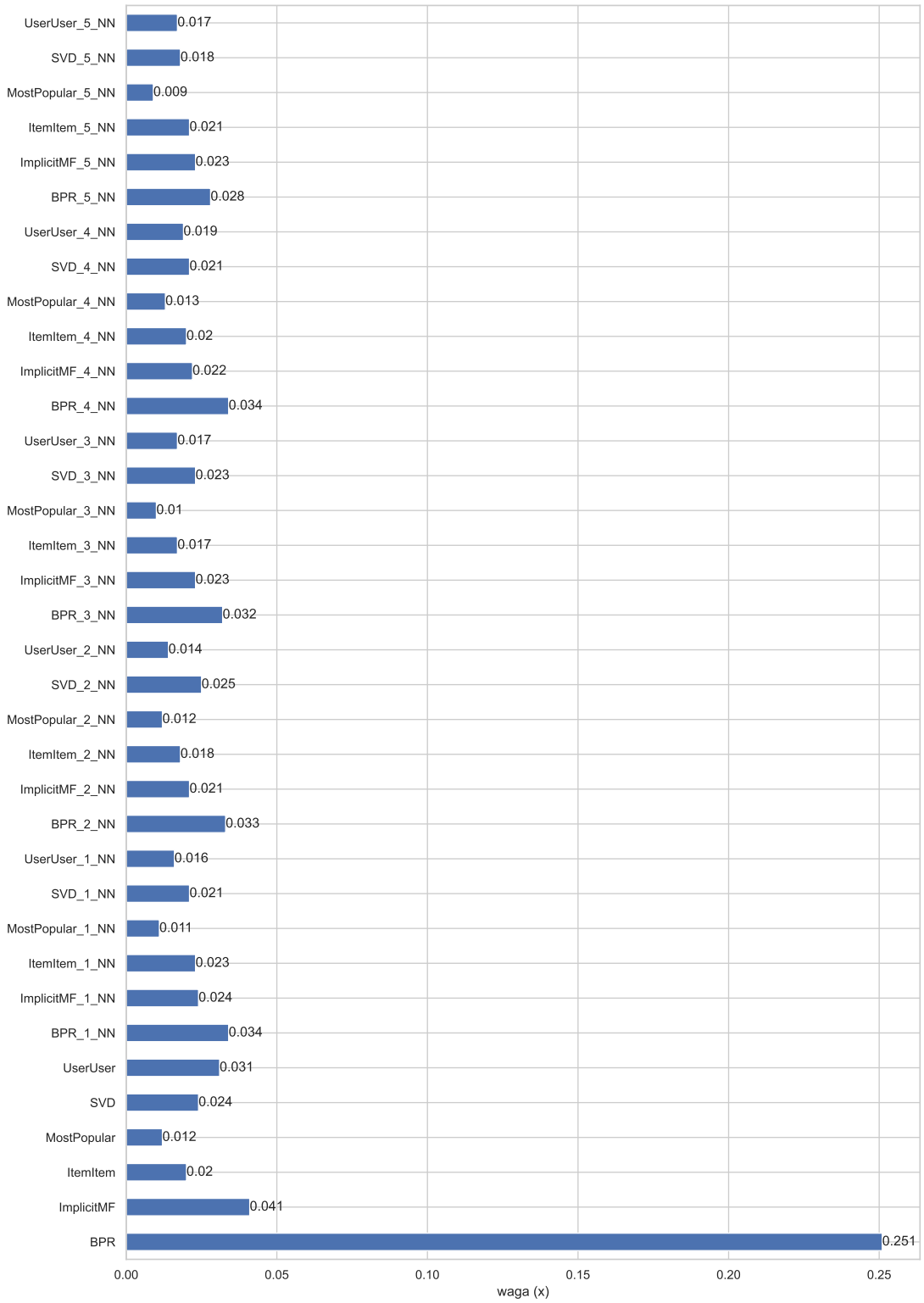


(E) Użytkownik 5

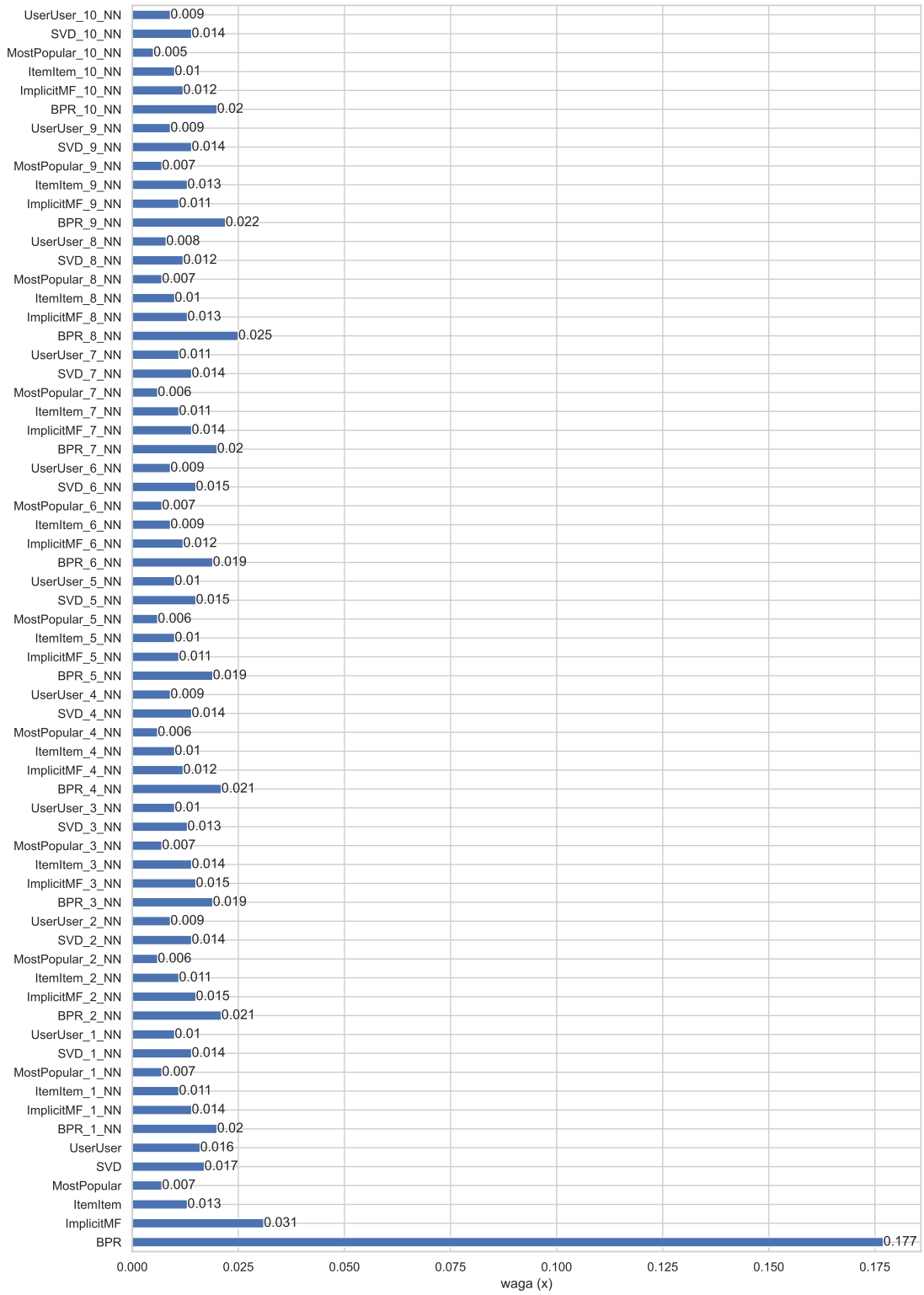


(F) Użytkownik 6

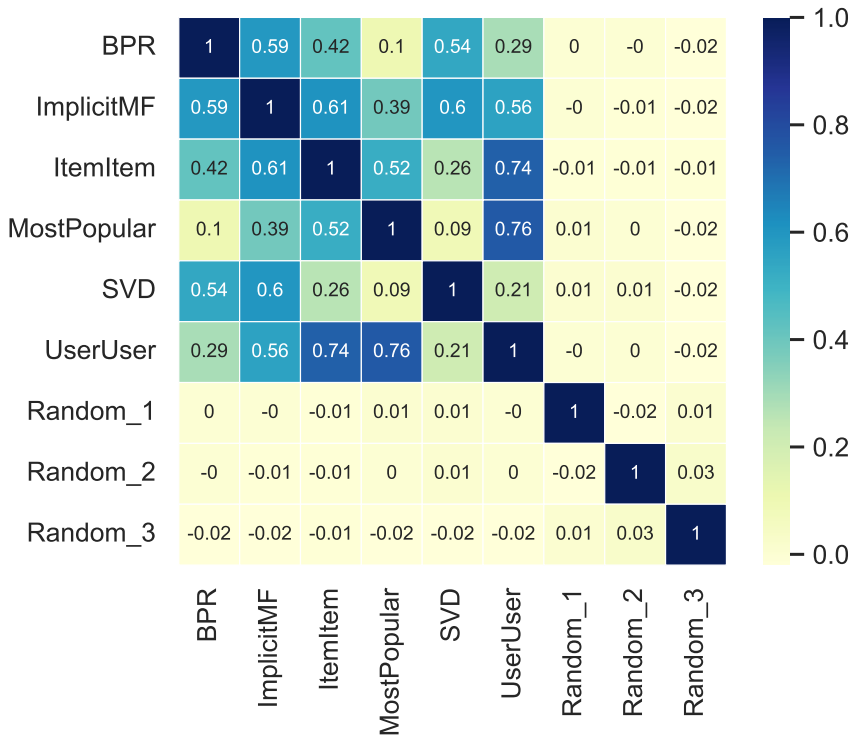
Rysunek A.3: Wykresy pudełkowe, przedstawiające rozkład wartości wektorów preferencji (wag), zaprezentowanych na rysunku A.2



Rysunek A.4: Wartości wag (w wektorze preferencji), które zostały przyporządkowane przez algorytm *EAR*. Na rysunku uwzględniono również wagi, które zostały przyporządkowane rekomendacjom wygenerowanym dla najbliższych sąsiadów (oznaczonych jako *NN*) użytkownika u_A (wariant $\text{EAR}_{\text{DE}_{\text{F3(5)}}}$)



Rysunek A.5: Wartości wag (w wektorze preferencji), które zostały przyporządkowane przez algorytm *EAR*. Na rysunku uwzględniono również wagi, które zostały przyporządkowane rekomendacjom wygenerowanym dla najbliższych sąsiadów (oznaczonych jako *NN*) użytkownika u_A (wariant $\text{EAR}_{\text{DE}_{\text{F3(10)}}}$)



Rysunek A.6: Macierz korelacji metod agregujących po uwzględnieniu algorytmów: *Random_1*, *Random_2*, *Random_3*, które generowały losowe rekomendacje. Korelacja pomiędzy rankingami została obliczona przy pomocy korelacji Tau Kendalla

Bibliografia

[1] A. Abbas, L. Zhang, S. U. Khan. A survey on context-aware recommender systems based on computational intelligence techniques. *Computing*, wolumen 97(7), 2015, strony 667–690.

