Uniwersytet im. Adama Mickiewicza w Poznaniu
Wydział Matematyki i Informatyki

# ROZPRAWA DOKTORSKA

Robert Kwieciński

## Modele rekomendacyjne wspólnej filtracji w serwisach ogłoszeniowych

Promotor
**prof. UAM dr hab. Tomasz Górecki**

Promotor pomocniczy
**prof. UEP dr hab. Agata Jolanta Filipowska**

Dyscyplina
**Informatyka**

Poznań, 2023

Dziękuję promotorom, którzy poprzez wiele inspirujących rozmów pokierowali moimi badaniami we właściwym kierunku oraz pomogli postawić pierwsze kroki w działalności naukowej.

Dziękuję współpracownikom z Grupy OLX, którzy aktywnie uczestniczyli we wdrażaniu badanych rozwiązań.

Dziękuję żonie za ogromną pomoc w jednoczesnym prowadzeniu badań i uczestnictwie w życiu rodzinnym, a także korektę gramatyczną rozprawy.

Dziękuję córeczce Alicji, która pojawiła się wśród nas w trakcie trwania doktoratu i motywowała mnie do jego planowego zakończenia.

Dziękuje rodzicom, którzy od najmłodszych lat wspierali moją pasję do nauk ścisłych.

Poznań, 3 grudnia 2023 r.

# Oświadczenie

Ja, niżej podpisany **Robert Kwieciński**, student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt. *Modele rekomendacyjne wspólnej filtracji w serwisach ogłoszeniowych* napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób. Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej. Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowi podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

[TAK] wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

[TAK] wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

# Streszczenie

W wielu praktycznych zastosowaniach liczba dostępnych produktów jest zbyt duża, by użytkownicy byli w stanie się z nimi zapoznać w czasie, jaki są skłonni na to przeznaczyć. Problem ten adresują systemy rekomendacyjne, których zadaniem jest wybór niewielkiej liczby przedmiotów w celu ich zaprezentowania poszczególnym użytkownikom. Wspomagając dokonanie wyboru przez użytkownika, systemy te mają duży wpływ na działalność serwisów ogłoszeniowych. Powszechnie stosowane są modele wspólnej filtracji, w których wykorzystywane są wyłącznie informacje o interakcjach między użytkownikami a przedmiotami. W rozprawie przedstawiamy wyniki badań dotyczących modeli wspólnej filtracji w serwisach ogłoszeniowych, na przykładzie serwisów Grupy OLX.

W prezentowanej pracy omawiamy szczególne cechy serwisów ogłoszeniowych mające wpływ na dobór metod rekomendacji oraz ewaluacji. Przedstawiamy wyniki ewaluacji offline dla istniejących modeli oraz pokazujemy przewagę modelu RP3Beta nad pozostałymi metodami. Ponadto weryfikujemy uzyskane wyniki w przeprowadzonych testach A/B z udziałem milionów użytkowników serwisów Grupy OLX.

Przedstawiamy infrastrukturę pozwalającą na generowanie rekomendacji w czasie rzeczywistym oraz pokazujemy możliwość jej zastosowania dla wielu istniejących modeli rekomendacji. Następnie proponujemy model RP3Beta real-time wykorzystujący tę infrastrukturę. Prezentujemy wyniki testów A/B pokazujących statystycznie istotną przewagę tego modelu nad modelem bazowym.

Wprowadzamy nową grafową metodę rekomendacji, P3LTR, będącą uogólnieniem modelu RP3Beta. Zaproponowana metoda, w przeciwieństwie do modelu RP3Beta, posiada parametry optymalizowane podczas trenowania modelu, a także umożliwia uwzględnienie cech interakcji, użytkowników i przedmiotów. Pokazaliśmy przewagę modelu P3LTR nad modelem RP3Beta pod względem metryk dokładności oraz pokrycia podczas ewaluacji offline.

Proponujemy także nową grafową sieć neuronową, P3GNN. Pokazujemy jej przewagę pod względem metryk dokładności nad istniejącymi grafowymi sieciami neuronowymi

na rozważanym zbiorze danych. Proponowana sieć uzyskuje podobne wartości metryk dokładności do modelu RP3Beta, lecz umożliwia generowanie reprezentacji wektorowych wierzchołków, co jest użyteczne w licznych zastosowaniach. Ponadto istnieje wiele elementów sieci P3GNN, które mogą zostać w przyszłości udoskonalone.

Prezentujemy także opublikowany przez autora zbiór danych pozwalający innym badaczom na reprodukcję części raportowanych wyników przy wykorzystaniu opublikowanego kodu źródłowego. Wskazujemy liczne zalety tego zbioru w porównaniu z innymi zbiorami danych stosowanymi do ewaluacji modeli rekomendacyjnych ofert pracy.

Ponadto, w rozprawie opisujemy szczegóły procesu skutecznego wdrożenia omawianych rozwiązań w serwisach Grupy OLX, włącznie z ewaluacją ich wpływu na metryki biznesowe.

# Abstract

In many practical applications, the number of available products often exceeds what users can feasibly familiarize themselves with, given the time they are willing to spend. Recommendation systems address this problem by selecting a few items to be presented to each user. By facilitating user decisions, these systems have a significant impact on online classifieds. Commonly used are collaborative filtering models that solely utilize information about interactions between users and items. In the dissertation, we present the results of a study on collaborative filtering models in online classifieds, using OLX Group's websites as examples.

In this work, we discuss the specific characteristics of online classifieds that influence the selection of recommendation and evaluation methods. We present offline evaluation results for existing models and show the advantage of the RP3Beta model over other methods. Furthermore, we verify these results by conducting A/B tests involving millions of users of the OLX Group's websites.

We present the infrastructure created to generate recommendations in real-time and show its applicability to many existing recommendation models. We then propose the RP3Beta real-time model that utilizes this infrastructure. We present the results of A/B tests showing a statistically significant advantage of this model over the baseline model.

We introduce a new graph-based recommendation method, P3LTR, which is a generalisation of the RP3Beta model. The proposed method, unlike the RP3Beta model, has parameters that are optimised when training the model and allows including features of interactions, users and items. We demonstrate the advantage of the P3LTR model over the RP3Beta model in terms of accuracy and coverage metrics during offline evaluation.

We also propose a new graph neural network, P3GNN. We show its superiority in terms of accuracy metrics over existing graph neural networks on the considered dataset. The proposed network achieves similar values of accuracy metrics to the RP3Beta model but allows the generation of vector representations of vertices, which is useful for numerous applications. In addition, there are many elements of the P3GNN network that can be further improved.

We also present a dataset published by the author that allows other researchers to reproduce parts of the reported results using the published source code. We point out several advantages of this dataset over other datasets used to evaluate job recommendation models.

Additionally, we describe the details of the process of successful implementation of the discussed solutions on OLX Group's websites, including an evaluation of their impact on business metrics.

# Spis treści

# Spis rysunków

# Spis tabel

# Oznaczenia

Poniżej przedstawiamy listę oznaczeń, które wykorzystywane są w wielu miejscach w pracy:

- $\mathcal{U}$ - zbiór wszystkich użytkowników,
- $u$ - użytkownik,
- $\mathcal{I}$ - zbiór wszystkich przedmiotów,
- $i$ - przedmiot,
- $r_{ui}$ - ustalona wartość (zwana *oceną*) przypisana zadanej parze użytkownik–przedmiot, która świadczy o preferencji użytkownika względem przedmiotu (przykładowo ocena przedmiotu w skali od 1 do 5, bądź liczba wyświetleń danego przedmiotu przez zadanego użytkownika),
- $\boldsymbol{R}$ - macierz ocen $(r_{ui})$ o wymiarach $|\mathcal{U}| \times |\mathcal{I}|$,
- $\hat{r}_{ui}$ - predykcja modelu względem oceny $r_{ui}$,
- $\hat{\boldsymbol{R}}$ - macierz predykcji ocen $(\hat{r}_{ui})$ o wymiarach $|\mathcal{U}| \times |\mathcal{I}|$,
- $\boldsymbol{x}_u$ - reprezentacja wektorowa użytkownika $u$,
- $\boldsymbol{y}_i$ - reprezentacja wektorowa przedmiotu $i$,
- $\mathcal{N}(u)$ - zbiór przedmiotów, z którymi użytkownik $u$ wszedł w interakcję,
- $\mathcal{N}(i)$ - zbiór użytkowników, którzy weszli w interakcję z przedmiotem $i$.

Rozdział 1

# Wprowadzenie

Wraz ze wzrostem ilości dostępnych informacji rośnie także znaczenie systemów rekomendacyjnych, których rolą jest sugerowanie użytkownikom odpowiednich treści [7]. Temat ten jest szeroko podejmowany zarówno w literaturze [24, 46], jak i w praktycznych zastosowaniach [2, 84], gdzie poprawa jakości modeli rekomendacji przekłada się na znaczące zyski firm [14, 51]. W rozprawie przedstawiamy nowe metody rekomendacji opracowane dla serwisów ogłoszeniowych oraz wykazujemy ich przewagę względem metod opisanych w literaturze.

Rozprawa powstała w ramach realizacji III Edycji programu „Doktorat wdrożeniowy" organizowanego przez Ministerstwo Nauki i Szkolnictwa Wyższego (obecnie Ministerstwo Edukacji i Nauki) w ramach współpracy Uniwersytetu im. Adama Mickiewicza w Poznaniu oraz Grupy OLX sp. z o.o. (dalej określanej jako Pracodawca). Celem zawarcia umowy przez Pracodawcę było zwiększenie liczby osób odpowiadających na zamieszczone na serwisach Pracodawcy ogłoszenia poprzez dostarczanie użytkownikom spersonalizowanych rekomendacji. Ze względu na priorytety Pracodawcy badania skoncentrowane były na kategorii praca, przy czym tworzone rozwiązania miały być w przyszłości wdrażane również w innych kategoriach. W momencie rozpoczęcia współpracy w serwisach Grupy OLX nie istniały modele rekomendacyjne dedykowane tej kategorii. Osiągnięcie celu wymagało zatem:

- przygotowania zbioru danych uczących,
- wdrożenia infrastruktury koniecznej do trenowania modeli rekomendacji,
- wdrożenia infrastruktury pozwalającej na generowanie rekomendacji użytkownikom w poszczególnych kanałach dostępu.

Ponadto, w celu uzyskania dobrej jakości rekomendacji konieczne było:

- przygotowanie środowiska do ewaluacji offline modeli,
- przygotowanie środowiska do ewaluacji online modeli,
- ewaluacja istniejących metod rekomendacji,
- implementacja i wdrożenie nowych metod rekomendacji, z uwzględnieniem potrzeb Pracodawcy.

Wszystkie te elementy zostały osiągnięte z dużym udziałem autora. Część wynikających z tych działań prac jest pominięta lub jedynie wspomniana w rozprawie, ponieważ nie stanowią one istotnego wkładu w rozwój nauki. Podejmowanie przez autora tych działań było związane z wdrożeniowym charakterem doktoratu. Celem autora oraz Pracodawcy było wdrożenie opracowanych metod, a nie jedynie opracowanie metod możliwych do wdrożenia. W rozprawie szczegółowo opisujemy działania stanowiące wkład w rozwój nauki.

W pracy koncentrujemy się na modelach wspólnej filtracji[1] (ang. *collaborative filtering*) ze względu na ich wysoką skuteczność oraz łatwość przygotowania danych uczących. Ich skuteczność względem innych metod rekomendacji była wielokrotnie badana w serwisach Pracodawcy w ramach prowadzonych przez autora prac. Niektóre z przeprowadzonych eksperymentów opisujemy w podrozdziale 1.3.2. Modele wspólnej filtracji wykorzystują jedynie informacje o interakcjach między użytkownikami i przedmiotami. Łatwość przygotowania danych uczących wynika zatem z braku konieczności przygotowywania danych dotyczących wartości cech użytkowników oraz przedmiotów. Cechy użytkowników nie zawsze są dostępne, ponieważ wielu z nich jest anonimowych, a ponadto nie zawsze mogą być swobodnie przetwarzane ze względu na politykę prywatności. Cechy przedmiotów dostępne są w częściowo nieustrukturyzowanej formie, ponieważ większość informacji zawarta jest w opisach ogłoszeń. Pozyskanie tych cech wymaga wyspecjalizowanych w tym celu modeli, a obecność serwisu w różnych krajach, czasem dwujęzycznych, dodatkowo komplikuje to zadanie. W pracy wielokrotnie jednak wspominamy o możliwości wykorzystania cech użytkowników i przedmiotów w omawianych modelach wspólnej filtracji.

## 1.1. Systemy rekomendacyjne w serwisach ogłoszeniowych

Serwisy ogłoszeniowe (ang. *online classifieds*) to serwisy internetowe, na których ogłoszeniodawcy umieszczają ogłoszenia dotyczące sprzedaży bądź wynajmu dóbr, świadczenia usług, bądź pracy [40]. Są one internetowym odpowiednikiem ofert zamieszczanych w gazetach. Systemy rekomendacyjne odgrywają dużą rolę w działalności serwisów ogłoszeniowych, o czym świadczą prowadzone w tej dziedzinie badania [1, 2, 84], a także wyniki prezentowane w rozprawie. Wymagają one jednak zaadresowania szczególnych problemów związanych z cechami takich serwisów, które omawiamy poniżej.

- **Czas emisji ogłoszenia jest ograniczony** (na przykład do 30 dni). Systemy rekomendacyjne uczone są na danych historycznych, lecz rekomendują wyłącznie aktualnie dostępne przedmioty. W serwisach ogłoszeniowych przedmioty stają się niedostępne znacznie szybciej niż w przypadku filmów bądź przepisów kulinarnych. Podobny

[1] Pojęcie „modele wspólnej filtracji", podobnie jak wiele innych wykorzystanych w pracy tłumaczeń terminologii anglojęzycznej, pochodzi z książki pt. *Praktyczne systemy rekomendacji* [39].

problem możemy zaobserwować w serwisach informacyjnych, gdzie rekomendowane są zwykle najświeższe wiadomości.

- **Ogłoszeniodawca potrzebuje zwykle tylko jednego użytkownika do skutecznego przeprowadzenia transakcji**, ponieważ często posiada tylko jeden sprzedawany przedmiot, bądź wakat. Zatem ten sam przedmiot nie powinien być rekomendowany zbyt dużej liczbie użytkowników. Sytuacja ta jest inna niż w przypadku serwisów informacyjnych, gdzie najważniejsze wiadomości są często trafną rekomendacją dla dużej liczby osób. Podobny problem możemy zaobserwować w serwisach randkowych.
- **Zawarcie transakcji najczęściej przebiega poza serwisem ogłoszeniowym.** Serwisy ogłoszeniowe umożliwiają nawiązanie kontaktu między użytkownikami a ogłoszeniodawcą poprzez wiadomości tekstowe bądź kontakt telefoniczny, a także śledzą fakt zaistnienia takiego kontaktu. Najczęściej jednak nie jest możliwe rozstrzygnięcie, który z takich kontaktów zakończył się zawarciem transakcji. Informacja o zawarciu transakcji mogłaby ułatwić zadanie generowania trafnych rekomendacji. Przykładowo użytkownik po zakupie lodówki, prawdopodobnie nie będzie zainteresowany zakupem kolejnej, więc od momentu zawarcia transakcji powinien on przestać otrzymywać rekomendacje ogłoszeń dotyczących sprzedaży innych lodówek.
- **Większość użytkowników jest anonimowa.** W serwisach ogłoszeniowych zwykle nie jest wymagane założenie konta w celu przeglądania ofert i możliwości kontaktu z ogłoszeniodawcą poprzez kontakt telefoniczny. Zatem konieczne jest zastosowanie metod rekomendacji, które nie wymagają istnienia profilu użytkownika oraz są w stanie generować trafne rekomendacje na podstawie, często krótkiej, historii przeglądanych ogłoszeń.

Ponadto, w celu doboru właściwych metod rekomendacji istotne jest uwzględnienie liczby użytkowników oraz przedmiotów. W przypadku serwisów Pracodawcy:

- rekomendacje generowane są dla kilkunastu milionów użytkowników (kilku milionów w przypadku ofert pracy),
- rekomendowane jest kilkanaście milionów przedmiotów (około stu tysięcy w przypadku ofert pracy).

W przypadku większości kategorii ogłoszeń (na przykład dóbr czy usług) ogłoszeniodawca nie ma szczególnych wymagań wobec użytkownika, z którym zawiera transakcję. Sytuacja ta jest odmienna w przypadku ofert pracy, gdzie pracodawca zwykle oczekuje od użytkownika konkretnych umiejętności, doświadczenia czy wykształcenia. Generując rekomendacje powinniśmy zatem nie tylko znajdować oferty, na które użytkownik najchętniej będzie aplikował, lecz uwzględniać także informację o prawdopodobieństwie zaakceptowania jego aplikacji. Problem ten znany jest w literaturze jako problem wzajem-

nych rekomendacji (ang. *reciprocal recommenders*, [102]). Nie adresujemy tego problemu w rozprawie ze względu na wymienione niżej powody.

- Zwykle nie posiadamy informacji o tym, który kandydat został finalnie zatrudniony przez pracodawcę.
- Najczęściej użytkownicy aplikują na oferty pracy, w których spełniają co najmniej część wymagań zawartych w tych ogłoszeniach. Rekomendowane oferty są zwykle podobne do dotychczas przeglądanych, zatem często użytkownicy spełniają wymagania w nich stawiane.
- Estymacja prawdopodobieństwa zaakceptowania aplikacji jest trudnym zadaniem, ponieważ proces myślowy rekrutera może uwzględniać dodatkowe, niezawarte w ogłoszeniu informacje (na przykład dopasowanie kandydata do zespołu w którym ma pracować), a także opiera się w większości na nieustrukturyzowanych danych (CV, list motywacyjny).
- Modele rekomendacji w serwisie Indeed[2], jednym ze światowych liderów na rynku ofert pracy, są optymalizowane pod maksymalizację liczby aplikacji, a nie zatrudnionych osób [94], ponieważ zatrudnienie jest ich zdaniem „zbyt rzadkim sygnałem". Wzorując się zatem na konkurencji, również zdecydowaliśmy się na takie uproszczenie.

W opracowywanych i wdrażanych metodach adresujemy wymienione cechy serwisów ogłoszeniowych poprzez dobór metod rekomendacji oraz ich ewaluację.

## 1.2. Serwisy ogłoszeniowe Grupy OLX

Grupa OLX, z siedzibą w Amsterdamie, jest właścicielem szesnastu marek prowadzących serwisy ogłoszeniowe w około trzydziestu krajach i zatrudnia łącznie ponad 10 tysięcy pracowników. Jest ona częścią grupy Naspers[3] należącej do grupy Prosus[4]. Do najważniejszych marek Grupy OLX działających w Polsce należy OLX, Otomoto, Otodom oraz Fixly. Marka OLX[5], założona w 2006 roku, jest obecna w około piętnastu krajach i obsługuje setki milionów użytkowników miesięcznie. W ramach serwisów tej marki użytkownicy zamieszczają ogłoszenia dotyczące ofert pracy, sprzedaży nieruchomości, usług i towarów. Prace prowadzone w ramach doktoratu zostały wdrożone w siedmiu serwisach marki OLX działających w Europie i Azji: w Polsce (`olx.pl`), Ukrainie (`olx.ua`), Rumunii (`olx.ro`), Portugalii (`olx.pt`), Bułgarii (`olx.bg`), Kazachstanie (`olx.kz`) i Uzbekistanie (`olx.uz`).

[2] `https://www.indeed.com/`, dostęp: 2023-11-18
[3] `https://www.naspers.com/`, dostęp: 2023-11-18
[4] `https://www.prosus.com/`, dostęp: 2023-11-18
[5] `https://www.olxgroup.com/brands/olx`, dostęp: 2023-11-18

## 1.3. Rozwój rekomendacji ofert pracy w OLX

W podrozdziale przedstawiamy zarys wdrożonych rozwiązań oraz ich wpływ na serwisy Pracodawcy. Celem jest umożliwienie ewaluacji wdrożeniowego aspektu doktoratu. Szczegółowy opis zastosowanych metod i przeprowadzonej ewaluacji przedstawiony jest w dalszej części rozprawy.

Rozpoczynając realizację doktoratu wdrożeniowego autor samodzielnie pracował nad opracowaniem i wdrożeniem pierwszego modelu rekomendacji dedykowanego kategorii praca w serwisie `olx.pl`. Po przedstawieniu wyników eksperymentu, Pracodawca zdecydował się powołać zespół dedykowany temu zadaniu, którego autor był członkiem. W okresie realizacji doktoratu (do 2023-06-20) autor wraz z zespołem przeprowadził 75 testów A/B obejmujących między innymi testowanie nowych modeli, ich parametrów, metod agregacji rekomendacji pochodzących z różnych modeli, wykorzystywanych przez modele cech, wdrażanie modeli w nowych kanałach dostępu, zmiany infrastruktury oraz zmiany interfejsu użytkownika. Najważniejsze zmiany jakich dokonaliśmy podsumowane są w tabeli 1.1.

### 1.3.1. Skalowanie rozwiązań

Początkowo priorytetem Pracodawcy było dostarczanie użytkownikom spersonalizowanych rekomendacji poprzez przesyłanie im powiadomień email oraz push w serwisie `olx.pl`, co zostało zrobione podczas pierwszego roku doktoratu. Przykładowy email z rekomendacjami przesyłany użytkownikom przedstawiony jest na rysunku 1.1. W tym samym roku wprowadzony został tak zwany profil kandydata, w którym użytkownicy mogą podać informacje na temat swojej edukacji, doświadczenia i umiejętności, a także opisać preferencje dotyczące przyszłej pracy, takie jak lokalizacja, zarobki oraz typ i rodzaj umowy. Następnie otrzymują oni spersonalizowane rekomendacje ofert pracy, których generowanie uwzględnia zarówno ich interakcje z ogłoszeniami, jak i uzupełnione informacje (rysunek 1.2).

W kolejnym roku priorytetem była implementacja wdrożonych metod w rozwiązaniach oferowanych na innych rynkach. W rezultacie, po drugim roku prac, rozwiązania zostały wdrożone w siedmiu serwisach Pracodawcy. Rozszerzyliśmy również zasięg rekomendacji o kolejny kanał dostępu – rekomendowanie ogłoszeń na stronie głównej, gdzie użytkownicy serwisów OLX otrzymują rekomendacje z różnych kategorii. W przypadku, gdy użytkownik został przez nas zaklasyfikowany jako osoba aktywnie poszukująca pracy, otrzymywał on rekomendacje przygotowane przez nasz dedykowany w tym celu model.

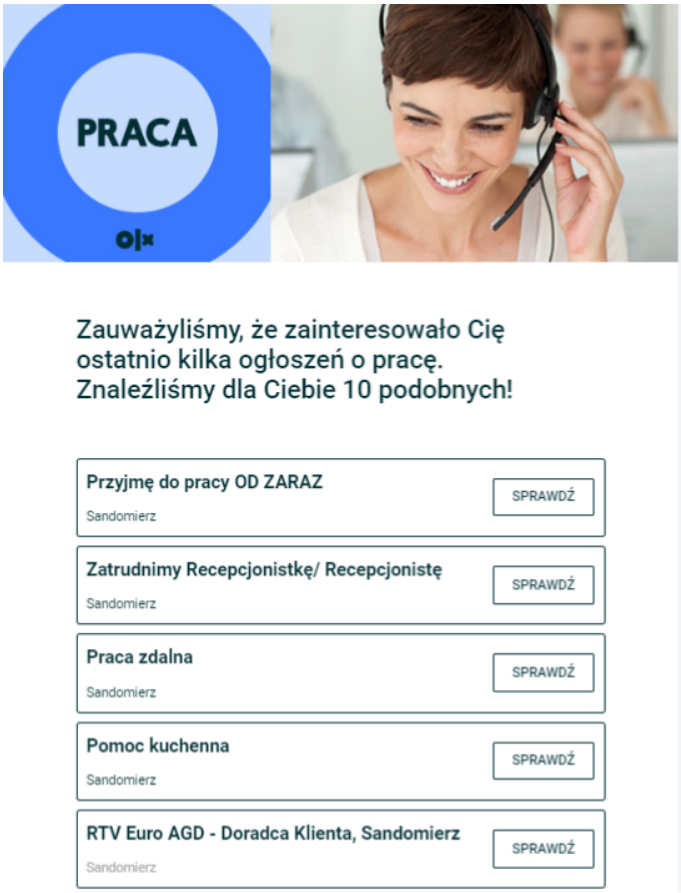
W trzecim roku wdrożone przez nas rozwiązania wykorzystane zostały w kolejnych kanałach dostępu. Nasze modele zaczęły być wykorzystywane na stronie przedmiotu poprzez generowanie przedmiotów podobnych do zadanego. Rozpoczęliśmy również

**Tabela 1.1.** Rozwój modeli rekomendacji dedykowanych kategorii praca w OLX

| **Rok akademicki** | **Serwisy** | **Kanały dostępu** | **Wdrożone modele rekomendacji** |
|---|---|---|---|
| 2018/2019 | brak | brak | brak |
| 2019/2020 | `olx.pl` | powiadomienia email<br>powiadomienia push<br>profil kandydata | ALS<br>Solr<br>tf-idf |
| 2020/2021 | `olx.pl`<br>`olx.ro`<br>`olx.pt`<br>`olx.bg`<br>`olx.ua`<br>`olx.kz`<br>`olx.uz` | powiadomienia email<br>powiadomienia push<br>profil kandydata<br>strona główna | Elasticsearch<br>ALS rerank<br>RP3Beta<br>Item2Vec |
| 2021/2022 | `olx.pl`<br>`olx.ro`<br>`olx.pt`<br>`olx.bg`<br>`olx.ua`<br>`olx.kz`<br>`olx.uz` | powiadomienia email<br>powiadomienia push<br>profil kandydata<br>strona główna<br>strona przedmiotu<br>po aplikowaniu na ofertę pracy<br>lista wyszukiwanych ogłoszeń | RP3Beta rerank<br>RP3Beta mlrank<br>RP3Beta real-time<br>i2i reranker |

prezentowanie użytkownikom rekomendacji po aplikacji na oferty pracy. Kolejnym kanałem dostępu jest lista wyszukiwanych przez użytkownika ogłoszeń, którą wzbogaciliśmy o wygenerowane dla niego rekomendacje.

Wskutek wdrożonych rozwiązań, **w pierwszym kwartale 2023 roku, prawie** 5 **milionów odpowiedzi na oferty pracy było następstwem udzielonych przez nasze modele rekomendacji**.

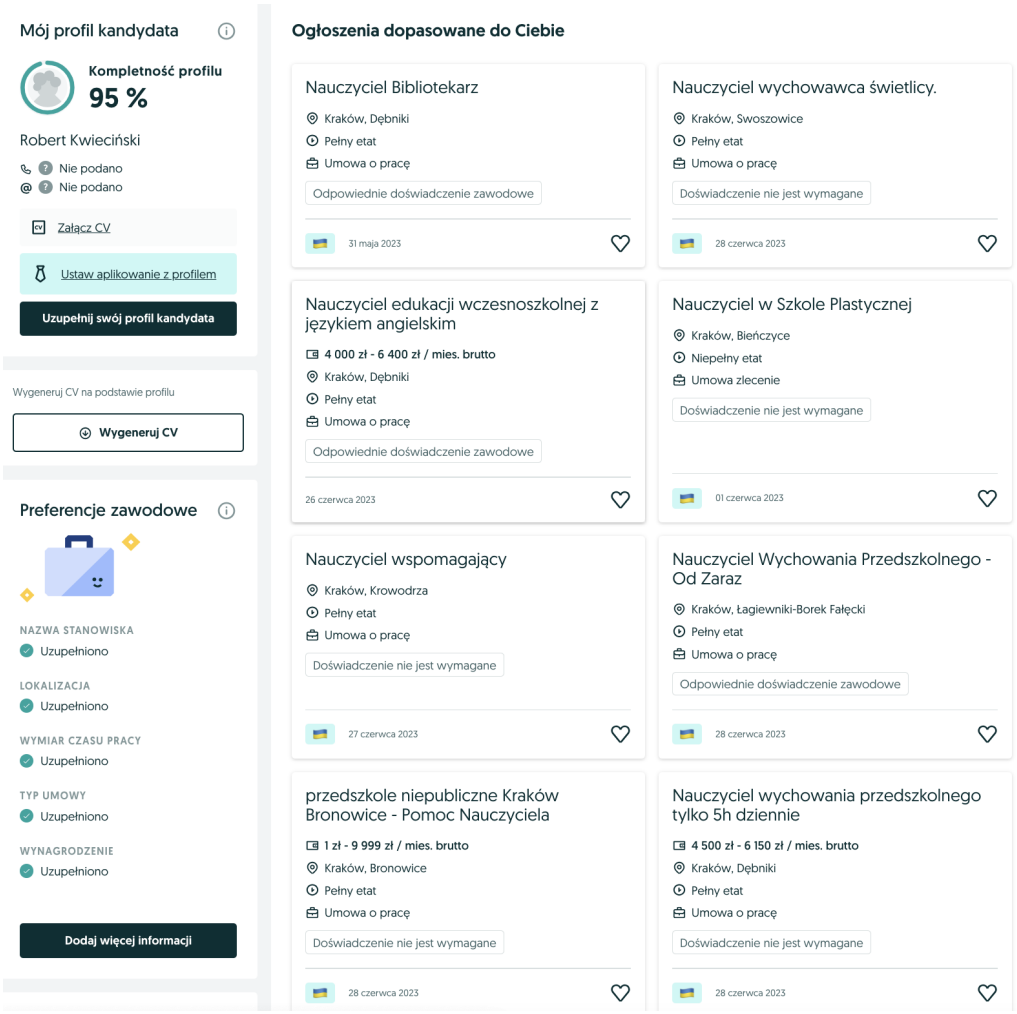



**Rysunek 1.1.** Przykładowy email z rekomendacjami przesyłany użytkownikom serwisu `olx.pl`

### 1.3.2. Wdrożone modele rekomendacji

W podrozdziale opisujemy modele rekomendacji, które zostały przez nas wdrożone i ewaluowane względem innych podejść poprzez testy A/B przeprowadzone w serwisach Grupy OLX. Lista umieszczona w tabeli 1.1 przedstawia wyłącznie modele ewaluowane online z użytkownikami. Pominięte zostały w niej modele, które podczas ewaluacji offline nie uzyskały dostatecznie dobrych wyników. Ewaluacja offline pominiętych modeli wspólnej filtracji będzie przedstawiona w dalszej części rozprawy. Wiele z przedstawionych w tabeli 1.1 modeli było wielokrotnie modyfikowanych, bądź wzbogacanych o nowe cechy, czego szczegółowo nie opisujemy. Celem tego rozdziału jest jedynie nakreślenie procesu rozwoju modeli rekomendacyjnych ofert pracy w serwisach OLX.

Pierwszym wdrożonym modelem był model **ALS** [64] będący modelem wspólnej filtracji opartym o metodę faktoryzacji macierzy (ang. *matrix factorization*). Szczegółowy




**Rysunek 1.2.** Przykładowe rekomendacje prezentowane użytkownikom serwisu `olx.pl` w profilu kandydata

opis tego modelu znajduje się w podrozdziale 4.1.1. Model ten został wdrożony w powiadomieniach email oraz push, a później także w profilu kandydata. Niestety, jako model wspólnej filtracji, nie uwzględniał on cech użytkowników zawartych w ich profilach, na podstawie których zobowiązaliśmy się dostarczać użytkownikom rekomendacji. Zdecydowaliśmy się zatem wdrożyć model oparty o silnik wyszukiwania **Solr**[6], który wyszukiwał oferty o tytułach podobnych do aktualnego stanowiska użytkownika, uwzględniając również lokalizację i kategorię wyszukiwania. W rezultacie część rekomendacji prezentowanych przez nas w profilu kandydata pochodziła z modelu ALS, część z modelu Solr. Następnie podjęliśmy próbę poprawy jakości rekomendacji opartych o profil kandydata,

[6] `https://solr.apache.org/`, dostęp: 2023-11-18

poprzez zastąpienie modelu Solr modelem **tf-idf** [105]. Model ten mierzył podobieństwo między cechami zawartymi w profilu użytkownika a odpowiednimi cechami ogłoszeń, poprzez analizę częstości występowania tych samych słów w odpowiadających sobie cechach. Wykorzystywał on więcej cech zawartych w profilu kandydata niż model Solr (do którego z powodów technicznych trudne było dodanie kolejnych cech). Po przeprowadzeniu testu A/B model Solr okazał się lepszy od modelu tf-idf.

W kolejnym roku wdrożyliśmy model oparty o silnik wyszukiwania **Elasticsearch**[7], który po przeprowadzeniu testów A/B, zastąpił model Solr. Model Elasticsearch był przez nas wielokrotnie wzbogacany o nowe cechy i optymalizowany. Kolejne prace dotyczyły adaptacji modelu wspólnej filtracji, w taki sposób, aby generowane rekomendacje uwzględniały również informacje zawarte w profilu kandydata. Zdecydowaliśmy się na przesortowanie rekomendacji wygenerowanych z modelu wspólnej filtracji za pomocą informacji zawartych w profilu. Takie rozwiązanie pozwoliło na utworzenie rozwiązania niezależnego od modelu bazowego. Rozpoczęliśmy od regułowego sortowania wykorzystującego dopasowanie preferencji kandydata do rekomendowanych ogłoszeń. Powstał model **ALS rerank**, który dawał podobne rezultaty do modelu bazowego ALS. Następnie pracowaliśmy nad zastąpieniem modelu bazowego ALS, innym modelem wspólnej filtracji, ponieważ był on jedynym modelem wykorzystywanym dla użytkowników nieposiadających profilu kandydata, którzy stanowili zdecydowaną większość użytkowników serwisów OLX. Wynikiem tych prac było zastąpienie modelu ALS, modelem grafowym **RP3Beta** [98] opisanym w podrozdziale 4.1.4.