[2] M. Abdel-Basset, L. Abdel-Fatah, A. K. Sangaiah. Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review. *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Intelligent Data-Centric Systems, Academic Press, 2018, strony 185–231.

[3] H. Abdollahpouri, R. Burke, B. Mobasher. Controlling Popularity Bias in Learning-to-Rank Recommendation. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, Como, Italy: Association for Computing Machinery, 2017, strony 42–46.

[4] G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin. Context-Aware Recommender Systems. *AI Magazine*, wolumen 32, 2011, strony 67–80.

[5] C. C. Aggarwal. Advanced Topics in Recommender Systems. *Recommender Systems: The Textbook*, Cham: Springer International Publishing, 2016, strony 411–448.

[6] C. C. Aggarwal. An Introduction to Recommender Systems. *Recommender Systems: The Textbook*, Cham: Springer International Publishing, 2016, strony 1–28.

[7] C. C. Aggarwal. Evaluating Recommender Systems. *Recommender Systems: The Textbook*, Cham: Springer International Publishing, 2016, strony 225–254.

[8] C. C. Aggarwal. Knowledge-Based Recommender Systems. *Recommender Systems: The Textbook*, Cham: Springer International Publishing, 2016, strony 167–197.

[9] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama. Optuna: A Next-Generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, strony 2623–2631.

[10] X. Amatriain, J. Basilico. Recommender Systems in Industry: A Netflix Case Study. *Recommender Systems Handbook*, Boston, MA: Springer US, 2015, strony 385–419.

[11] D. Anand, K. K. Bharadwaj. Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. *Expert Systems with Applications*, wolumen 38(5), 2011, strony 5101–5109.

[12] V. W. Anelli, A. Bellogín, T. Di Noia, D. Jannach, C. Pomo. Top-N Recommendation Algorithms: A Quest for the State-of-the-Art. *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '22, Barcelona, Spain: Association for Computing Machinery, 2022, strony 121–131.

[13] J. Arabas. Wykłady z algorytmów ewolucyjnych. Wydawnictwo Naukowo-Techniczne, 2004.

[14] R. Armstrong. The Long Tail: Why the Future of Business is Selling Less of More. *Canadian Journal of Communication*, wolumen 33, 2008.

[15] K. J. Arrow. Social Choice and Individual Values. New York, NY, USA: Wiley: New York, 1951.

[16] L. Baltrunas, T. Makcinskas, F. Ricci. Group Recommendations with Rank Aggregation and Collaborative Filtering. *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, Barcelona, Spain: Association for Computing Machinery, 2010, strony 119–126.

[17] E. Bassani. ranx: A Blazing-Fast Python Library for Ranking Evaluation and Comparison. *ECIR (2)*, wolumen 13186, Lecture Notes in Computer Science, Springer, 2022, strony 259–264.

[18] Z. Batmaz, A. Yurekli, A. Bilge, C. Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, wolumen 52, 2018, strony 1–37.

[19] D. Bawden, L. Robinson. Information Overload: An Overview. *Oxford Encyclopedia of Political Decision Making*, Oxford: Oxford University Press, 2020.

[20] M. Bałchanowski, U. Boryczka. A Comparative Study of Rank Aggregation Methods in Recommendation Systems. *Entropy*, wolumen 25(1), 132, 2023.

[21] M. Bałchanowski, U. Boryczka. Aggregation of Rankings Using Metaheuristics in Recommendation Systems. *Electronics*, wolumen 11(3), 369, 2022.

[22] M. Bałchanowski, U. Boryczka. Collaborative Rank Aggregation in Recommendation Systems. *Procedia Computer Science*, wolumen 207, 2022, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES 2022, strony 2213–2222.

[23] R. M. Bell, Y. Koren, C. Volinsky. All Together Now: A Perspective on the Netflix Prize. *CHANCE*, wolumen 23(1), 2010, strony 24–29.

[24] R. M. Bell, Y. Koren, C. Volinsky. The BellKor solution to the Netflix Prize. Technical report, AT&T Labs, 2007.

[25] A. Bellet, A. Habrard, M. Sebban. Metric learning. *Synthesis lectures on artificial intelligence and machine learning*, wolumen 9(1), 2015, strony 1–151.

[26] J. Bennett, S. Lanning, N. Netflix. The Netflix Prize. *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[27] I. Beregovskaya, M. Koroteev. Review of Clustering-Based Recommender Systems. *arXiv preprint arXiv:2109.12839*, 2021.

[28] J. Bergstra, D. Yamins, D. D. Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, Atlanta, GA, USA: JMLR.org, 2013, strony 115–123.

[29] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl. Algorithms for Hyper-Parameter Optimization. *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, Granada, Spain: Curran Associates Inc., 2011, strony 2546–2554.

[30] J. Blank, K. Deb. pymoo: Multi-Objective Optimization in Python. *IEEE Access*, wolumen 8, 2020, strony 89497–89509.