W kolejnym roku wykorzystaliśmy stworzony wcześniej moduł służący do przesortowania rekomendacji modelu RP3Beta uzyskując model **RP3Beta rerank**, który jednak również nie był istotnie lepszy od modelu bazowego RP3Beta. Następnie wdrożyliśmy model **RP3Beta mlrank**, w którym zamiast regułowej formy przesortowywania rekomendacji zastosowaliśmy model uczenia maszynowego wytrenowany na reakcjach użytkowników na historycznie prezentowane im rekomendacje. Uzyskaliśmy istotną poprawę względem modelu bazowego. Wkrótce jednak wdrożyliśmy model **RP3Beta real-time**, który pozwalał na generowanie rekomendacji uwzględniających najświeższe interakcje użytkowników. Temu wdrożeniu poświęcony jest rozdział 5. Przetestowaliśmy również model **i2i reranker**, którego zadaniem było przesortowanie rekomendacji na stronie przedmiotu. Dawał on jednak gorsze wyniki od bazowego modelu RP3Beta.

Wszystkie te modele zostały wdrożone w kilkuosobowym zespole z udziałem autora. Wdrożenia opisane szczegółowo w kolejnych rozdziałach tej rozprawy zostały opracowane głównie przez autora (szczegółowy opis dostępny jest w oświadczeniach o współautorstwie).

W trakcie trwania prac, inny zespół w firmie opracował model rekomendacyjny

[7] `https://www.elastic.co/`, dostęp: 2023-11-18

**Item2Vec**[8] oparty o metody uczenia głębokiego, który wykorzystuje zarówno informacje o interakcjach użytkowników, jak i szczegółowe informacje o przedmiotach. Został on wytrenowany dla każdej kategorii osobno. Model Item2Vec wytrenowany dla kategorii praca został przez nas porównany z modelem RP3Beta poprzez testy A/B w różnych kanałach dostępu. Zaobserwowaliśmy istotną statystycznie przewagę modelu RP3Beta, mimo że model Item2Vec wykorzystuje dodatkową wiedzę o ogłoszeniach o pracę. Potwierdziliśmy zatem, że właściwie dobrany i dostrojony prosty model może przewyższać znacznie bardziej zaawansowane modele głębokiego uczenia. Jest to zgodne z wynikami badań, w których porównano kilka podejść nieneuronowych, w tym RP3Beta, z bardziej zaawansowanymi modelami neuronowymi [12, 32]. Wynik ten jest jednym z powodów, dla których większość prac autora skoncentrowana była na rozwoju modeli wspólnej filtracji. W rozprawie nie opisujemy szczegółowo eksperymentów, w których dokonaliśmy porównania modeli RP3Beta oraz Item2Vec, ponieważ:

- model Item2Vec powstał bez udziału autora,
- autor nie może opublikować odpowiedniego zbioru danych ani kodu źródłowego,
- model Item2Vec nie należy do grupy modeli wspólnej filtracji, na których koncentrujemy się w rozprawie.

Przedstawiliśmy zarys procesu wdrażania i skalowania rozwiązań w serwisach Pracodawcy. W dalszej części pracy szczegółowo omówimy badane i opracowane modele wspólnej filtracji.

## 1.4. Cele rozprawy

Celem głównym rozprawy doktorskiej jest **przedstawienie nowych metod rekomendacji opracowanych dla serwisów ogłoszeniowych oraz wykazanie ich przewagi względem metod opisanych w literaturze**.

Cele pomocnicze rozprawy to:

1. przedstawienie opublikowanego przez autora zbioru danych,
2. zaprezentowanie sposobu doboru i ewaluacji modeli rekomendacyjnych w przypadku serwisów ogłoszeniowych oraz przedstawienie wyników takiej ewaluacji dla metod opisanych w literaturze,
3. opisanie procesu skutecznego wdrożenia modeli rekomendacyjnych w serwisach ogłoszeniowych Pracodawcy.

Cele te będą realizowane w przedstawionej dysertacji, a w podsumowaniu zostanie zawarta dyskusja nad stopniem ich zrealizowania.

[8] `https://tech.olx.com/item2vec-neural-item-embeddings-to-enhance-recommendations-1fd948a6f293`, dostęp: 2023-11-18

## 1.5. Struktura rozprawy

Rozprawa składa się z ośmiu rozdziałów wliczając wprowadzenie i podsumowanie.

Rozdział pierwszy rozpoczęliśmy od zaznaczenia wdrożeniowego aspektu rozprawy. Następnie przedstawiliśmy przedsiębiorstwo, w którym wdrożyliśmy opracowane modele, a także rozwój i zakres wdrażanych rozwiązań. Wskazaliśmy również cele rozprawy oraz opisaliśmy jej strukturę.

W drugim rozdziale zdefiniowaliśmy systemy rekomendacyjne, opisaliśmy ich rodzaje, wyzwania oraz sposoby ewaluacji.

Trzeci rozdział poświęcony jest przedstawieniu zbioru danych, OLX Jobs Interactions, który opublikowaliśmy w celu uzyskania pełnej reprodukowalności uzyskanych przez nas wyników. Zbiór ten jest największym znanym autorowi publicznie dostępnym zbiorem danych zawierających interakcje użytkowników z ofertami pracy.

W rozdziale czwartym przestawiamy wyniki ewaluacji istniejących modeli rekomendacyjnych na opublikowanym zbiorze danych. Prezentujemy również wyniki testów A/B przeprowadzonych w serwisach Pracodawcy. Wyniki te zostały opublikowane w artykule *Job recommendations: benchmarking of collaborative filtering methods for classifieds* [78].

W kolejnym rozdziale prezentujemy stworzoną przez autora adaptację modelu RP3Beta, która umożliwia generowanie użytkownikom rekomendacji uwzględniając ich najświeższe interakcje z ogłoszeniami o pracę. Przedstawiamy tam również infrastrukturę techniczną wykorzystywaną w OLX do generowania rekomendacji za pomocą zaproponowanego modelu oraz wyniki testu A/B pokazujące skuteczność tego modelu. Opisane wyniki zostały opublikowane w pracy *Comparison of Real-Time and Batch Job Recommendations* [80].

W rozdziale szóstym proponujemy nowy model rekomendacyjny, P3 Learning to Rank (P3LTR), który jest uogólnieniem modelu RP3Beta. Zaproponowany model uczy się istotności poszczególnych interakcji między użytkownikiem i przedmiotem w procesie uczenia modelu. Następnie inferencja wykonywana jest w sposób podobny do modelu RP3Beta, dzięki czemu zachowana jest wydajność oraz interpretowalność generowanych rekomendacji. Model ten został przedstawiony w publikacji naukowej *Learning edge importance in bipartite graph-based recommendations* [79].

W rozdziale siódmym badamy metody rekomendacji wykorzystujące grafowe sieci neuronowe. Dokonujemy ewaluacji istniejących metod na zbiorze danych pochodzącym z jednego z serwisów Pracodawcy. Proponujemy model P3GNN, który uzyskuje lepsze wyniki od rozważanych grafowych modeli neuronowych. Proponujemy także nowe funkcje straty oraz metodę generowania przykładów negatywnych.

Dysertację kończy podsumowanie wskazujące na stopień osiągnięcia postawionych w pracy celów oraz prezentujące kierunki dalszych prac.

Rozdział 2

# Systemy rekomendacyjne

Celem rozdziału jest zdefiniowanie i klasyfikacja systemów rekomendacji, przedstawienie najważniejszych wyzwań oraz stosowanych metod ewaluacji. Do wszystkich tych elementów odnosić będziemy się w kolejnych rozdziałach rozprawy.

W literaturze wykazano, że prezentowanie użytkownikom zbyt wielu możliwości może doprowadzić do paradoksu wyboru (ang. *overchoice* [52]), w efekcie zmniejszając konsumpcję produktów [26, 68] i przychody witryny [18]. Aby zmniejszyć liczbę prezentowanych możliwości, użytkownicy mogą zostać poproszeni o dodatkowe informacje na temat ich oczekiwań dotyczących produktów. Niestety, nawet jeśli użytkownik jest skłonny poświęcić czas na podanie tych informacji, liczba możliwości jest zwykle wciąż duża. Adresując ten problem, wiele firm, takich jak Netflix [57] czy Amazon [110], wdrożyło spersonalizowane systemy rekomendacji, których rolą jest sugerowanie użytkownikom odpowiednich przedmiotów [7].

Istnieją dwa główne sposoby zdefiniowania systemu rekomendacji [7]: podejście predykcyjne i rankingowe.

W **podejściu predykcyjnym** zadanie polega na estymowaniu wartości ocen, które użytkownicy wystawiają przedmiotom. Przyjmijmy, że mamy $n$ użytkowników oraz $m$ przedmiotów. Oceny użytkowników mogą zatem zostać przedstawione w postaci macierzy $\boldsymbol{R}$ o wymiarach $n \times m$, w której część wartości jest nieznana. Problem polega na stworzeniu metody pozwalającej na uzupełnienie brakujących wartości w tej macierzy (ang. *matrix completion*). W większości praktycznych zastosowań liczba dostępnych przedmiotów jest o kilka rzędów wielkości większa niż średnia liczba przedmiotów ocenionych przez użytkowników, zatem jedynie niewielka część wartości macierzy $\boldsymbol{R}$ jest znana.

W **podejściu rankingowym** celem jest wyznaczenie pewnej liczby $k$ najbardziej odpowiednich względem zadanej miary ewaluacji przedmiotów. W większości praktycznych zastosowań generowane jest co najwyżej kilkadziesiąt rekomendacji dla użytkownika. W tym przypadku, nie jest konieczna predykcja ocen wszystkich przedmiotów w celu wybrania przedmiotów z najwyższymi estymowanymi ocenami. Ponadto, podejście predykcyjne może prowadzić do wyboru nieoptymalnego algorytmu z punktu widzenia

wyznaczania $k$ najbardziej odpowiednich rekomendacji. Przykładowo, niedokładne estymacje ocen przedmiotów, które znajdują się na dole listy mogą nie mieć żadnego wpływu na wygenerowane rekomendacje, zaś wpływać na metrykę ewaluacji zadania predykcji (często stosowaną metryką jest pierwiastek błędu średniokwadratowego). Podobnie, przekształcenie estymowanych ocen poprzez dowolną funkcję niemalejącą nie wpływa na kolejność przedmiotów, a może mieć wpływ na metrykę ewaluacji zadania predykcji.

Systemy rekomendacyjne są wykorzystywane w wielu dziedzinach, między innymi w muzyce [112], turystyce [83], modzie [66] i żywności [49]. Przeprowadzono nad nimi także liczne badania w dziedzinie handlu elektronicznego (ang. *e-commerce*) [71, 85], serwisów ogłoszeniowych [116] i rekomendacji ofert pracy [1, 2, 11].

Metody rekomendacji dla milionów użytkowników i milionów przedmiotów są z powodzeniem wdrażane od co najmniej 25 lat. Już w roku 1998 Amazon wdrożył model nazwany przez twórców *item-to-item collaborative filtering* [88, 110]. Systemy rekomendacji zyskały popularność w 2007 roku, kiedy firma Netflix zorganizowała konkurs *Netflix Prize* i zaoferowała milion dolarów za stworzenie modelu rekomendacji przewyższającego algorytm stworzony przez firmę o 10% pod względem pierwiastka błędu średniokwadratowego [16]. W 2016 roku Netflix opisał swoje systemy rekomendacji i ogłosił, że 80% czasu oglądania w Netflix pochodzi z rekomendacji [51]. W ciągu ostatnich kilku lat obserwujemy przemysłowe zastosowania bardziej zaawansowanych technik rekomendacji. W 2018 r. Pinterest opisał PinSage [135], model grafowej sieci neuronowej wytrenowany na grafie z 3 miliardami wierzchołków i 18 miliardami krawędzi, który został wdrożony w Pinterest i przewyższył jakością rekomendacji poprzednie modele. W 2022 roku Pinterest opublikował pracę opisującą ItemSage [14], nową metodę rekomendacji opartą na PinSage i architekturze transformer, której wdrożenie zwiększyło wskaźnik GMV (ang. *Gross Merchandise Value*) o około 7%. Z kolei, austriacka platforma pracy Studo Jobs opublikowała wyniki testów A/B przeprowadzonych w celu oceny wydajności wykorzystania reprezentacji wektorowych ogłoszeń do generowania rekomendacji ofert pracy w czasie rzeczywistym [81]. W 2020 roku LinkedIn poinformował o znacznej poprawie swojego systemu rekomendacji osiągniętej dzięki zwiększeniu jakości reprezentacji ofert pracy [84]. Wykorzystana została stworzona przez firmę taksonomia tytułów ogłoszeń oraz umiejętności, metody głębokiego uczenia transferowego oraz automatyczne uwzględnianie informacji zwrotnej od użytkowników.

Wspomniane przykłady pokazują powszechność oraz duże znaczenie modeli rekomendacji w wielu dziedzinach. Pozwoliły one określić Pracodawcy potencjalny wpływ rozwoju modeli rekomendacyjnych na działalność serwisów, a ponadto stanowiły inspirację dla autora pracy.

## 2.1. Typy systemów rekomendacyjnych

Zazwyczaj rozróżniamy trzy główne podejścia do dostarczania rekomendacji: filtrowanie oparte na treści, wspólną filtrację oraz podejścia hybrydowe łączące obie te techniki [5,7].

W **filtrowaniu opartym na treści** [70,99] wykorzystujemy informacje o przedmiotach (np. cena, lokalizacja, kategoria), a czasem również użytkownikach (np. zainteresowania, wykształcenie, umiejętności). Przy obliczaniu rekomendacji dla zadanego użytkownika, systemy te zazwyczaj wykorzystują również historię jego interakcji (oceny, wizyty, zakupy, odpowiedzi), natomiast nie jest wykorzystywana historia interakcji innych użytkowników. Modele te rekomendują zwykle przedmioty o podobnych cechach do przedmiotów, z którymi użytkownik wszedł w interakcje. Z tego powodu rekomendacje mogą wydawać się rozsądne, lecz ich skuteczność może obniżać brak elementu zaskoczenia użytkownika.

W podejściu **wspólnej filtracji** [24,125] wykorzystujemy wyłącznie informacje o interakcjach między użytkownikami a przedmiotami. Podobieństwo między przedmiotami może być tutaj wyliczane z wykorzystaniem informacji o innych przedmiotach odwiedzonych przez oglądających wspomniane przedmioty użytkowników, nie zaś podobieństwo cech tych przedmiotów. Załóżmy przykładowo, że osoby kupujące pieluszki często kupują również słuchawki wygłuszające. W odróżnieniu od metod filtrowania opartego na treści, metody wspólnej filtracji prawdopodobnie zarekomendują słuchawki wygłuszające osobie kupującej pieluszki. Mimo braku informacji o cechach użytkowników i przedmiotów, systemy te są zdolne do generowania trafnych rekomendacji, szczególnie w przypadku dużej liczby interakcji między użytkownikami a przedmiotami.

Wraz z rozwojem systemów rekomendacji wyszczególniono także wiele innych ich typów [69,111], które przedstawiamy poniżej.

- **Systemy demograficzne** (ang. *demographic* [10]) bazują na wykorzystaniu cech demograficznych użytkowników, takich jak wiek, płeć, miejsce zamieszkania czy wykonywany zawód. Poprzez porównanie cech użytkowników są one w stanie znajdować użytkowników podobnych do zadanego i dostarczyć rekomendacji użytkownikowi nawet zanim wykona on jakąkolwiek interakcję z przedmiotami.
- **Systemy bazujące na wiedzy** (ang. *knowledge-based* [7]) wykorzystują informacje o wskazanych przez użytkownika preferencjach oraz cechy przedmiotów. Nie wykorzystują one interakcji użytkowników do generowania rekomendacji. Przykładem takiego systemu jest, wspomniany wcześniej, wdrożony w serwisach Pracodawcy model Elasticsearch, który generuje rekomendacje ofert pracy w oparciu o podobieństwo przedmiotów do preferencji użytkownika opisanych w jego profilu.
- **Systemy świadome kontekstu** (ang. *context-aware* [6,77]) uwzględniają zmienność preferencji użytkownika powodowaną między innymi zmianą pory dnia, pogody czy lokalizacji w jakiej użytkownik się znajduje. Przykładowo, użytkownik może preferować spokojniejsze utwory muzyczne w godzinach wieczornych.

- **Systemy oparte na sieciach społecznościowych** (ang. *social network-based*, [61]) wykorzystują relacje społeczne w procesie tworzenia rekomendacji. Opierają się na badaniach pokazujących, że preferencje znanych sobie osób są podobne [61]. Uwzględnienie relacji społecznych może być szczególnie skuteczne w przypadku użytkowników z niewielką liczbą interakcji z przedmiotami.
- **Systemy konwersacyjne** (ang. *conversational* [69]) umożliwiają interakcje użytkowników z systemem. Przykładowo użytkownik może zadawać dodatkowe pytania dotyczące wygenerowanych rekomendacji lub je oceniać.

Wybór odpowiedniej techniki rekomendacji zależy również od typu interakcji między użytkownikami a przedmiotami oraz sposobu uczenia się modelu. Omawiamy te kwestie w kolejnych podrozdziałach.

### 2.1.1. Podział systemów w zależności od jawności ocen

Jakość metod rekomendacji często zależy od typu interakcji między użytkownikami a przedmiotami. Wyróżniamy **oceny jawne** (ang. *explicit ratings*) oraz **oceny niejawne** (ang. *implicit ratings*). W przypadku ocen jawnych użytkownik wprost wyraża swoją preferencję w stosunku do przedmiotu oceniając go w skali liczbowej, bądź wybierając jedną z kilku możliwości (na przykład określając czy przedmiot mu się podoba czy nie). W przypadku ocen niejawnych, ocena użytkownika jest wnioskowana na podstawie jego interakcji z przedmiotem, na przykład wyświetlenia ogłoszenia, bądź obejrzenia filmu. Hu, Koren i Volinsky [64] przedstawili model wspólnej filtracji, który szczegółowo przedstawiamy w podrozdziale 4.1.1, dla zbiorów danych z niejawnymi ocenami wykorzystujący metody faktoryzacji macierzy [74] stosowane wcześniej dla zbiorów danych z jawnymi ocenami. W swojej pracy wymienili oni cztery cechy zbiorów danych z niejawnymi ocenami, ze względu na które metody opracowane dla zbiorów danych z jawnymi ocenami nie powinny być bezpośrednio stosowane w ich przypadku.

1. **Brak negatywnych ocen.** Oceny niejawne zwykle traktowane są jako pozytywne, ponieważ trudno jest rzetelnie wywnioskować z nich negatywną preferencję wobec przedmiotu. Modele stworzone z myślą o jawnych ocenach koncentrują się zwykle wyłącznie na ocenionych przedmiotach, ignorując nieocenione. W przypadku ocen niejawnych, obserwowane interakcje nie są w stanie w pełni opisać preferencji użytkownika. Konieczne jest zatem wykorzystanie informacji o nieocenionych przedmiotach, co można uczynić zakładając, że przedmioty oceniane negatywnie występują częściej wśród przedmiotów nieocenionych.
2. **Niejawne oceny są z natury zaszumione.** Przykładowo, zakupione przedmioty nie zawsze podobają się użytkownikowi.
3. **Wartości numeryczne ocen jawnych świadczą o *preferencji*, zaś wartości numeryczne ocen niejawnych świadczą o *pewności*.** Celem modeli rekomendacji jest

trafne przewidywanie preferencji użytkowników. W przypadku ocen jawnych, są one wprost wyrażane przez użytkowników przez oceny przedmiotów w skali liczbowej. W przypadku ocen niejawnych za wartość numeryczną oceny możemy przyjąć liczbę odtworzeń danej audycji telewizyjnej. W tej sytuacji autor może codziennie oglądać program informacyjny, ponieważ jest emitowany o dogodnej dla niego porze. Mimo to, może bardziej preferować program, który ogląda rzadziej ze względu na mniej dogodną dla niego porę. Wysoka liczba odtworzeń programu świadczy o pewnej *preferencji* użytkownika (przykładowo jednokrotne odtworzenie mogłoby być przypadkowe), lecz niekoniecznie świadczy o sile tej *preferencji* w stosunku do programów wyświetlanych rzadziej.

4. **Wybór metryki ewaluacji w przypadku ocen jawnych jest zwykle bardziej oczywisty.** Możemy bowiem liczbowo porównać estymowaną ocenę z oceną rzeczywistą, na przykład za pomocą pierwiastka błędu średniokwadratowego.

W prowadzanych badaniach użytkownicy nie wyrażają swoich preferencji poprzez oceny jawne. Z tego powodu badamy i opracowujemy metody rekomendacji odpowiednie dla zbiorów danych z niejawnymi ocenami.

### 2.1.2. Podział systemów w zależności od metody uczenia się rangowania

Metody skoncentrowane na optymalizacji kolejności wygenerowanych przedmiotów określane są jako metody uczenia się rangowania (ang. *Learning to Rank (LTR)*, [7]) i adresują one rankingowe sformułowanie problemu rekomendacji. W pracy [115] porównano skuteczność 87 metod uczenia się rangowania. Wyróżniamy następujące ich typy: punktowe (ang. *pointwise*), oparte na parach (ang. *pairwise*) oraz oparte na listach (ang. *listwise*). Różnica między nimi sprowadza się do wykorzystywanej w procesie uczenia funkcji straty.

**Przykład 2.1.** W celu zilustrowania różnicy między tymi podejściami posłużymy się przykładem zadania rangowania przedmiotów $i_1$, $i_2$, $i_3$ oraz $i_4$ dla zadanego użytkownika $u$. Przyjmijmy, że estymowane wyniki ocen tych przedmiotów obliczane są za pomocą wzoru $r_{ui} = f(u, i, \mathbf{\Theta})$, dla $i \in \{i_1, i_2, i_3, i_4\}$, gdzie $f$ jest funkcją przyjmującą wartości rzeczywiste, a $\mathbf{\Theta}$ oznacza listę parametrów modelu. Funkcja $f$ zależy od wybranego modelu rekomendacji. Przykładowo, w jednym z podejść faktoryzacji macierzy [74], funkcja $f$ to iloczyn skalarny między wektorem parametrów reprezentujących zadanego w pierwszym argumencie użytkownika a zadanym w drugim argumencie przedmiotem.

W przypadku podejścia punktowego mamy cztery przykłady uczące: $(u, i_1)$, $(u, i_2)$, $(u, i_3)$ oraz $(u, i_4)$. Problem ten jest klasycznym problemem regresji lub klasyfikacji. Przykładem modelu wykorzystującego tę metodę jest wspomniany wcześniej ALS [64] wdrożony w serwisach Pracodawcy, który szczegółowo definiujemy w podrozdziale 4.1.1.

W przypadku podejścia opartego na parach mamy sześć przykładów uczących $(u, i_1, i_2)$, $(u, i_1, i_3)$, $(u, i_1, i_4)$, $(u, i_2, i_3)$, $(u, i_2, i_4)$ oraz $(u, i_3, i_4)$. Możemy każdy z tych przypadków zaklasyfikować jako pozytywny, bądź negatywny w zależności od interakcji użytkownika $u$ z zadanymi przedmiotami. W tym przypadku zadanie staje się problemem klasyfikacji binarnej. W podrozdziale 4.1.2 definiujemy znane funkcje straty oparte na parach: BPR [104] oraz WARP [129]. W podrozdziale 7.5.2 proponujemy własne metody.

W przypadku podejścia opartego na listach mamy tyle przykładów uczących, ilu jest użytkowników: w rozważanym przykładzie jest to jeden przykład uczący $(u, i_1, i_2, i_3, i_4)$. Pierwszą zaproponowaną metodą tego typu jest ListNet [20]. W podejściu tym lista ocen każdego z przedmiotów przekształcana jest do rozkładu prawdopodobieństwa. W ten sam sposób przekształcana jest także lista predykcji ocen generując drugi rozkład prawdopodobieństwa. Następnie funkcja straty wyliczana jest za pomocą entropii krzyżowej między tymi rozkładami.

Zauważmy, że w przedstawionym przykładzie, niezależnie od przyjęcia metody uczenia się rangowania, wartość predykcji oceny przedmiotu $i$ przez użytkownika $u$, nie jest zależna od tego czy generujemy predykcje także dla innych przedmiotów. Predykcje ocen wielu przedmiotów wykorzystywane są jedynie w funkcji straty.

## 2.2. Wyzwania dla systemów rekomendacji

W sekcji wymieniamy kilka najbardziej powszechnych wyzwań związanych z modelami rekomendacji. Zostaną one omówione w kolejnych akapitach.

Problem **zimnego startu** (ang. *cold start problem*, [87]) rozumiany jest jako trudność w dostarczaniu rekomendacji użytkownikom nieposiadającym interakcji z przedmiotami, bądź rekomendowanie przedmiotów, z którymi nie wszedł w interakcje żaden użytkownik. Sytuacje te są szczególnie problematyczne w przypadku modeli wspólnej filtracji, ponieważ modele te nie wykorzystują dodatkowych informacji o użytkownikach ani przedmiotach, a bazują wyłącznie na interakcjach. Problemem zimnego startu nazywane są czasem także sytuacje, w których liczba interakcji z udziałem danego użytkownika lub przedmiotu jest mała – brak jednak tutaj precyzyjnie wyznaczonej liczby (w pracy [134] autorzy podają przykład co najwyżej 10 interakcji). W sytuacji niewielkiej liczby interakcji modele wspólnej filtracji są zwykle w stanie dostarczać spersonalizowane rekomendacje. Jednak jakość modelu względem liczby dostępnych interakcji może zmieniać się w różny sposób w zależności od użytego modelu rekomendacji, co pokażemy w dalszej części pracy.

Kolejnym, powiązanym wyzwaniem jest **rzadkość macierzy interakcji**. W większości praktycznych zastosowań średnia liczba interakcji użytkowników jest niewielka w porównaniu z liczbą dostępnych przedmiotów. Przykładowo w zbiorze MovieLens 20M [60] około 99,47% par (użytkownik, przedmiot) nie weszło ze sobą w interakcję. W rozdziale 3

przedstawiamy opublikowany przez nas zbiór danych, w którym liczba ta wynosi około 99,9923%.

W praktycznych zastosowaniach modeli rekomendacji występuję najczęściej wiele **obciążeń** (ang. *biases*). Niektóre z nich wymieniamy poniżej [23].

- Obciążenie związane z selekcją (ang. *selection bias*) wynika z nieprzypadkowego sposobu wyboru przez użytkowników przedmiotów, które oceniają. Pokazano, że użytkownicy chętniej wystawiają oceny szczególnie negatywne lub szczególnie pozytywne.
- Obciążenie związane z ekspozycją (ang. *exposure bias*) jest konsekwencją prezentowania użytkownikom przedmiotów pochodzących z pewnego działającego systemu (silnika wyszukiwania czy systemu rekomendacyjnego).
- Obciążenie związane z potrzebą użytkowników do uzyskania zgodności z innymi użytkownikami (ang. *conformity bias*) zostało zaobserwowane w badaniach pokazujących, że informacje o ocenach innych użytkowników wpływają na oceny wystawiane przez kolejnych użytkowników.
- Obciążanie związane z pozycją przedmiotu na liście przedmiotów rekomendowanych (ang. *position bias*) wynika z obserwacji, że użytkownicy częściej wchodzą w interakcje z przedmiotami prezentowanymi na początku listy.
- Obciążenie związane z popularnością (ang. *popularity bias*) odnosi się do obserwacji, że wiele modeli rekomendacji ma tendencje do częstego rekomendowania najbardziej popularnych przedmiotów.

Innym problemem jest **zmienność dostępnych o użytkowniku informacji**. Wraz z kolejnymi interakcjami użytkowników model rekomendacji może aktualizować rekomendacje z opóźnieniem, bądź w czasie rzeczywistym. Wykorzystywany model może być trenowany w regularnych odstępach czasu (na przykład raz dziennie), bądź wagi modelu mogą być aktualizowane po każdej interakcji. Temat ten rozwiniemy w rozdziale 5.

Powiązanym wyzwaniem jest identyfikacja **zmiennych w czasie preferencji użytkownika** (ang. *concept drift* [120]). Przykładowo użytkownicy poszukujący prezentów świątecznych dla różnych osób, mogą wielokrotnie zmieniać kategorię poszukiwanych produktów.

Innym ważnym aspektem jest **prywatność** użytkowników [7]. Modele rekomendacji wykorzystują informacje o użytkownikach bądź ich interakcje, które mogą ujawniać ich poglądy polityczne, religijne czy orientację seksualną. Dbałość o prywatność danych użytkowników stanowi wyzwanie w procesie wdrażania rozwiązań, jak i również podczas publikowania zbiorów danych użytych w badaniach.

Szereg wyzwań dla systemów rekomendacji związany jest także z wyborem sposobu oraz metryki **ewaluacji**. Poświęcamy tym zagadnieniom kolejny podrozdział.

## 2.3. Metody ewaluacji modeli rekomendacyjnych

Ewaluacja modeli rekomendacji jest złożonym zadaniem, któremu poświęcone zostało wiele badań [7,109,114]. Metody ewaluacji możemy podzielić na trzy główne kategorie: badania użytkowników, ewaluację online oraz ewaluację offline.

**Badania użytkowników.** W tej formie ewaluacji wybrana grupa użytkowników ocenia przedstawiony im system rekomendacji. Badania takie pozwalają na ocenę wielu aspektów działania systemu, takich jak sposób wyświetlania rekomendacji czy zgodność rekomendacji z profilem użytkownika lub jego oczekiwaniami. Osoby biorące udział w takich badaniach zwykle otrzymują wynagrodzenie. Podejście to posiada następujące wady:

- czas potrzebny na badania jest zwykle wprost proporcjonalny do liczby badanych użytkowników,
- zachowania badanych użytkowników podczas badań mogą różnić się od ich naturalnych zachowań,
- wybrani użytkownicy często nie są reprezentatywną próbką populacji wszystkich użytkowników.

**Ewaluacja online.** W tym przypadku badane są reakcje użytkowników na rekomendacje wygenerowane z ewaluowanych modeli poprzez prezentowanie ich użytkownikom, najczęściej w docelowym kanale dostępu. Metoda to pozbawiona jest wymienionych wyżej wad ewaluacji poprzez badania użytkowników. Ta forma ewaluacji przeprowadzana jest często poprzez testy A/B. W testach tych badani użytkownicy dzieleni są na ustaloną liczbę grup, najczęściej w sposób całkowicie losowy. Każdej grupie przypisany jest pewien system (na przykład model rekomendacji oraz sposób ich prezentowania), a użytkownicy otrzymują rekomendacje wyłącznie z tego systemu podczas całego okresu trwania testów. W ten sposób zadany system może być porównany z systemami prezentowanymi poszczególnym grupom poprzez porównanie zachowań użytkowników w każdej z grup. Poprzez ewaluację online możemy dość precyzyjnie zmierzyć oczekiwany wpływ wdrożenia bądź zmiany modelu rekomendacji na metryki biznesowe, takie jak liczba osób odpowiadających na ogłoszenia, wartość ich zakupów czy czas spędzony w zadanym serwisie. Niestety taka forma ewaluacji również posiada wady:

- jest ona kosztowna ze względu na konieczność wdrożenia ewaluowanego systemu,
- nie jest możliwa ewaluacja bardzo dużej liczby systemów (przykładowo setek wersji modeli różniących się wartościami hiperparametrów), ze względu na ograniczoną liczbę użytkowników,
- wpływa negatywnie na użytkowników serwisu, jeśli ewaluowany wariant jest gorszy od bazowego.

Wady te nie występują w przypadku ewaluacji offline.

**Ewaluacja offline.** Bazuje ona na historycznych danych użytkowników, lecz nie wymaga ich bezpośredniego udziału. Najczęściej w metodach tych zbiór interakcji użytkowników dzielony jest na zbiór treningowy i testowy. Na podstawie zbioru treningowego generowane są rekomendacje dla użytkowników, a ich skuteczność ewaluowana jest na zbiorze testowym. Z tego powodu możliwa jest ewaluacja proponowanych metod w różnych domenach poprzez wykorzystanie odpowiednich zbiorów danych. W przeciwieństwie do ewaluacji online, w wyniku ewaluacji offline trudno jest oszacować rzeczywisty wpływ na metryki biznesowe, których optymalizacja jest ostatecznym celem usprawniania metod rekomendacji. Poniżej prezentujemy powody tych trudności.

- **Podczas ewaluacji offline jesteśmy w stanie ocenić trafność rekomendacji jedynie dla przedmiotów, z którymi użytkownik wszedł w interakcję.** W praktyce, szczególnie w przypadku zbiorów danych z niejawnymi ocenami, przyjmuje się, że przedmioty z którymi użytkownik nie wszedł w interakcje nie są dla niego właściwymi rekomendacjami. Najczęściej stosowane miary jakości, traktują jako sukces te rekomendowane przedmioty, z którymi użytkownik wszedł w interakcje w zbiorze testowym. W ten sposób istnieje ryzyko niskiej oceny modelu rekomendującego przedmioty, których użytkownik sam prawdopodobnie by nie odkrył, lecz oceniłby je wysoko. Gdybyśmy mieli pełną wiedzę na temat preferencji co najmniej części użytkowników względem wszystkich przedmiotów, moglibyśmy na podstawie podzbioru tych preferencji generować rekomendacje dla tych użytkowników, a podczas ewaluacji nie bylibyśmy zmuszeni do rozstrzygania jakości rekomendacji przedmiotów, wobec których nie znamy preferencji użytkownika. Niestety taki scenariusz jest mało realny ze względu na dużą liczbę przedmiotów w większości praktycznych zastosowań.
- **Historyczne interakcje użytkowników najczęściej są konsekwencją miejsc i sposobów prezentacji rekomendacji użytkownikom (obciążenie związane z ekspozycją).** Przykładowo ogłoszenia promowane mogą być bardziej popularne ze względu na ich wyraźniejszą ekspozycję. Ogłoszenia zawierające pewne słowa kluczowe mogą pojawiać się wyżej w wynikach wyszukiwarki. Największym zagrożeniem dla wiarygodnej ewaluacji są interakcje będące następstwem rekomendacji z wdrożonych systemów rekomendacyjnych. Wdrożone modele mogą uzyskiwać lepsze wyniki podczas ewaluacji offline, nie ze względu na ich skuteczność, lecz ze względu na częstsze występowanie interakcji z rekomendowanymi przedmiotami w zbiorze testowym.
- **Metryki biznesowe dotyczą często ogólnego zachowania użytkowników w serwisie**. Przykładami takich metryk jest częstotliwość wizyt użytkowników w serwisie, czy średnia liczba odwiedzonych przez nich ogłoszeń. W przypadku ewaluacji online zbadanie wpływu zmiany modelu rekomendacji na takie metryki jest znacznie prostsze niż w przypadku ewaluacji offline.