[31] M. Blondel, O. Teboul, Q. Berthet, J. Djolonga. Fast Differentiable Sorting and Ranking. *Proceedings of the 37th International Conference on Machine Learning*, wolumen 119, Proceedings of Machine Learning Research, PMLR, 2020, strony 950–959.

[32] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, wolumen 26, 2012, strony 225–238.

[33] D. Bollegala, N. Noman, H. Iba. RankDE: Learning a Ranking Function for Information Retrieval Using Differential Evolution. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, Dublin, Ireland: Association for Computing Machinery, 2011, strony 1771–1778.

[34] L. Boratto, S. Carta. State-of-the-Art in Group Recommendation and New Approaches for Automatic Identification of Groups. *Information Retrieval and Mining in Distributed Environments*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, strony 1–20.

[35] J.-C. de Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[36] U. Boryczka, M. Bałchanowski. Speed up Differential Evolution for ranking of items in recommendation systems. *Procedia Computer Science*, wolumen 192, 2021, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021, strony 2229–2238.

[37] U. Boryczka, M. Bałchanowski. Using Differential Evolution in order to create a personalized list of recommended items. *Procedia Computer Science*, wolumen 176, 2020, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020, strony 1940–1949.

[38] U. Boryczka, P. Juszczuk, L. Kłosowicz. A Comparative Study of Various Strategies in Differential Evolution. *Evolutionary Computing and Global Optimization*, KAEiOG'09, 2009, strony 19–26.

[39] J. S. Breese, D. Heckerman, C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, strony 43–52.

[40] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, wolumen 10(6), 2006, strony 646–657.

[41] A. Brown, W. Xie, V. Kalogeiton, A. Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. *European Conference on Computer Vision*, Springer, 2020, strony 677–694.

[42] D. J. Brown. Aggregation of Preferences. *The Quarterly Journal of Economics*, wolumen 89(3), 1975, strony 456–469.

[43] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender. Learning to Rank Using Gradient Descent. *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, Bonn, Germany: Association for Computing Machinery, 2005, strony 89–96.

[44] C. J. C. Burges, R. Ragno, Q. V. Le. Learning to Rank with Nonsmooth Cost Functions. *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, Canada: MIT Press, 2006, strony 193–200.

[45] E. Çano, M. Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, wolumen 21(6), 2017, strony 1487–1524.

[46] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, Corvalis, Oregon, USA: Association for Computing Machinery, 2007, strony 129–136.

[47] P. Castells, N. J. Hurley, S. Vargas. Novelty and Diversity in Recommender Systems. *Recommender Systems Handbook*, Boston, MA: Springer US, 2015, strony 881–918.

[48] P. Cichosz. Uczenie się aproksymacji funkcji. *Systemy uczące się*, Wydawnictwa Naukowo-Techniczne, 2000, strony 432–492.

[49] O. Cordón, E. Herrera-Viedma, C. López-Pujalte, M. Luque, C. Zarco. A review on the application of evolutionary computation to information retrieval. *International Journal of Approximate Reasoning*, wolumen 34(2), 2003, Soft Computing Applications to Intelligent Information Retrieval on the Internet, strony 241–264.

[50] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Algorytmy aproksymacyjne. *Wprowadzenie do algorytmów*, Wydawnictwo Naukowe PWN, 2012, strony 1131–1167.

[51] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. NP-zupełność. *Wprowadzenie do algorytmów*, Wydawnictwo Naukowe PWN, 2012, strony 1073–1130.

[52] K. Crammer, Y. Singer. Pranking with Ranking. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, Vancouver, British Columbia, Canada: MIT Press, 2001, strony 641–647.

[53] P. Cremonesi, D. Jannach. Progress in Recommender Systems Research: Crisis? What Crisis?, *AI Magazine*, wolumen 42, 2022, strony 43–54.

[54] M. F. Dacrema, S. Boglio, P. Cremonesi, D. Jannach. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems*, wolumen 39(2), 2021, strony 1–49.

[55] C. Darwin. On the Origin of Species. London: John Murray, 1859.

[56] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, D. Sampath. The YouTube Video Recommendation System. *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, Barcelona, Spain: Association for Computing Machinery, 2010, strony 293–296.

[57] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, M. Quadrana. Content-Based Video Recommendation System Based on Stylistic Visual Features. *Journal on Data Semantics*, wolumen 5, 2016, strony 1–15.

[58] M. Deshpande, G. Karypis. Item-Based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.*, wolumen 22(1), 2004, strony 143–177.

[59] C. Desrosiers, G. Karypis. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. *Recommender Systems Handbook*, 2011, strony 107–144.

[60] Q. Dong, Q. Yuan, Y.-B. Shi. Alleviating the recommendation bias via rank aggregation. *Physica A: Statistical Mechanics and its Applications*, wolumen 534, 2019.

[61] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma. A survey on ensemble learning. *Frontiers of Computer Science*, wolumen 14(2), 2020, strony 241–258.

[62] M. Dorigo. "Optimization, learning and natural algorithms", prac. dokt., Politecnico di Milano, Włochy, 1992.

[63] C. Dwork, R. Kumar, M. Naor, D. Sivakumar. Rank Aggregation Methods for the Web. *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, Hong Kong, Hong Kong: Association for Computing Machinery, 2001, strony 613–622.

[64] M. D. Ekstrand. LensKit for Python: Next-Generation Software for Recommender Systems Experiments. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, Virtual Event, Ireland: Association for Computing Machinery, 2020, strony 2999–3006.