**Tabela 2.1.** Tabela przedstawia przykładowe interakcje użytkowników z przedmiotami w zbiorze testowym, wygenerowane dla tych użytkowników rekomendacje oraz odpowiadające im wartości metryk ewaluacji

| | Użytkownik 0 | Użytkownik 1 | Użytkownik 2 | **Wszyscy użytkownicy** |
|---|---|---|---|---|
| Interakcje w zbiorze testowym | [6] | [4] | [0,2,7] | ... |
| Rekomendacje | [3,6,5] | [4,6,7] | [4,7,0] | ... |
| Precision@3 | 0,33 | 0,33 | 0,67 | **0,44** |
| Recall@3 | 1,00 | 1,00 | 0,67 | **0,89** |
| HR@3 | 1,00 | 1,00 | 1,00 | **1,00** |
| MRR@3 | 0,50 | 1,00 | 0,50 | **0,67** |
| NDCG@3 | 0,63 | 1,00 | 0,53 | **0,72** |
| MAP@3 | 0,50 | 1,00 | 0,39 | **0,63** |
| LAUC@3 | 0,75 | 1,00 | 0,42 | **0,72** |
| Pokrycie zbioru testowego | - | - | - | **0,8** |
| Entropia Shannona | - | - | - | **1,35** |
| Indeks Giniego | - | - | - | **0,36** |

W kolejnych podrozdziałach zdefiniujemy metryki, które wykorzystaliśmy do ewaluacji modeli rekomendacji przedstawionych w pracy.

### 2.3.1. Metryki dokładności w ewaluacji offline

Metryki dokładności rekomendacji są najczęściej głównym kryterium oceny modelu rekomendacji podczas ewaluacji offline. W przypadku podejścia predykcyjnego, koncentrują się one na minimalizacji błędu między predykowanymi a rzeczywistymi ocenami. Często stosowaną metryką jest pierwiastek błędu średniokwadratowego [16]. Powszechniejszym podejściem jest podejście rankingowe, w którym oceniana jest liczba oraz kolejność przedmiotów ze zbioru testowego wśród przedmiotów rekomendowanych. Tamm, Damdinov oraz Vasilev pokazali [114], że wiele popularnych metryk nie jest jednakowo definiowane w pracach naukowych, a także nie jest jednakowo implementowane w bibliotekach, co prowadzi do trudności w porównywaniu raportowanych wartości metryk. W rozdziale przedstawiamy definicje wskazane przez autorów wspomnianej pracy. Ich wartości dla przykładowego zbioru danych złożonego z trzech użytkowników i ośmiu przedmiotów przedstawione są w tabeli 2.1.

Wszystkie zdefiniowane metryki obliczane są niezależnie dla każdego użytkownika, a finalna wartość metryki dla systemu rekomendacji obliczana jest jako średnia arytmetyczna tych wartości. Z tego powodu w definicjach przedstawiamy jedynie wzory

na wartości metryk dla pojedynczego użytkownika $u$, co oznaczamy przez metric($u$). Wszystkie przedstawione metryki zależą od liczby $k$ rekomendacji, na których chcemy dokonać ewaluacji. Fakt ten oznaczamy dopiskiem @$k$ dodawanym po nazwie metryki. Polskie nazwy niektórych z tych metryk nie są powszechnie używane. W celu zachowania spójności odwołujemy się do nazw angielskich wszystkich rozważanych metryk.

Przyjmijmy, podobnie jak w [114], następujące oznaczenia:

- $\text{rec}_k(u)$ oznacza listę $k$ przedmiotów rekomendowanych użytkownikowi $u$,
- $\text{rank}(u, i)$ to pozycja przedmiotu $i$ na liście rekomendowanych przedmiotów $\text{rec}_k(u)$,
- $\text{rel}(u)$ oznacza listę relewantnych przedmiotów dla użytkownika $u$, to znaczy takich przedmiotów, których rekomendację uznajemy za sukces (są to zatem te przedmioty, z którymi użytkownik wszedł w interakcję w zbiorze testowym),
- $\mathbb{I}$ oznacza funkcję charakterystyczną zbioru (ang. *indicator function*).

**Precision**

Metryka Precision@$k$ (precyzja przy $k$) mierzy odsetek relewantnych przedmiotów wśród rekomendowanych i zdefiniowana jest następującym wzorem:

$$\text{Precision@}k(u) = \frac{|\text{rel}(u) \cap \text{rec}_k(u)|}{k}.$$

**Recall**

Metryka Recall@$k$ (czułość przy $k$) mierzy odsetek relawantnych przedmiotów, które zostały zarekomendowane:

$$\text{Recall@}k(u) = \frac{|\text{rel}(u) \cap \text{rec}_k(u)|}{|\,\text{rel}(u)|}.$$

**HR**

Zarówno Precision jak i Recall maksymalizują liczbę relewantnych rekomendacji. W rezultacie system, który rekomenduje trzy relewantne przedmioty jednemu użytkownikowi, zaś zero relewantnych przedmiotów drugiemu użytkownikowi, jest przez te metryki lepiej oceniany od systemu dostarczającego po jednej trafnej rekomendacji każdemu z użytkowników. Metryką, która przeciwnie rozstrzyga jakość tych systemów jest HR (Hit rate), która mierzy odsetek użytkowników otrzymujących co najmniej jedną relewantną rekomendację:

$$\text{HR@}k(u) = \mathbb{I}\left[|\text{rel}(u) \cap \text{rec}_k(u)| > 0\right].$$

Wszystkie powyższe metryki ignorują kolejność przedmiotów wśród $k$ generowanych rekomendacji. W praktyce, korzystniejsze wydaje się, aby trafne rekomendacje znajdowały się bliżej pierwszej pozycji. Użytkownik może bowiem zdecydować się nie przeglądać kolejnych rekomendacji, jeśli pierwszych kilka propozycji okaże się nietrafnych. Kolejne metryki adresują ten problem.

### MRR

Metryka MRR (Mean Reciprocal Rank) zwraca uwagę na najniższą pozycję wśród relewantnych rekomendacji. Jeśli wśród rekomendacji nie ma relewantnych przedmiotów, przyjmujemy $\mathrm{MRR@}k(u) = 0$ w przeciwnym wypadku:

$$\mathrm{MRR@}k(u) = \frac{1}{\min\limits_{i \in \mathrm{rel}(u) \cap \mathrm{rec}_k(u)} \mathrm{rank}(u,i)}.$$

Podobnie do metryki HR, dla metryki MRR nie jest istotna (w bezpośredni sposób) liczba relewantnych rekomendacji.

### NDCG

Metryka NDCG@$k$ (Normalized Discounted Cumulative Gain, znormalizowany zdyskontowany zysk skumulowany) uwzględnia zarówno kolejność, jak i liczbę relewantnych rekomendacji. Zdefiniowana jest ona następująco:

$$\mathrm{NDCG}\,@k(u) = \frac{\mathrm{DCG}\,@k(u)}{\mathrm{IDCG}\,@k(u)},$$

gdzie

$$\mathrm{DCG}\,@k(u) = \sum_{i \in \mathrm{rec}_k(u)} \frac{\mathbb{I}[i \in \mathrm{rel}(u)]}{\log_2(\mathrm{rank}(u,i)+1)},$$

zaś $\mathrm{IDCG}\,@k(u)$ oznacza wartość $\mathrm{DCG}\,@k(u)$ idealnego systemu rekomendacji (czyli systemu, w którym pierwsze $|\mathrm{rel}(u)|$ rekomendacji jest relewantnych). Zauważmy, że metryka NDCG@$k$ przyjmuje wartości w przedziale $[0,1]$. Istnieje również wariant metryki NDCG, w którym brana jest pod uwagę liczbowa wartość wystawianych ocen [114].

### MAP

Innym sposobem na uwzględnienie liczby i kolejności generowanych rekomendacji jest uśrednienie wartości metryki Precision@$k$ dla różnych wartości liczby $k$. Takie podejście stosowane jest w metryce MAP@$k$ (Mean Average Precision, średnia dokładność uśredniona) zdefiniowanej następująco:

$$\mathrm{MAP}\,@k(u) = \frac{1}{\min(k, |\mathrm{rel}(u)|)} \sum_{i \in \mathrm{rec}_k(u)} \mathbb{I}[i \in \mathrm{rel}(u)]\,\mathrm{Precision@rank}(u,i)(u).$$

### LAUC

Popularną metodą ewaluacji modeli klasyfikacji binarnej jest metryka AUC (ang. *area under the curve*), która mierzy pole pod wykresem krzywej ROC (ang. *receiver operating characteristic*) [19]. Krzywa ROC przedstawia wartości metryki TPR (ang. *true positive rate*) umieszczone na osi Y względem wartości metryki FPR (ang. *false positive rate*)

umieszczonych na osi X. Wartości par (TPR, FPR) uzyskuje się poprzez zmianę wartości progowej, powyżej której obserwacje klasyfikowane są jako pozytywne. Podejście to może zostać zastosowane także w systemach rekomendacji. W tym przypadku obserwacje zaklasyfikowane jako pozytywne to te przedmioty, które rekomendowane są użytkownikowi, zaś pozostałe przedmioty traktujemy jak zaklasyfikowane jako negatywne. Różne wartości par (TPR, FPR) uzyskiwane są poprzez wybór różnych wartości liczby rekomendowanych przedmiotów. W praktycznych zastosowaniach systemów rekomendacji najczęściej co najwyżej kilkadziesiąt przedmiotów jest rekomendowanych użytkownikowi, więc kolejność pozostałych przedmiotów jest nieistotna i nie powinna mieć wpływu na metrykę ewaluacji. W celu zaadresowania tego problemu zaproponowana została metryka LAUC@$k$ [106]. Metryka ta różni się od AUC definicją krzywej, pod którą mierzone jest pole. Do wykreślenia krzywej wykorzystywane są jedynie pary (TPR, FPR) uzyskane poprzez wybór liczby rekomendowanych przedmiotów nie większej od zadanej wartości $k$. Pozostała część krzywej powstaje poprzez połączenie punktu (TPR, FPR) dla liczby rekomendacji równej $k$, z punktem (1,1). Podobnie jak w przypadku innych metryk, wartość metryki LAUC jest obliczana dla każdego użytkownika niezależnie, a finalna wartość jest średnią arytmetyczną uzyskanych wartości. Podczas obliczania metryki AUC (a dokładniej wyznaczania FPR) przyjmujemy, że liczba dostępnych przedmiotów równa jest liczbie różnych przedmiotów występujących w zbiorze testowym. Ograniczyliśmy się do rozważania przedmiotów ze zbioru testowego z następujących powodów:

- liczba dostępnych przedmiotów w katalogu jest zmienna w czasie,
- liczba przedmiotów w katalogu nie powinna zależeć od zbioru treningowego, ponieważ nie moglibyśmy porównać modeli trenowanych na różnych zbiorach treningowych (np. sprawdzić ile dni interakcji użytkowników powinniśmy uwzględniać),
- chcieliśmy umożliwić ewaluację za pomocą wygenerowanych rekomendacji i zbioru testowego, bez konieczności przechowywania dodatkowych informacji na temat dostępnych przedmiotów,
- w przypadku naszego zbioru danych, większość dostępnych przedmiotów znajduje się w zbiorze testowym, co ogranicza możliwy wpływ tej modyfikacji na wartość metryki.

Podobne ograniczenie wprowadziliśmy także do omawianych w dalszej części metryk pokrycia.

Poszukuje się metryk ewaluacji offline, które są najbardziej skorelowane z metrykami biznesowymi, co często zależy od konkretnego zastosowania. W przypadku rekomendacji ofert pracy, serwis Indeed, jeden z liderów branży, przeprowadził analizę korelacji kilku znanych metryk ewaluacji offline z optymalizowaną metryką biznesową (stosunkiem liczby aplikacji na rekomendowane oferty pracy do liczby rekomendacji wyświetlonych użytkownikom) [94]. Najbardziej skorelowaną okazała się metryka Precision@$k$. Z tego powodu, przyjęliśmy tę metrykę jako główne kryterium wyboru modelu w serwisach

ogłoszeniowych Pracodawcy. Raportujemy jednak wartości wszystkich zdefiniowanych wyżej metryk.

### 2.3.2. Pozostałe metryki w ewaluacji offline

Poza dokładnością istnieje wiele aspektów modeli rekomendacji, których optymalizacja może przynieść pozytywne skutki biznesowe, szczególnie w dłuższym horyzoncie czasu [7]. Wymieniamy je poniżej.

- **Pokrycie** (ang. *coverage*) - metryki w tym obszarze koncentrują się na liczbie użytkowników, dla których system jest w stanie wygenerować co najmniej $k$ rekomendacji (pokrycie użytkowników), bądź liczbie przedmiotów, które są rekomendowane przynajmniej jednemu użytkownikowi (pokrycie katalogu).
- **Nowość** (ang. *novelty*) - w tym kontekście wyżej oceniane są systemy, które rekomendują przedmioty dotychczas nieznane użytkownikowi.
- **Zdolność do przypadkowych odkryć** (ang. *serendipity*) - własność ta jest silniejsza od nowości, ponieważ wymaga dodatkowo, aby rekomendacje były nieoczywiste. Przykładowo rekomendacja nieznanej użytkownikowi kolejnej części filmu, który wcześniej obejrzał, prawdopodobnie zostanie oceniona negatywnie przez metryki z tej grupy.
- **Różnorodność** (ang. *diversity*) ocenia zróżnicowanie między przedmiotami rekomendowanymi danemu użytkownikowi. Dostarczanie zróżnicowanych rekomendacji może zwiększyć szansę na dostarczenie użytkownikowi przynajmniej jednej trafnej rekomendacji.
- **Odporność** (ang. *robustness*) - metryki te oceniają podatność modelu na ataki polegające na generowaniu nieprawdziwych danych takich jak fałszywe oceny przedmiotów, bądź wyświetlenia strony.
- **Skalowalność** (ang. *scalability*) - aspekt ten dotyczy możliwości wdrożenia modelu w zależności od rozmiaru zbioru danych. Często oceniany jest czas oraz pamięć potrzebna na trenowanie modelu oraz generowanie rekomendacji.

Wśród wymienionych wyżej grup, w dalszej części pracy raportujemy jedynie metryki skalowalności oraz pokrycia. Zdecydowaliśmy się rekomendować użytkownikom jedynie przedmioty, których dotychczas nie wyświetlili, więc nie mierzymy metryk nowości. Właściwe zdefiniowanie i mierzenie metryk z pozostałych grup jest czasochłonnym zadaniem w stosunku do oczekiwanego przez nas wpływu. Poniżej prezentujemy wykorzystywane przez nas metryki pokrycia. Podobnie jak w przypadku metryk dokładności, wartości tych metryk dla przykładowego zbioru danych przedstawione są w tabeli 2.1.

#### Pokrycie zbioru testowego

Pokrycie zbioru testowego zdefiniowane jest jako stosunek liczby przedmiotów rekomendowanych co najmniej jednemu użytkownikowi (i występujących w zbiorze testowym) do liczby przedmiotów w zbiorze testowym. Metryka ta jest równoważna klasycznej definicji pokrycia katalogu w przypadku, gdy wszystkie przedmioty pojawiają się co najmniej raz w zbiorze testowym. Powody, dla których ograniczyliśmy się do rozważania przedmiotów ze zbioru testowego opisaliśmy przedstawiając metrykę LAUC w podrozdziale 2.3.1. Podobne ograniczenie przyjęliśmy w przypadku pozostałych metryk pokrycia.

#### Entropia Shannona

Entropia Shannona (ang. *Shannon entropy*) [107] zdefiniowana jest wzorem:

$$H = -\sum_{i=1}^{n} p(i) \ln p(i),$$

gdzie $n$ oznacza liczbę przedmiotów w zbiorze testowym, $p(i)$ oznacza stosunek liczby rekomendacji przedmiotu $i$ do liczby wszystkich rekomendacji przedmiotów ze zbioru testowego. Metryka ta przyjmuje wartość 0, gdy tylko jeden, ten sam przedmiot rekomendowany jest wszystkim użytkownikom, zaś wartość $\ln n$, gdy wszystkie przedmioty w zbiorze testowym rekomendowane są równie często.

#### Indeks Giniego

Indeks Giniego (ang. *Gini index*) [107] definiujemy wzorem:

$$G = \frac{1}{n-1} \sum_{j=1}^{n} (2j - n - 1) p\left(i_j\right),$$

gdzie $i_1, \cdots, i_n$ to ciąg przedmiotów posortowany względem rosnącej wartości $p(i)$, zaś pozostałe oznaczenia są identyczne jak w definicji entropii. Metryka ta przyjmuje wartość 0, gdy wszystkie przedmioty ze zbioru testowego rekomendowane są równie często, zaś wartość 1, gdy tylko jeden, ten sam przedmiot rekomendowany jest wszystkim użytkownikom.

### 2.3.3. Metryki dopasowania do preferencji

Poza klasycznymi metrykami ewaluacji offline, w ramach ewaluacji modeli rekomendacji ofert pracy stosowaliśmy również metryki specyficzne dla naszego zadania. Pewien podzbiór użytkowników dostarcza nam informacji na temat swoich preferencji dotyczących przyszłej pracy (odnośnie lokalizacji, płacy oraz typu i rodzaju umowy). Podczas ewaluacji offline sprawdzamy jak często generowane przez nas rekomendacje spełniają preferencje użytkowników. W rozprawie nie raportujemy tych metryk, ponieważ do ich

wyliczenia wykorzystywane są dane niedostępne w opublikowanym przez nas zbiorze danych. Chcemy jednak nadmienić, iż modele wspólnej filtracji, mimo iż nie wykorzystują informacji o preferencjach użytkowników, uzyskują niewiele gorsze wyniki od modeli wykorzystujących te informacje. Jest to kolejny powód, dla którego ograniczamy się do rozważania modeli wspólnej filtracji.

### 2.3.4. Testy A/B

W przeprowadzonych przez nas testach A/B każdy użytkownik został losowo przypisany do jednego z testowanych wariantów. Po zakończeniu eksperymentu oznaczyliśmy każdego z użytkowników jako przypadek pozytywny bądź negatywny na podstawie ustalonej w danym eksperymencie metryki biznesowej. Przykładowo, przypadki pozytywne mogli stanowić użytkownicy, którzy aplikowali na co najmniej jedną ofertę pracy w trakcie trwania eksperymentu. W celu wskazania najlepszego modelu stosowaliśmy test niezależności $\chi^2$ Pearsona [91]. Test ten stosowany jest do zmiennych jakościowych, a jego hipoteza zerowa głosi, że dwie zmienne losowe są niezależne. W naszym przypadku przestrzenią zdarzeń elementarnych jest zbiór $n$ użytkowników, zaś rozważane zmienne losowe to odpowiednio funkcje przypisujące użytkownikom wariant eksperymentu oraz wartość metryki biznesowej (pozytywna bądź negatywna). Przyjmijmy, że pierwsza z tych zmiennych przyjmuje $k$ różnych wartości, zaś druga zmienna $l$ różnych wartości (w naszym przypadku $l = 2$). Niech $O_{ij}$ oznacza liczbę obserwowanych przypadków, dla których pierwsza zmienna przyjmuje $i$-tą wartość, zaś druga zmienna przyjmuje $j$-tą wartość. Przy prawdziwości hipotezy zerowej możemy spodziewać się, że wartość $O_{ij}$ będzie bliska oczekiwanej liczebności:

$$E_{ij} = \frac{\sum_{m=1}^{k} O_{mj} \sum_{m=1}^{l} O_{im}}{n}.$$

Pokazano, że statystyka testowa:

$$\chi^2 = \sum_{i=1}^{k} \sum_{j=1}^{l} \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}}$$

ma rozkład $\chi^2$ z $(k-1)(l-1)$ stopniami swobody.

W eksperymentach przejęliśmy poziom istotności równy 0,05. Raportujemy również $p$-wartość wyliczoną na podstawie wartości statystyki testowej.

## 2.4. Podsumowanie

W rozdziale zdefiniowaliśmy pojęcie systemów rekomendacji oraz przedstawiliśmy najważniejsze ich typy, wśród których znajdują się modele wspólnej filtracji będące

obiektem badań prowadzonych w rozprawie. Wyjaśniliśmy wpływ braku ocen jawnych na wybór modelu rekomendacyjnego, co ma miejsce w przypadku rozważanego przez nas problemu rekomendacji ofert pracy. Przedstawiliśmy typy metod uczenia się rangowania, co pozwoli lepiej zrozumieć proces uczenia metod opisywanych w dalszej części rozprawy.

Przedstawiliśmy także najbardziej powszechne wyzwania związane z modelami rekomendacji, takie jak problem zimnego startu, rzadkość macierzy interakcji, występowanie różnego rodzaju obciążeń, zmienność dostępnych danych, zmienność preferencji użytkowników, prywatność i ewaluacja. Wiele z tych wyzwań adresujemy w rozważanych z rozprawie metodach.

Ostatni podrozdział poświęciliśmy przedstawieniu typów oraz metod ewaluacji systemów rekomendacyjnych. Przedstawiliśmy definicje wykorzystywanych w rozprawie metryk dokładności i pokrycia stosowanych podczas ewaluacji offline. Zdefiniowaliśmy także wykorzystaną metodę przeprowadzania oraz ewaluacji testów A/B.

Rozdział 3

# Zbiór danych

Głównym celem badań opisywanych w rozprawie było wdrożenie skutecznych metod rekomendacji w serwisach Pracodawcy. Przygotowanie odpowiedniego zbioru danych na podstawie dostępnych w przedsiębiorstwie informacji było jednym z kluczowych zadań do realizacji tego celu. Autor, za zgodą Pracodawcy, podjął również dodatkowy wysiłek opublikowania przygotowanych danych. Publiczny zbiór danych umożliwia poszukiwanie lepszych rozwiązań dla problemu rekomendacji ofert pracy przez innych badaczy, co może przynieść korzyści zarówno naukowe, biznesowe jak i społeczne (ułatwiając ludziom znajdowanie pracy). Ponadto, wiele wyników prezentowanych w rozprawie wykorzystuje opublikowany zbiór danych dzięki czemu możliwa jest ich weryfikacja przy użyciu opublikowanego kodu źródłowego[1].

Celem rozdziału jest przedstawienie opublikowanego zbioru danych, porównanie z istniejącymi zbiorami danych stosowanymi do ewaluacji modeli rekomendacji ofert pracy, a także przedstawienie licznych statystyk opisujących specyfikę zbioru. Opisujemy również zastosowaną w naszych badaniach metodę podziału przedstawionego zbioru danych na zbiór treningowy i testowy.

## 3.1. Opis zbioru danych

Opublikowany zbiór danych OLX Jobs Interactions zawiera 65 502 201 interakcji wykonanych na serwisie `olx.pl` przez 3 295 942 użytkowników, którzy weszli w interakcję z 185 395 ogłoszeniami o pracę w ciągu dwóch tygodni 2020 roku. Zbiór ten jest publicznie dostępny na platformie Kaggle[2].

Fragment opublikowanego zbioru danych przedstawiony jest w tabeli 3.1. Każdy wiersz zbioru danych reprezentuje interakcję wykonaną przez danego użytkownika w odniesieniu do danego ogłoszenia w danym momencie wyrażonym w czasie POSIX. W zbiorze danych reprezentowane są następujące typy interakcji:

[1] `https://github.com/rob-kwiec/olx-jobs-recommendations`, dostęp: 2023-12-02

[2] `https://www.kaggle.com/olxdatascience/olx-jobs-interactions`, dostęp: 2023-11-18

**Tabela 3.1.** Przykładowe wiersze ze zbioru danych OLX Jobs Interactions

| Użytkownik | Przedmiot | Typ interakcji | Znacznik czasu |
|---|---|---|---|
| 1745587 | 168661 | click | 1582216025 |
| 843008 | 62838 | click | 1582485868 |
| 14285 | 30469 | bookmark | 1582247367 |
| 1142944 | 80122 | click | 1581805847 |
| 2835659 | 23728 | chat_click | 1582397836 |

Źródło: publikacja autora [78]

- click: użytkownik odwiedził stronę przedmiotu,
- bookmark: użytkownik dodał przedmiot do zakładek,
- chat_click: użytkownik otworzył czat, aby skontaktować się z ogłoszeniodawcą,
- contact_phone_click_1: użytkownik odkrył numer telefonu ogłoszeniodawcy,
- contact_phone_click_2: użytkownik skorzystał z przycisku przekierowującego na ekran wybierania numeru telefonu do ogłoszeniodawcy,
- contact_phone_click_3: użytkownik skorzystał z przycisku przekierowującego na ekran wiadomości SMS do ogłoszeniodawcy,
- contact_partner_click: użytkownik użył przycisku przekierowującego na zewnętrzną stronę ogłoszeniodawcy,
- contact_chat: użytkownik wysłał wiadomość do ogłoszeniodawcy odnośnie danego ogłoszenia.

Zachowanie poufności ogłoszeń i użytkowników było priorytetem podczas przygotowywania tego zbioru danych. Środki podjęte w celu ochrony prywatności obejmowały następujące elementy:

- identyfikatory użytkowników i przedmiotów zostały zastąpione unikalnymi losowymi liczbami całkowitymi,
- pewna nieujawniona stała liczba całkowita została dodana do wszystkich znaczników czasu,
- odfiltrowano pewną część interakcji,
- dodano pewną liczbę nieprawdziwych interakcji.

Autor nie został upoważniony do ujawniania szczegółów wymienionych wyżej działań.

## 3.2. Porównanie z istniejącymi zbiorami danych

Modele rekomendacji ofert pracy ewaluowane są na różnych zbiorach danych. Nie istnieje zbiór danych, po który sięga większość autorów [38]. Wiele wyników raportowanych jest

na zbiorach, które nie są publicznie dostępne [81, 82, 84, 136]. W rozdziale porównujemy kilka najczęściej wykorzystywanych zbiorów danych.

De Ruijt i Bhulai [33] przeanalizowali artykuły naukowe dotyczące rekomendacji ofert pracy opublikowane między 1 stycznia 2011 roku a 1 stycznia 2021 roku. Zaklasyfikowali oni 87 znalezionych prac względem użytych metod i zbiorów danych. Wyróżnili oni cztery grupy wykorzystywanych zbiorów danych: zbiór danych **CareerBuilder12**, zbiór danych **RecSys16**, zbiór danych **RecSys17** oraz inne zbiory danych. Trzy wymienione zbiory danych zostały udostępnione podczas konkursów organizowanych przez właścicieli serwisów ogłoszeniowych. Dane te były jednak wykorzystywane w badaniach również po ich zakończeniu. Około 32% analizowanych prac wykorzystywało co najmniej jeden z tych zbiorów. Tylko w jednej z analizowanych prac [82] wykorzystano więcej niż jeden zbiór danych dotyczący ofert pracy.

Zbiór danych **CareerBuilder12** został opublikowany w 2012 roku na potrzeby konkursu Job Recommendation Engine Challenge organizowanego przez serwis CareerBuilder[3]. Dane dostępne są na platformie Kaggle[4]. Jest to jeden z najczęściej wykorzystywanych zbiorów danych zawierających interakcje użytkowników z ofertami pracy [82, 93, 127]. Zbiór ten zawiera interakcje użytkowników wykonane podczas 13 tygodni działalności serwisu. Podzielone zostały one na 7 części składających się z kolejnych 13-dniowych okresów. Każdy użytkownik i przedmiot przypisany został do dokładnie jednej części. Jedynie interakcje wykonane w ciągu pierwszych 9 dni każdej części zostały udostępnione. Pozostałe 4 dni każdej części stanowiły dane testowe wykorzystywane do ewaluacji rozwiązań. Dane testowe nie zostały upublicznione po zakończeniu konkursu. W przypadku tego zbioru danych występuje tylko jeden typ interakcji – aplikacja w odpowiedzi na ofertę pracy. Poza interakcjami, w zbiorze dostępne są też informacje na temat cech użytkowników i przedmiotów.

Jak wspomnieliśmy, często wykorzystywane są także zbiory danych **RecSys16** oraz **RecSys17** pochodzące odpowiednio z konkursów RecSys Challenge 2016 [1] oraz RecSys Challenge 2017 [2]. Zbiory te były udostępnione użytkownikom tych konkursów, lecz obecnie nie są dostępne, co potwierdza Maurera i in. [100]. Zbiory te wciąż jednak są często wykorzystywane w pracach naukowych [22, 34, 82, 132]. W celu przedstawienia statystyk tych zbiorów w rozprawie, autor zgłosił prośbę o ich udostępnienie poprzez utworzenie pytań na repozytoriach dedykowanych konkursom[5,6]. Autor nie otrzymał żadnej odpowiedzi, w związku z czym podjął on próbę bezpośredniego kontaktu z organizatorami. Jeden z organizatorów odpowiedział, iż zbiory te zostały udostępnione na ściśle określonych warunkach i nie są już dostępne (*„XING made the data available under strictly limited conditions and those data sets are no longer available."*, email z dnia

[3] `https://www.careerbuilder.com/`, dostęp: 2023-11-18

[4] `https://www.kaggle.com/competitions/job-recommendation`, dostęp: 2023-11-18

[5] `https://github.com/recsyschallenge/2016`, dostęp: 2023-11-18

[6] `https://github.com/recsyschallenge/2017`, dostęp: 2023-11-18

2023-07-27 od prof. M. Larson). Autor nie uzyskał dostępu do tych zbiorów danych. Poniżej przedstawiamy jednak ich opis przygotowany na podstawie literatury.

Zbiór danych **RecSys16** zawiera informacje o interakcjach wykonanych w ciągu 12 tygodni na serwisie Xing[7], będącym portalem ogłoszeniowym z ofertami pracy obsługującym kilkanaście milionów użytkowników [3]. Zbiór ten zawiera ponad 8 milionów interakcji czterech typów: wyświetlenie, dodanie przedmiotu do ulubionych, odpowiedź oraz usunięcie, które stanowią odpowiednio 81%, 2%, 5%, oraz 12% wszystkich interakcji. Usunięcie oznacza ukrycie prezentowanego na liście rekomendacji przedmiotu. Przedmiot taki nie jest więcej rekomendowany użytkownikowi, a w jego miejsce użytkownik otrzymuje inną rekomendację. Ponadto, zbiór ten zawiera informacje o impresjach, tzn. przedmiotach prezentowanych użytkownikom, które niekoniecznie prowadziły do interakcji użytkownika. Dostępne są informacje o ponad miliardzie impresji [90]. W zbiorze tym możliwe jest wystąpienie wielu interakcji pomiędzy zadanym użytkownikiem i przedmiotem, co było analizowane przez uczestników konkursu [138].

Zbiór danych **RecSys17** udostępniony został podczas kolejnej edycji konkursu – RecSys Challenge 2017. W tej edycji rozwiązania użytkowników były oceniane nie tylko podczas ewaluacji offline, lecz także podczas testów online z użytkownikami serwisu Xing. Wprowadzony został także nowy typ interakcji, „Recruiter", która oznaczała zainteresowanie pracodawcy danym kandydatem. Podobnie jak w przypadku zbioru danych RecSys16, część użytkowników i przedmiotów nie posiadała żadnych interakcji w opublikowanym zbiorze danych, przez co wiele rozwiązań koncentrowało się na problemie zimnego startu. Dodatkowo zbiór ten zawiera informacje o ponad 300 milionach impresji [90].

W tabeli 3.2 prezentujemy podstawowe statystyki i charakterystyki omówionych zbiorów. Liczba unikalnych interakcji oznacza liczbę różnych par (użytkownik, przedmiot) oznaczających, że użytkownik wszedł w co najmniej jedną interakcję z danym przedmiotem. Ze względu na brak dostępu do zbiorów danych RecSys16 oraz RecSys17 statystyki dotyczące tych zbiorów pochodzą z prac innych badaczy (m.in. [3, 4, 90]). Niestety autor nie znalazł informacji na temat liczby unikalnych interakcji dla tych zbiorów. W metrykach wykorzystujących tę liczbę, podano górne oszacowanie wynikające z obserwacji, że liczba unikalnych interakcji jest mniejsza od liczby wszystkich interakcji, ponieważ w zbiorach tych istnieli użytkownicy wchodzący w interakcje wielokrotnie z tymi samymi przedmiotami. Ponadto, podobnie do organizatorów konkursów, nie traktujemy impresji jako typu interakcji. W przypadku zbioru CareerBuilder12, ze względu na nieopublikowanie przez organizatorów zbioru testowego, w tabeli 3.2 uwzględniamy jedynie przedmioty oraz użytkowników, dla których istnieje co najmniej jedna interakcja w opublikowanych danych.