[65] M. D. Ekstrand, J. T. Riedl, J. A. Konstan. Collaborative Filtering Recommender Systems. *Found. Trends Hum.-Comput. Interact.*, wolumen 4(2), 2011, strony 81–173.

[66] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee. Comparing Partial Rankings. *SIAM Journal on Discrete Mathematics*, wolumen 20(3), 2006, strony 628–648.

[67] K. Falk. Rangowanie i nauka rangowania. *Praktyczne systemy rekomendacji*, Wydawnictwo Naukowe PWN, 2020, strony 385–414.

[68] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, R. Kashef. Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Applied Sciences*, wolumen 10(21), 7748, 2020.

[69] V. Feoktistov. Differential Evolution. *Differential Evolution: In Search of Solutions*, Boston, MA: Springer US, 2006, strony 1–24.

[70] E. A. Fox, J. A. Shaw. Combination of Multiple Searches. *TREC*, wolumen 500-215, NIST Special Publication, National Institute of Standards i Technology (NIST), 1993, strony 243–252.

[71] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer. An Efficient Boosting Algorithm for Combining Preferences. *J. Mach. Learn. Res.*, wolumen 4, 2003, strony 933–969.

[72] A. Friedman, B. P. Knijnenburg, K. Vanhecke, L. Martens, S. Berkovsky. Privacy Aspects of Recommender Systems. *Recommender Systems Handbook*, Boston, MA: Springer US, 2015, strony 649–688.

[73] N. Fuhr. Probabilistic models in information retrieval. *The computer journal*, wolumen 35(3), 1992, strony 243–255.

[74] W. V. Gehrlein. The Condorcet criterion and committee selection. *Mathematical Social Sciences*, wolumen 10(3), 1985, strony 199–209.

[75] A. Guimarães, T. F. Costa, A. Lacerda, G. L. Pappa, N. Ziviani. Guard: a genetic unified approach for recommendation. *Journal of Information and Data Management*, wolumen 4(3), 2013, strony 295–295.

[76] N. Halko, P. G. Martinsson, J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, wolumen 53(2), 2011, strony 217–288.

[77] F. M. Harper, J. A. Konstan. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, wolumen 5(4), 2015.

[78] F. Hdioud, B. Frikh, B. Ouhbi, I. Khalil. Multi-Criteria Recommender Systems: A Survey and a Method to Learn New User's Profile. *International Journal of Mobile Computing and Multimedia Communications*, wolumen 8, 2017, strony 20–48.

[79] M. Held, A. J. Hoffman, E. L. Johnson, P. Wolfe. Aspects of the traveling salesman problem. *IBM Journal of Research and Development*, wolumen 28(4), 1984, strony 476–486.

[80] J. L. Herlocker, J. A. Konstan, J. Riedl. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information Retrieval*, wolumen 5, 2004, strony 287–310.

[81] Y. Ho, S. Fong, Z. Yan. A Hybrid GA-based Collaborative Filtering Model for Online Recommenders. *ICE-B*, 2007, strony 200–203.

[82] T. Horváth, A. C. Carvalho. Evolutionary Computing in Recommender Systems: A Review of Recent Research. *Natural Computing: An International Journal*, wolumen 16(3), 2017, strony 441–462.

[83] Y. Hu, Y. Koren, C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining*, 2008, strony 263–272.

[84] N. Idrissi, A. Zellou. A systematic literature review of sparsity issues in recommender systems. *Social Network Analysis and Mining*, wolumen 10, 2020, strony 1–23.

[85] A. Jameson, B. Smyth. Recommendation to Groups. *The Adaptive Web: Methods and Strategies of Web Personalization*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, strony 596–627.

[86] D. Jannach, M. Jugovac. Measuring the Business Value of Recommender Systems. *ACM Transactions on Management Information Systems*, wolumen 10(4), 2019, strony 1–23.

[87] K. Järvelin, J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, wolumen 20(4), 2002, strony 422–446.

[88] G. Jawaheer, M. Szomszor, P. Kostkova. Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, Barcelona, Spain: Association for Computing Machinery, 2010, strony 47–51.

[89] Z. Kang, C. Peng, Q. Cheng. Top-N Recommender System via Matrix Completion. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, Phoenix, Arizona: AAAI Press, 2016, strony 179–184.

[90] D. Karaboga. An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06. Sty. 2005.

[91] A. Karatzoglou, L. Baltrunas, Y. Shi. Learning to Rank for Recommender Systems. *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, Hong Kong, China: Association for Computing Machinery, 2013, strony 493–494.

[92] A. Karpus, I. Vagliano, K. Goczyła, M. Morisio. An Ontology-based contextual prefiltering technique for Recommender Systems. *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016, strony 411–420.

[93] M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, wolumen 30(1/2), 1938, strony 81–93.

[94] J. Kennedy, R. Eberhart. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, wolumen 4, 1995, strony 1942–1948 vol.4.

[95] S. Khatwani, M. Chandak. Building Personalized and Non Personalized recommendation systems. *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, 2016, strony 623–628.

[96] K.-j. Kim, H. Ahn. Using a Clustering Genetic Algorithm to Support Customer Segmentation for Personalized Recommender Systems. *Artificial Intelligence and Simulation*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, strony 409–415.

[97] S. Kirkpatrick, C. Gelatt, M. Vecchi. Optimization by Simulated Annealing. *Science (New York, N.Y.)*, wolumen 220, 1983, strony 671–80.

[98] Y. Koren, R. Bell, C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, wolumen 42(8), 2009, strony 30–37.