[7] `https://www.xing.com/`, dostęp: 2023-11-18

**Tabela 3.2.** Porównanie zbiorów danych wykorzystywanych do trenowania modeli rekomendacji ofert pracy. Literą $D$ oznaczyliśmy odchylenie standardowe

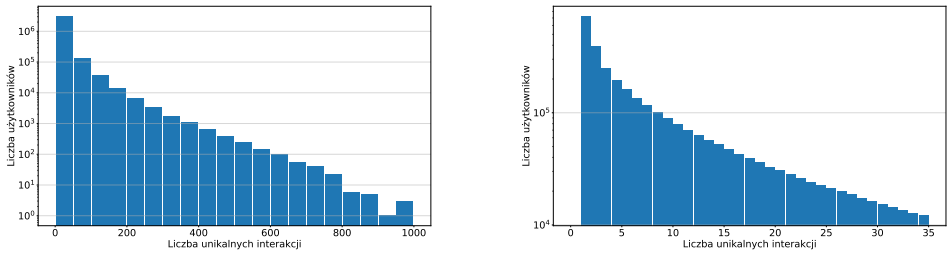
| | **OLX Jobs Interactions** | **CareerBuilder12** | **RecSys16** | **RecSys17** |
|---|---|---|---|---|
| Liczba interakcji | 65,50M | 1,60M | 8,83M | 8,27M |
| Liczba unikalnych interakcji | 47,17M | 1,60M | - | - |
| Liczba użytkowników | 3,30M | 0,32M | 1,37M | 1,50M |
| Średnia liczba unikalnych interakcji na użytkownika | 14,31 ($D$ = 29,23) | 4,99 ($D$ = 11,42) | <6,46 | <5,51 |
| Liczba przedmiotów | 0,19M | 0,37M | 1,36M | 1,31M |
| Średnia liczba unikalnych interakcji na przedmiot | 254,42 ($D$ = 426,02) | 4,38 ($D$ = 8,19) | <6,50 | <6,31 |
| Gęstość macierzy interakcji (%) | 0,0077% | 0,0014% | <0,0005% | <0,0004% |
| Liczba typów interakcji | 8 | 1 | 4 | 5 |
| Znaczniki czasu | ✓ | ✓ | ✓ | ✓ |
| Cechy użytkowników | ✗ | ✓ | ✓ | ✓ |
| Cechy przedmiotów | ✗ | ✓ | ✓ | ✓ |

Poniżej wymieniamy najważniejsze różnice między opublikowanym przez nas zbiorem danych OLX Jobs Interactions a pozostałymi prezentowanymi zbiorami.

- Liczba interakcji użytkowników w zbiorze OLX Jobs Interactions jest o rząd wielkości większa niż w przypadku pozostałych zbiorów.
- Średnia liczba unikalnych interakcji przypadających na przedmiot w zbiorze OLX Jobs Interactions jest około 50 razy większa niż w pozostałych zbiorach.
- Zbiór OLX Jobs Interactions, podobnie jak zbiór CareerBuilder12, jest publicznie dostępny w przeciwieństwie do zbiorów RecSys16 oraz RecSys17.
- W zbiorze OLX Jobs Interactions dostępnych jest aż 8 typów interakcji, więcej niż w pozostałych zbiorach. W szczególności, zbiór CareerBuilder posiada tylko jeden typ interakcji.
- W zbiorze OLX Jobs Interactions, w przeciwieństwie do zbioru CareerBuilder, możliwe jest występowanie wielu interakcji pomiędzy zadanym użytkownikiem a przedmiotem.

Z drugiej strony, wadą udostępnionego przez nas zbioru danych jest brak cech użytkowników i przedmiotów. W przypadku modeli wspólnej filtracji, odpowiadających za znaczną część istniejących modeli rekomendacyjnych, cechy te nie są jednak wykorzystywane.

Autor ma nadzieję, że opublikowany zbiór danych OLX Jobs Interactions pozwoli




**(a)** Wszyscy użytkownicy

**(b)** 90% użytkowników posiadających najmniej unikalnych interakcji

**Rysunek 3.1.** Histogramy liczby unikalnych interakcji z perspektywy użytkowników. Dla każdego użytkownika obliczona została liczba różnych przedmiotów, z którymi wszedł on w interakcję

badaczom na ewaluację istniejących oraz opracowanie nowych metod rekomendacji ofert pracy.

## 3.3. Analiza zbioru danych

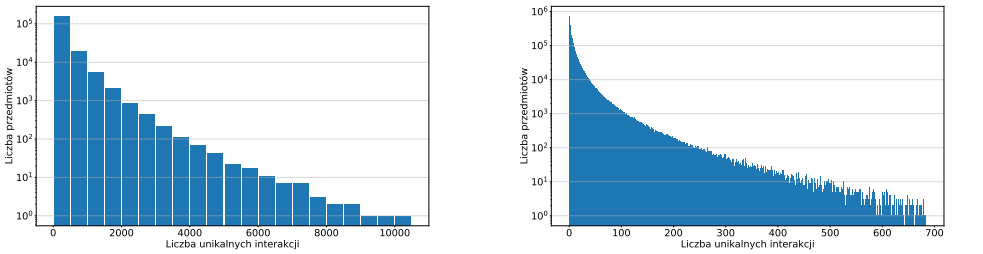
W tabeli 3.2 przedstawiliśmy podstawowe statystyki i cechy zbioru danych OLX Jobs Interactions. W podrozdziale natomiast przedstawiamy bardziej szczegółową analizę tego zbioru. Rozdział ten podzieliliśmy na dwie części: analizę unikalnych interakcji użytkowników oraz analizę wszystkich interakcji.

### 3.3.1. Analiza unikalnych interakcji

Na rysunku 3.1 przedstawiamy histogramy liczby unikalnych interakcji użytkowników. Użyliśmy skali logarytmicznej, ponieważ rozkłady są silnie prawoskośne. Wykres z lewej strony dotyczy wszystkich użytkowników. Dostarczenie trafnych rekomendacji użytkownikowi z małą liczbą interakcji jest często tak samo istotne jak dostarczenie trafnych rekomendacji użytkownikowi posiadającemu ich więcej. Przykładowo, opisane w rozdziale 2.3.1 metryki dokładności nie uwzględniają żadnych cech użytkownika. W przypadku wdrażanych przez nas rozwiązań, podczas ewaluacji online maksymalizujemy liczbę użytkowników odpowiadających na oferty pracy, więc również w tym przypadku użytkownicy traktowani są jednakowo. Jakość naszych rozwiązań zależy zatem w dużym stopniu od ich zdolności do generowania rekomendacji dla większości użytkowników. Kwantyle rozkładu liczby unikalnych użytkowników przedstawione są w tabeli 3.3. Widzimy, że 90% użytkowników weszło w interakcje z co najwyżej 34

**Tabela 3.3.** Rozkład liczby unikalnych interakcji użytkowników

| Rząd kwantyla | 0,01 | 0,10 | 0,25 | 0,50 | 0,75 | 0,90 | 0,99 |
|---|---|---|---|---|---|---|---|
| Wartość | 1 | 1 | 2 | 5 | 14 | 34 | 140 |




**(a)** Wszystkie przedmioty **(b)** 90% najmniej popularnych przedmiotów

**Rysunek 3.2.** Histogram popularności przedmiotów

**Tabela 3.4.** Rozkład popularności przedmiotów

| Rząd kwantyla | 0,01 | 0,10 | 0,25 | 0,50 | 0,75 | 0,90 | 0,99 |
|---|---|---|---|---|---|---|---|
| Wartość | 1 | 1 | 3 | 105 | 319 | 683 | 1993 |

przedmiotami. Z tego powodu na rysunku 3.1 przedstawiamy także histogram liczby unikalnych interakcji użytkowników dla użytkowników posiadających maksymalnie 34 unikalne interakcje. Widzimy zatem, że **zdolność modelu do generowania rekomendacji dla użytkowników posiadających od kilku do kilkunastu interakcji jest kluczowa** dla uzyskania dobrych wyników na zbiorze danych OLX Jobs Interactions.

Niech popularność danego przedmiotu oznacza liczbę różnych użytkowników, która weszła w interakcje z tym przedmiotem. Analogicznie jak w przypadku użytkowników, na rysunku 3.2 oraz w tabeli 3.4 przedstawiamy rozkład popularności przedmiotów. Rozkład ten również jest silnie prawoskośny. Zauważmy, że aż 25% przedmiotów to przedmioty, z którymi w interakcję weszło co najwyżej trzech użytkowników. Duża część z tych przedmiotów mogła być nieaktywna, bądź aktywna jedynie przez krótki czas rozważanego przez nas okresu 14 dni. Informacja o czasie dostępności przedmiotów nie jest dostępna w opublikowanym przez nas zbiorze danych. Zwróćmy jednak uwagę, że w przeciwieństwie do użytkowników, najmniej popularne przedmioty prawdopodobnie nie wpłyną istotnie na jakość modelu rekomendacji, ponieważ dotyczą one stosunkowo

**Tabela 3.5.** Rozkład popularności przedmiotów, z którymi użytkownicy weszli w interakcję. Dla każdej interakcji użytkownika $u$ z przedmiotem $i$ obliczyliśmy popularność przedmiotu $i$

| Rząd kwantyla | 0,01 | 0,10 | 0,25 | 0,50 | 0,75 | 0,90 | 0,99 |
|---|---|---|---|---|---|---|---|
| Wartość | 46 | 178 | 346 | 688 | 1286 | 2167 | 5121 |

niewielkiej liczby interakcji. Tabela 3.5 przedstawia kwantyle z rozkładu popularności przedmiotów, z którymi użytkownicy weszli w interakcję. Widzimy, że interakcje z przedmiotami, których popularność wynosi co najwyżej 46, stanowią jeden procent wszystkich interakcji. Z drugiej strony, połowa interakcji dotyczy przedmiotów, których popularność wynosi co najmniej 688 (stanowiących niecałe 10% najpopularniejszych przedmiotów). Obliczyliśmy ponadto, że 14% użytkowników weszło w interakcje wyłącznie z tymi przedmiotami. Zatem dostarczenie tym użytkownikom trafnych rekomendacji z modeli wspólnej filtracji zależy głównie od jakości reprezentacji, jaką uzyskamy dla najpopularniejszych przedmiotów.

### 3.3.2. Analiza wszystkich interakcji użytkowników

Jak pokazaliśmy w tabeli 3.2 zbiór danych OLX Jobs Interactions zawiera 8 typów interakcji, a użytkownik może wejść w interakcję z tym samym przedmiotem wielokrotnie. W podrozdziale przedstawiamy podstawowe statystyki zbioru danych związane z tymi informacjami.

Średnia liczba interakcji przypadająca na unikalną interakcję wynosi około 1,4. Wiele klasycznych modeli rekomendacji, nie uwzględnia informacji o typie i częstotliwości interakcji. Zaletą przedstawionego zbioru jest możliwość wykorzystania lub opracowania metod, które wykorzystują te dodatkowe, często dostępne w praktycznych zastosowaniach, informacje. Przykładowo, modele omawiane w rozdziale 4 uwzględniają jedynie fakt wystąpienia co najmniej jednej interakcji między użytkownikiem a przedmiotem, natomiast metoda proponowana w rozdziale 6 uwzględnia także typ i częstotliwość interakcji.

Rozkłady liczby interakcji z punktu widzenia użytkownika i przedmiotu przyjmują podobny kształt do rozkładów liczby unikalnych interakcji przedstawionych na rysunkach 3.1 oraz 3.2.

Krotnością unikalnej interakcji między użytkownikiem a przedmiotem nazywać będziemy liczbę interakcji między tym użytkownikiem a przedmiotem. Unikalną interakcję nazwiemy wielokrotną, jeśli jej krotność jest większa niż 1. W analizowanym zbiorze danych interakcje wielokrotne stanowią 19% wszystkich unikalnych interakcji. Ponadto,

**Tabela 3.6.** Rozkład krotności interakcji dla interakcji wielokrotnych

| Krotność interakcji | 2 | 3 | 4 | 5 | 6 | 7-10 | więcej niż 10 |
|---|---|---|---|---|---|---|---|
| Odsetek wszystkich interakcji wielokrotnych | 58,6% | 20,0% | 9,0% | 4,6% | 2,7% | 3,8% | 1,4% |

**Tabela 3.7.** Częstość występowania poszczególnych typów interakcji w zbiorze danych OLX Jobs Interactions

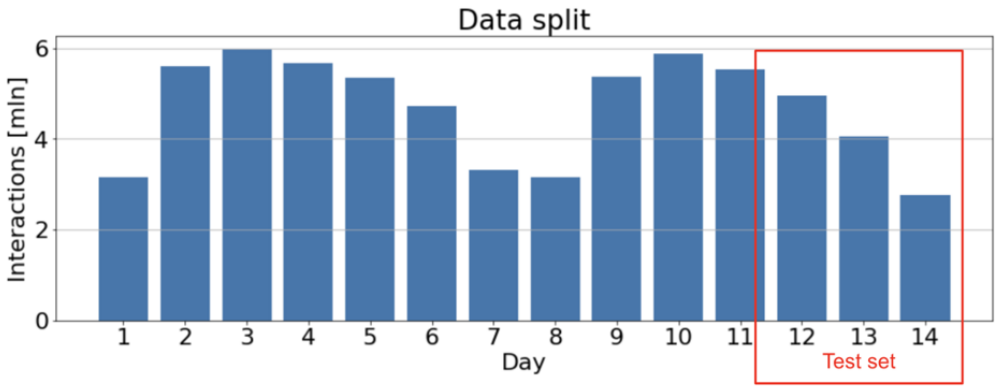
| Typ interakcji | Częstość (%) |
|---|---|
| click | 89,794 |
| contact_phone_click_1 | 2,628 |
| bookmark | 2,511 |
| chat_click | 2,136 |
| contact_chat | 1,448 |
| contact_partner_click | 0,701 |
| contact_phone_click_2 | 0,679 |
| contact_phone_click_3 | 0,103 |

Źródło: publikacja autora [78]

51% użytkowników posiada co najmniej jedną interakcję wielokrotną, zaś interakcje wielokrotne stanowią 22% wszystkich unikalnych interakcji dla tych użytkowników. Liczby te pokazują powszechność występowania interakcji wielokrotnych w naszym zbiorze danych. Zatem uwzględnienie ich przez model rekomendacji może mieć istotny wpływ na działanie systemu. W tabeli 3.6 przedstawiliśmy rozkład krotności interakcji wielokrotnych. Zauważmy, że około 92% interakcji wielokrotnych posiada krotność nie większą niż 5.

W dotychczasowych analizach nie uwzględniliśmy typów interakcji. Tabela 3.7 przedstawia powszechność wystąpienia poszczególnych typów. Zauważmy, że niemalże 90% interakcji to wyświetlenia strony przedmiotu. Sprawdziliśmy ponadto, że dla 64% użytkowników był to jedyny typ interakcji, który wykonali. Użytkownik wykonujący interakcję innego typu (z wyjątkiem dodania strony do ulubionych) musiał najpierw wyświetlić stronę przedmiotu. Możemy zatem podzielić interakcje wielokrotne na następujące grupy:

1. interakcje wielokrotne, w których typ każdej interakcji to click,
2. interakcje wielokrotne, w których typ każdej interakcji to bookmark,
3. interakcje wielokrotne, w których występujące interakcje miały co najmniej dwa różne typy,




**Rysunek 3.3.** Podział zbioru OLX Jobs Interactions na zbiór treningowy i testowy
Źródło: publikacja autora [78]

4. interakcje wielokrotne, w których typ każdej interakcji jest jednakowy, lecz inny niż click i bookmark – sytuacje takie mogą wystąpić w wyniku niedoskonałości systemu śledzenia zachowań użytkowników, bądź wskutek kroków podjętych w celu ochrony prywatności użytkowników opisanych w podrozdziale 3.1.

Sprawdziliśmy, że interakcje wielokrotne należące do grupy drugiej i czwartej stanowią łącznie około 0,1% wszystkich interakcji wielokrotnych, więc mają prawdopodobnie pomijalny wpływ na działanie systemu. Interakcje należące do grupy pierwszej i trzeciej stanowią odpowiednio 61% i 39% wszystkich interakcji wielokrotnych. Interakcje z pierwszej grupy mogą świadczyć o zainteresowaniu użytkownika, który co jakiś czas sprawdza czy oferta pracy jest jeszcze dostępna, lecz nie aplikuje na nią (np. ze względu na brak czasu na przygotowanie CV). W przypadku trzeciej grupy, użytkownik prawdopodobnie skontaktował się już z ogłoszeniodawcą. Zatem uwzględnienie przynależności interakcji do tych grup przez model rekomendacji może potencjalnie poprawić jego jakość.