[99] Y. Koren. Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Trans. Knowl. Discov. Data*, wolumen 4(1), 2010.

[100] J. Kozak, B. Probierz, K. Kania, P. Juszczuk. Preference-Driven Classification Measure. *Entropy*, wolumen 24(4), 531, 2022.

[101] M. Kuanr, P. Mohapatra. Recent Challenges in Recommender Systems: A Survey. *Progress in Advanced Computing and Intelligent Engineering*, Singapore: Springer Singapore, 2021, strony 353–365.

[102] U. Kuzelewska. Clustering algorithms in hybrid recommender system on movielens data. *Studies in logic, grammar and rhetoric*, wolumen 37(1), 2014, strony 125–139.

[103] P. B. Lamere. I've Got 10 Million Songs in My Pocket: Now What?, *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, Dublin, Ireland: Association for Computing Machinery, 2012, strony 207–208.

[104] S. Lestari, T. Adji, A. Permanasari. WP-Rank: Rank aggregation based collaborative filtering method in recommender system. *International Journal of Engineering and Technology(UAE)*, wolumen 7, 2018, strony 193–197.

[105] H. Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, wolumen 94(10), 2011, strony 1854–1862.

[106] P. Li, C. J. C. Burges, Q. Wu. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, Vancouver, British Columbia, Canada: Curran Associates Inc., 2007, strony 897–904.

[107] X. Li, X. Wang, G. Xiao. A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications. *Briefings in Bioinformatics*, wolumen 20(1), sierp. 2017, strony 178–189.

[108] H. Liang, X. Sun, Y. Sun, Y. Gao. Text feature extraction based on deep learning: a review. *Eurasip Journal on Wireless Communications and Networking*, wolumen 2017, 2017.

[109] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, wolumen 41(4 part 2), 2014, strony 2065–2073.

[110] G. Linden, B. Smith, J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, wolumen 7(1), 2003, strony 76–80.

[111] G. Lissowski. Wprowadzenie. *Studia Socjologiczne 2000 nr 1-2/2000*, Wydawnictwo Instytutu Filozofii i Socjologii PAN, 2000, strony 9–14.

[112] C. List. Social Choice Theory. *The Stanford Encyclopedia of Philosophy*, Spring 2022, Metaphysics Research Lab, Stanford University, 2022.

[113] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, H. Li. Supervised Rank Aggregation. *Proceedings of the 16th International Conference on World Wide Web*, Banff, Alberta, Canada: Association for Computing Machinery, 2007, strony 481–490.

[114] T. Loukil, J. Teghem, D. Tuyttens. Solving multi-objective production scheduling problems using metaheuristics. *European journal of operational research*, wolumen 161(1), 2005, strony 42–61.

[115] R. Meena, K. K. Bharadwaj. Group Recommender System Based on Rank Aggregation – An Evolutionary Approach. *Mining Intelligence and Knowledge Exploration*, Cham: Springer International Publishing, 2013, strony 663–676.

[116] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, wolumen 21(6), 1953, strony 1087–1092.

[117] E. Mezura-Montes, J. Velázquez-Reyes, C. A. Coello Coello. A comparative study of differential evolution variants for global optimization. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, strony 485–492.

[118] K. Miyahara, M. J. Pazzani. Collaborative Filtering with the Simple Bayesian Classifier. *PRICAI 2000 Topics in Artificial Intelligence*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, strony 679–689.

[119] M. H. Mohamed, M. H. Khafagy, M. H. Ibrahim. Recommender Systems Challenges and Solutions Survey. *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2019, strony 149–155.

[120] M. Molga, C. Smutnicki. Test functions for optimization needs. *Manuscript*, wolumen 101, 2005, strony 48.

[121] T. Morzy. Kombinacja klasyfikatorów. *Eksploracja danych. Metody i algorytmy*, Wydawnictwo Naukowe PWN, 2013, strony 313–345.

[122] N. Nanas, A. Roeck. A Review of Evolutionary and Immune-Inspired Information Filtering. *Natural Computing: An International Journal*, wolumen 9(3), 2010, strony 545–573.

[123] D. Oard, J. Kim. Implicit Feedback for Recommender Systems. *in Proceedings of the AAAI Workshop on Recommender Systems*, 1998, strony 81–83.

[124] S. Oliveira, V. Diniz, A. Lacerda, G. L. Pappa. Evolutionary rank aggregation for recommender systems. *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, strony 255–262.

[125] S. Oliveira, V. Diniz, A. Lacerda, G. L. Pappa. Multi-objective Evolutionary Rank Aggregation for Recommender Systems. *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, strony 1–8.

[126] S. E. L. Oliveira, V. Diniz, A. Lacerda, L. Merschmanm, G. L. Pappa. Is Rank Aggregation Effective in Recommender Systems? An Experimental Analysis. *ACM Trans. Intell. Syst. Technol.*, wolumen 11(2), 2020.

[127] S. Ç. Onar, B. Öztayşi, C. Kahraman, S. Yanık, Ö. Şenvar. A Literature Survey on Metaheuristics in Production Systems. *Metaheuristics for Production Systems*, Cham: Springer International Publishing, 2016, strony 1–24.

[128] K. Opara, J. Arabas. Comparison of mutation strategies in Differential Evolution – A probabilistic perspective. *Swarm and Evolutionary Computation*, wolumen 39, 2018, strony 53–69.

[129] K. R. Opara, J. Arabas. Differential Evolution: A survey of theoretical analyses. *Swarm and Evolutionary Computation*, wolumen 44, 2019, strony 546–558.

[130] M. G. Ozsoy, F. Polat. Trust based recommendation systems. *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, 2013, strony 1267–1274.

[131] D. Panda, S. Ray. Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. *Journal of Intelligent Information Systems*, 2022, strony 1–26.

[132] M. J. Pazzani, D. Billsus. Content-Based Recommendation Systems. *The Adaptive Web: Methods and Strategies of Web Personalization*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, strony 325–341.

[133] L. Peska, P. Vojtas. Off-Line vs. On-Line Evaluation of Recommender Systems in Small E-Commerce. *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, HT '20, Virtual Event, USA: Association for Computing Machinery, 2020, strony 291–300.

[134] T. Pessemier, S. Dooms, L. Martens. Comparison of Group Recommendation Algorithms. *Multimedia Tools Appl.*, wolumen 72(3), 2014, strony 2497–2541.

[135] G. Piatetsky-Shapiro. Interview with Simon Funk. *SIGKDD Explorations*, wolumen 9, 2007, strony 38–40.

[136] A. Pujahari, D. S. Sisodia. Aggregation of preference relations to enhance the ranking quality of collaborative filtering based group recommender system. *Expert Systems with Applications*, wolumen 156, 2020.

[137] T. Rambharose, A. Nikov. Computational Intelligence-Based Personalization of Interactive Web Systems. *WSEAS Trans. Info. Sci. and App.*, wolumen 7(4), 2010, strony 484–497.

[138] A. M. Rashid, G. Karypis, J. Riedl. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.*, wolumen 10(2), 2008, strony 90–100.

[139] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, Montreal, Quebec, Canada: AUAI Press, 2009, strony 452–461.

[140] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, Chapel Hill, North Carolina, USA: Association for Computing Machinery, 1994, strony 175–186.

[141] M. T. Ribeiro, N. Ziviani, E. S. D. Moura, I. Hata, A. Lacerda, A. Veloso. Multiobjective Pareto-Efficient Approaches for Recommender Systems. *ACM Trans. Intell. Syst. Technol.*, wolumen 5(4), 2015.

[142] F. Ricci, L. Rokach, B. Shapira. Recommender Systems: Introduction and Challenges. *Recommender Systems Handbook*, Boston, MA: Springer US, 2015, strony 1–34.

[143] H. Robbins, S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, wolumen 22(3), 1951, strony 400–407.

[144] S. Sagiroglu, D. Sinanc. Big data: A review. *2013 International Conference on Collaboration Technologies and Systems (CTS)*, 2013, strony 42–47.

[145] M. Salehi, I. N. Kamalabadi, M. B. Ghaznavi-Ghoushchi et al. Attribute-based collaborative filtering using genetic algorithm and weighted c-means algorithm. *International Journal of business information systems*, wolumen 13(3), 2013, strony 265–283.

[146] M. Salehi, M. Pourzaferani, S. A. Razavi. Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model. *Egyptian Informatics Journal*, wolumen 14(1), 2013, strony 67–78.

[147] B. Sarwar, G. Karypis, J. Konstan, J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, Hong Kong, Hong Kong: Association for Computing Machinery, 2001, strony 285–295.

[148] B. M. Sarwar, G. Karypis, J. Konstan, J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of the fifth international conference on computer and information technology*, wolumen 1, 2002, strony 291–324.

[149] A. Schein, A. Popescul, L. Ungar, D. Pennock. Methods and Metrics for Cold-Start Recommendations. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 2002, strony 253–260.

[150] G. Schröder, M. Thiele, W. Lehner. Setting goals and choosing metrics for recommender system evaluations. *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, wolumen 23, 2011, strony 53.

[151] O. Senvar, E. Turanoğlu Bekar, C. Kahraman. Usage of Metaheuristics in Engineering: A Literature Review. *Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance*, 2012, strony 484–528.

[152] P. Shah, R. Sekhar, A. Kulkarni, P. Siarry. Metaheuristic Algorithms in Industry 4.0. Advances in Metaheuristics, CRC Press, 2021.

[153] G. Shani, A. Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, Springer, 2011, strony 257–297.

[154] A. Shashua, A. Levin. Ranking with Large Margin Principle: Two Approaches. *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, Cambridge, MA, USA: MIT Press, 2002, strony 961–968.

[155] S. Sidana, M. Trofimov, O. Horodnitskii, C. Laclau, Y. Maximov, M.-R. Amini. Representation Learning and Pairwise Ranking for Implicit Feedback in Recommendation Systems. 2017, arXiv: 1705.00105 [stat.ML].

[156] E. Q. da Silva, C. G. Camilo, L. M. L. Pascoal, T. C. Rosa. An evolutionary approach for combining results of recommender systems techniques based on Collaborative Filtering. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, strony 959–966.

[157] S. F. Smith. "A Learning System Based on Genetic Adaptive Algorithms", prac. dokt., University of Pittsburgh, USA, 1980.

[158] B. Smyth, P Cotter. Personalized TV listings service for the digital TV age. *Knowledge-Based Systems*, wolumen 13, 2000, strony 53–59.

[159] H. Steck. Evaluation of Recommendations: Rating-Prediction and Ranking. *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, Hong Kong, China: Association for Computing Machinery, 2013, strony 213–220.

[160] R. Storn, K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization*, wolumen 11(4), 1997, strony 341–359.

[161] X. Su, T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Adv. in Artif. Intell.*, wolumen 2009, 2009.

[162] Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang, H. Yan. Recommender systems based on social networks. *Journal of Systems and Software*, wolumen 99, 2015, strony 109–119.

[163] Z. Sun, D. Yu, H. Fang, J. Yang, X. Qu, J. Zhang, C. Geng. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, Virtual Event, Brazil: Association for Computing Machinery, 2020, strony 23–32.

[164] K. Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, wolumen 22(1), 2015, strony 3–18.

[165] F. Tahmasei, M. Meghdadi, S. Ahmadian, K. Valiallahi. A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. *Multimedia Tools and Applications*, wolumen 80, 2021.

[166] M. Taifi. Senior data science platform engineer blog. https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832, dostęp: 1-09-2022.

[167] Y. Tang, Q. Tong. BordaRank: A ranking aggregation based approach to collaborative filtering. *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, strony 1–6.

[168] R. Torres, S. McNee, M. Abel, J. Konstan, J. Riedl. Enhancing digital libraries with TechLens. *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.* 2004, strony 228–236.

[169] B. Twardowski, P. Zawistowski, S. Zaborowski. Metric Learning for Session-Based Recommendations. *Advances in Information Retrieval*, Springer International Publishing, 2021, strony 650–665.

[170] S. Ujjin, P. J. Bentley. Learning user preferences using evolution. *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Singapore*, 2002.

[171] S. Ujjin, P. J. Bentley. Particle swarm optimization recommender system. *2003 IEEE Swarm Intelligence Symposium, SIS 2003, Indianapolis, IN, USA, April 24-26, 2003*, IEEE, 2003, strony 124–131.

[172] D. Vanderpooten, M. Farah. An outranking approach for rank aggregation in information retrieval. 2007.

[173] O. Velez-Langs, A. de Antonio. Learning User's Characteristics in Collaborative Filtering through Genetic Algorithms: Some New Results. *Studies in Fuzziness and Soft Computing*, wolumen 312, 2014, strony 309–326.

[174] J. Wang, B. Sarwar, N. Sundaresan. Utilizing Related Products for Post-Purchase Recommendation in e-Commerce. *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, Chicago, Illinois, USA: Association for Computing Machinery, 2011, strony 329–332.

[175] S. Wang, M. Gong, L. Ma, Q. Cai, L. Jiao. Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, strony 672–679.

[176] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, D. Lian. A Survey on Session-Based Recommender Systems. *ACM Comput. Surv.*, wolumen 54(7), 2021.

[177] A. Wani, I. Joshi, S. Khandve, V. Wagh, R. Joshi. Evaluating Deep Learning Approaches for Covid19 Fake News Detection. *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, Springer International Publishing, 2021, strony 153–163.

[178] S. Wierzchoń, M. Kłopotek. Algorytmy kombinatorycznej analizy skupień. *Algorytmy analizy skupień*, Wydawnictwo Naukowe PWN, 2017, strony 88–165.

[179] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Breakthroughs in Statistics: Methodology and Distribution*, New York, NY: Springer New York, 1992, strony 196–202.

[180] D. Wolpert, W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, wolumen 1(1), 1997, strony 67–82.

[181] Q. Wu, C. J. Burges, K. M. Svore, J. Gao. Adapting Boosting for Information Retrieval Measures. *Inf. Retr.*, wolumen 13(3), 2010, strony 254–270.

[182] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, H. Li. Listwise Approach to Learning to Rank: Theory and Algorithm. *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, Helsinki, Finland: Association for Computing Machinery, 2008, strony 1192–1199.

[183] J. Xu, H. Li. AdaRank: A Boosting Algorithm for Information Retrieval. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, Amsterdam, The Netherlands: Association for Computing Machinery, 2007, strony 391–398.

[184] E. Yalcin, F. Ismailoglu, A. Bilge. An entropy empowered hybridized aggregation technique for group recommender systems. *Expert Systems with Applications*, wolumen 166, 2021.

[185] X.-S. Yang. A New Metaheuristic Bat-Inspired Algorithm. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, strony 65–74.

[186] X.-S. Yang. Harmony search as a metaheuristic algorithm. *Music-inspired harmony search algorithm*, Springer, 2009, strony 1–14.

[187] X.-S. Yang, S. Deb. Cuckoo Search via Lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, strony 210–214.

[188] H. P. Young, A. Levenglick. A Consistent Extension of Condorcet's Election Principle. *SIAM Journal on Applied Mathematics*, wolumen 35(2), 1978, strony 285–300.

[189] J. B. Yves Raimond. Netflix Technology Blog. https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429, dostęp: 1-09-2022.

[190] E. Zangerle, C. Bauer. Evaluating Recommender Systems: Survey and Framework. *ACM Comput. Surv.*, wolumen 55(8), 2022.

[191] F. Zhang, T. Gong, V. E. Lee, G. Zhao, C. Rong, G. Qu. Fast algorithms to evaluate collaborative filtering recommender systems. *Knowledge-Based Systems*, wolumen 96, 2016, strony 96–103.

[192] Y. Zuo, M. Gong, J. Zeng, L. Ma, L. Jiao. Personalized Recommendation Based on Evolutionary Multi-Objective Optimization [Research Frontier]. *IEEE Computational Intelligence Magazine*, wolumen 10(1), 2015, strony 52–62.

[193] W. S. Zwicker, H. Moulin. Introduction to the Theory of Voting. *Handbook of Computational Social Choice*, Cambridge University Press, 2016, strony 23–56.

Spis rysunków

Spis tabel

Spis algorytmów