## 3.4. Podział na zbiór treningowy i testowy

W sekcji opisujemy przyjęty przez nas podział na zbiór treningowy i testowy. Zdecydowaliśmy się umieścić 20% najnowszych interakcji w zbiorze testowym, zaś pozostałe interakcje w zbiorze treningowym. Oznacza to, że w zbiorze testowym umieściliśmy interakcje wykonane w ciągu około trzech ostatnich dni czternastodniowego okresu, z którego pochodzą dane, co przedstawiamy na rysunku 3.3. Autor zaproponował taki sposób podziału ze względu na praktyczne zastosowanie modeli wytrenowanych na tym zbiorze danych do generowania rekomendacji przesyłanych użytkownikom poprzez wiadomości email oraz push. Rekomendacje te wysyłane są raz dziennie i z tą samą częstotliwością trenowane są wykorzystywane do tego modele. Taki podział zbioru danych jest więc zbliżony do praktycznego wykorzystania w przedsiębiorstwie. Zauważmy, że w celu

trenowania modeli wykorzystywanych do generowania rekomendacji w trakcie sesji użytkownika (rekomendacji prezentowanych na stronie) lepszym podziałem mogłoby być przypisanie najświeższych 20% interakcji każdego użytkownika do zbioru testowego.

Ponadto, wprowadziliśmy następujące modyfikacje do zbioru testowego:

- ograniczyliśmy się jedynie do pierwszej interakcji między zadanym użytkownikiem a przedmiotem,
- usunęliśmy użytkowników, którzy nie wystąpili w zbiorze treningowym (ponieważ nie możemy dostarczyć dla nich spersonalizowanych rekomendacji),
- usunęliśmy interakcje, dla których interakcja pomiędzy użytkownikiem a przedmiotem wystąpiła w zbiorze treningowym (w celu uniknięcia rekomendowania przedmiotów, z którymi użytkownik wszedł już w interakcje).

W kolejnych rozdziałach wykorzystamy tak przygotowany zbiór treningowy do trenowania modeli rekomendacyjnych oraz przeprowadzimy ich ewaluację na zbiorze testowym.

## 3.5. Podsumowanie

W rozdziale zrealizowaliśmy pierwszy cel pomocniczy rozprawy: *przedstawienie opublikowanego przez autora zbioru danych*. Porównaliśmy istniejące zbiory danych wykorzystywane do ewaluacji modeli rekomendacji ofert pracy. Wskazaliśmy, że liczba interakcji w przypadku opublikowanego przez autora zbioru danych OLX Jobs Interactions jest o rząd wielkości większa niż w przypadku pozostałych porównywanych zbiorów. Ponadto, zbiór został umieszczony na popularnej platformie Kaggle, co czyni go łatwo dostępnym dla innych badaczy, w przeciwieństwie do zbiorów danych RecSys16 oraz RecSys17. Do dnia 10 listopada 2023 roku zbiór ten został wyświetlony 3516 razy oraz pobrany 107 razy, co świadczy o zainteresowaniu innych badaczy.

W dalszej części rozprawy przedstawimy wyniki ewaluacji istniejących oraz nowych metod rekomendacji czym jednocześnie ustanowimy punkt odniesienia dla innych badaczy testujących swoje metody na omawianym zbiorze danych.

Rozdział 4

# Porównanie istniejących metod

W rozdziale prezentujemy szczegółowe porównanie skuteczności znanych metod rekomendacji na opublikowanym zbiorze danych OLX Jobs Interactions.

Zarówno dobór ewaluowanych metod jak i sposób ich ewaluacji adresują szczególne cechy problemu rekomendacji ofert pracy w serwisie ogłoszeniowym. Cechy te wymieniamy poniżej.

- **Większość użytkowników jest anonimowa** - jest to jeden z powodów, dla których ograniczamy się do metod wspólnej filtracji.
- **Zawarcie transakcji najczęściej przebiega poza serwisem ogłoszeniowym** - z tego powodu rozpatrujemy metody rekomendacji stworzone dla zbiorów danych z niejawnymi ocenami.
- **Ogłoszeniodawca potrzebuje zwykle tylko jednego użytkownika do przeprowadzenia transakcji** - chcemy zatem uniknąć sytuacji, w której to samo ogłoszenie rekomendowane jest zbyt dużej liczbie użytkowników. W tym celu raportujemy wartości metryk pokrycia.
- **Czas emisji ogłoszenia jest ograniczony** - z tego powodu liczba dostępnych przedmiotów dynamicznie się zmienia. Modele wspólnej filtracji są w stanie generować spersonalizowane rekomendacje jedynie dla przedmiotów, które posiadały interakcje podczas treningu modelu. W celu uwzględnienia niedawno dodanych przedmiotów zdecydowaliśmy się na trenowanie modeli kilka razy dziennie. Konsekwencją tej decyzji jest konieczność uwzględnienia wydajności rozpatrywanych modeli podczas ewaluacji. W pracy raportujemy czas oraz wymaganą pamięć do treningu i predykcji każdej z rozpatrywanych metod.

Dokonując wyboru kierowaliśmy się także popularnością rozpatrywanych metod, zarówno w pracach naukowych, jak i praktycznym ich wykorzystaniu w przedsiębiorstwach obsługujących miliony użytkowników. Wybraliśmy metody reprezentujące różne podejścia, aby w kolejnych pracach ograniczyć się do rozwoju modeli z konkretnej grupy. Wybrane modele reprezentują cztery grupy metod rekomendacji: metody oparte na faktoryzacji macierzy, metody wykorzystujące algorytm najbliższych sąsiadów, metody

oparte na grafach oraz metody oparte na algorytmie Word2Vec. Ostatecznie wybraliśmy modele LightFM, ALS, SLIM, RP3Beta i Prod2Vec. Przedstawiamy szczegółowy opis matematyczny każdego z rozpatrywanych modeli.

Na podstawie wyników ewaluacji offline podjęliśmy decyzję o przeprowadzeniu dwóch testów A/B z użytkownikami serwisu `olx.pl`. Podczas testów użytkownicy otrzymywali powiadomienia email oraz push zawierające rekomendacje wygenerowane przez wybrane modele. Celem testów było zbadanie wpływu poszczególnych modeli rekomendacji na liczbę osób odpowiadających na oferty pracy. Przedstawiamy szczegółowe wyniki tych eksperymentów.

Wiele wyników prezentowanych w tym rozdziale pochodzi z publikacji autora [78].

## 4.1. Opis porównywanych metod

W kolejnych podrozdziałach prezentujemy szczegółowy opis matematyczny porównywanych modeli rekomendacji.

### 4.1.1. ALS

Techniki faktoryzacji macierzy są z powodzeniem wykorzystywane w systemach rekomendacji od czasu konkursu Netflix Prize [74]. Ich główną ideą jest przybliżenie macierzy interakcji użytkownik–przedmiot za pomocą iloczynu dwóch mniejszych, gęstych macierzy.

Model **ALS** (Alternating Least Squares, [64]), znany również jako WRMF (Weighted Regularized Matrix Factorization), jest jedną z metod faktoryzacji macierzy stworzoną dla zbiorów danych z niejawnymi ocenami. Wybraliśmy tę metodę ze względu na jej sprawdzoną wydajność, skalowalność i popularność w badaniach naukowych oraz praktycznych zastosowaniach. Model ALS został przez autora zaimplementowany w jednym z serwisów Pracodawcy jeszcze przed powstaniem opisywanego w rozdziale porównania z innymi metodami. Decyzja o wdrożeniu została podjęta na podstawie wyników eksperymentu, który szczegółowo opiszemy pod koniec tego rozdziału.

Niech $\mathcal{U}$ będzie zbiorem użytkowników, a $\mathcal{I}$ zbiorem przedmiotów. Dla każdego użytkownika $u \in \mathcal{U}$ oraz przedmiotu $i \in \mathcal{I}$ definiujemy ocenę $r_{ui}$. W naszych badaniach przyjmujemy $r_{ui} = 1$, jeśli użytkownik $u$ wszedł w interakcję z przedmiotem $i$, a $r_{ui} = 0$ w przeciwnym razie.

Zdefiniujmy **pewność** interakcji między użytkownikiem $u$ a przedmiotem $i$ jako $c_{ui} = 1 + \alpha r_{ui}$, gdzie $\alpha > 0$ jest hiperparametrem (autorzy tej metody uzyskali dobre wyniki dla $\alpha = 40$). Ponadto, zdefiniujmy **preferencję** użytkownika $u$ względem przedmiotu $i$ jako $p_{ui} = \mathbb{I}(r_{ui} > 0)$, gdzie $\mathbb{I}$ jest funkcją indykatorową. W modelu ALS minimalizujemy

wyrażenie:

$$\sum_{u,i} c_{ui}(p_{ui} - \boldsymbol{x}_u^\top \boldsymbol{y}_i)^2 + \lambda(\sum_u \|\boldsymbol{x}_u\|^2 + \sum_i \|\boldsymbol{y}_i\|^2), \tag{4.1}$$

gdzie $\boldsymbol{x}_u$ jest $d$-wymiarową reprezentacją użytkownika $u$, $\boldsymbol{y}_i$ jest $d$-wymiarową reprezentacją przedmiotu $i$, zaś $\lambda \geq 0$ jest parametrem regularyzacji. Celem treningu jest zatem znalezienie reprezentacji użytkowników i przedmiotów, dla których wyrażenie (4.1) przyjmuje możliwie najmniejszą wartość. Następnie, dla zadanego użytkownika $u$, rekomendowane są te przedmioty, dla których iloczyn skalarny $\boldsymbol{x}_u^\top \boldsymbol{y}_i$ osiąga największe wartości.

Nazwa modelu - ALS - odnosi się do metody naprzemiennych najmniejszych kwadratów (ang. *Alternating Least Squares*) wykorzystywanej do minimalizacji wyrażenia (4.1). Algorytm ten naprzemiennie:

- przyjmuje, że reprezentacje *użytkowników* są ustalone i w sposób analityczny wyznacza nowe reprezentacje *przedmiotów* minimalizując wyrażenie (4.1),
- przyjmuje, że reprezentacje *przedmiotów* są ustalone i w sposób analityczny wyznacza nowe reprezentacje *użytkowników* minimalizując wyrażenie (4.1).

Procedura ta jest powtarzana wielokrotnie w bardzo wydajny sposób [64, 113]. Zwróćmy uwagę na liczbę parametrów modelu ALS, która wynosi $d(|\mathcal{U}| + |\mathcal{I}|)$. Przyjmijmy liczbę wymiarów reprezentacji użytkowników i przedmiotów równą 200. Wtedy dla zbioru danych złożonego z miliona użytkowników i miliona przedmiotów, model posiada 400 milionów parametrów. Tak duża liczba parametrów pozwala modelowi uwzględnić złożone relacje w danych. Z drugiej zaś strony stwarza ona ryzyko nadmiernego dopasowania modelu do danych uczących.

Zauważmy również, że dwóch użytkowników z dokładnie takimi samymi interakcjami może otrzymać różne rekomendacje, ponieważ podczas trenowania modelu uzyskali oni inne reprezentacje. Poza modelami faktoryzacji macierzy (ALS i LightFM), pozostałe rozważane modele (SLIM, RP3Beta, Prod2Vec) generują taką samą listę rekomendacji dla użytkowników z takimi samymi interakcjami. Generowanie różnych rekomendacji dla użytkowników o tym samym zbiorze interakcji, może być korzystne ze względu na pozyskiwanie w ten sposób bardziej różnorodnych danych pozwalających na wytrenowanie w przyszłości lepszych modeli. Z drugiej strony, nie jest jasne jakie rekomendacje byłyby najbardziej odpowiednie dla nowego użytkownika (nieobecnego podczas trenowania modelu), posiadającego zadany zbiór interakcji, nawet jeśli istnieli w zbiorze treningowym użytkownicy posiadający dokładnie ten sam zbiór interakcji. Wpływa to na skuteczność proponowanego przez nas rozwiązania umożliwiającego generowanie rekomendacji w czasie rzeczywistym, co omawiamy w podrozdziale 5.3.2.

### 4.1.2. LightFM

Model **LightFM** (Light Factorization Machine [75]) to model hybrydowy zaproponowany w celu przezwyciężenia problemu zimnego startu metod faktoryzacji macierzy. Jest on uproszczoną wersją modelu Factorization Machine [103]. W naszym przypadku, bez dodatkowych informacji o użytkownikach i przedmiotach, model LightFM sprowadza się do klasycznej faktoryzacji macierzy. Mimo to zdecydowaliśmy się na zastosowanie tego podejścia ze względu na jego wydajną implementację udostępnioną przez autora tej metody, która obsługuje wiele funkcji straty: logistyczną, BPR [104], WARP [129] i $k$-OS WARP [130].

W przypadku braku dodatkowych cech, model LightFM przewiduje ocenę $r_{ui}$ jako:

$$\hat{r}_{ui} = \boldsymbol{x}_u \cdot \boldsymbol{y}_i + b_u + b_i,$$

gdzie $\boldsymbol{x}_u$ i $\boldsymbol{y}_i$ oznaczają $d$-wymiarowe reprezentacje użytkownika $u$ i przedmiotu $i$, $b_u$ to wyraz wolny związany z użytkownikiem $u$, zaś $b_i$ to wyraz wolny związany z przedmiotem $i$. Wszystkie te parametry są znajdowane podczas trenowania modelu.

Logistyczna funkcja straty powinna być używana, gdy dostępne są zarówno pozytywne, jak i negatywne interakcje, więc nie jest odpowiednia dla naszego zbioru danych, co potwierdziliśmy eksperymentalnie podczas optymalizacji hiperparametrów.

Funkcje straty BPR, WARP i $k$-OS WARP to podejścia uczenia się na parach (przedstawionego w podrozdziale 2.1.2). Ich celem jest maksymalizacja różnicy między predykcjami ocen przedmiotów $i \in \mathcal{N}(u)$, z którymi użytkownik $u$ wszedł w interakcję, a predykcjami ocen przedmiotów $i \notin \mathcal{N}(u)$, z którymi użytkownik ten nie wszedł w interakcję.

W szczególności, w funkcji straty **BPR** [104] staramy się zmaksymalizować wyrażenie:

$$\sum_{(u,i,j)\in\mathcal{S}} \ln \sigma(\hat{r}_{ui} - \hat{r}_{uj}) - \lambda_{\boldsymbol{\Theta}} \|\boldsymbol{\Theta}\|^2,$$

gdzie

$$\mathcal{S} = \{(u,i,j) | u \in \mathcal{U}, i \in \mathcal{N}(u) \text{ oraz } j \notin \mathcal{N}(u)\},$$

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

$\lambda_{\boldsymbol{\Theta}} \geq 0$ jest parametrem regularyzacji specyficznym dla modelu, $\boldsymbol{\Theta}$ jest wektorem parametrów, a $\|\cdot\|$ jest normą euklidesową.

Przedstawimy funkcję straty **WARP** [129] jako szczególny przypadek funkcji ***k*-OS WARP** [130]. Funkcja straty $k$-OS WARP jest sumą wartości strat wyliczonych dla poszczególnych użytkowników:

$$\sum_{u\in\mathcal{U}} L(\hat{\boldsymbol{r}}_u, \mathcal{N}(u)),$$

gdzie $\hat{\boldsymbol{r}}_u$ jest wektorem estymowanych ocen $\hat{r}_{ui}$ dla wszystkich przedmiotów $i \in \mathcal{I}$. Dla zadanego użytkownika $u$ sortujemy przedmioty ze zbioru $\mathcal{N}(u)$, z którymi wszedł on w interakcje, w porządku malejącym względem estymowanych ocen. Niech $i_1, i_2, \ldots, i_{|\mathcal{N}(u)|}$ będzie uzyskaną listą, dla której:

$$\hat{r}_{ui_1} \geq \hat{r}_{ui_2} \geq \ldots \geq \hat{r}_{ui_{|\mathcal{N}(u)|}}.$$

Wtedy:

$$L(\hat{\boldsymbol{r}}_u, \mathcal{N}(u)) = \frac{1}{Z} \sum_{s=1}^{|\mathcal{N}(u)|} P(s)\Phi(\text{rank}_{i_s}(\hat{\boldsymbol{r}}_u)),$$

gdzie $\text{rank}_i(\hat{\boldsymbol{r}}_u) = \sum_{j \in \mathcal{I} \setminus \mathcal{N}(u)} \mathbb{I}(1 + \hat{r}_{uj} \geq \hat{r}_{ui})$, $\mathbb{I}$ oznacza funkcję indykatorową, $\Phi(n) = \sum_{m=1}^{n} \frac{1}{m}$, $P$ jest ustaloną funkcją przypisującą wagę każdej pozycji na rozpatrywanej liście przedmiotów, zaś $Z = \sum_{s=1}^{|\mathcal{N}(u)|} P(s)$ normalizuje wagi wygenerowane przez funkcję $P$.

W praktyce losujemy pozytywny przedmiot $i$ z uwzględnieniem funkcji ważenia $P$ i losowo wybieramy przedmioty $j \notin \mathcal{N}(u)$ do momentu, gdy po pewnej liczbie $N$ losowań uzyskamy $1 + \hat{r}_{uj} \geq \hat{r}_{ui}$. Następnie obliczamy gradient funkcji $\Phi(\lfloor \frac{|\mathcal{I} \setminus \mathcal{N}(u)|}{N} \rfloor)(1 - \hat{r}_{ui} + \hat{r}_{uj})$ i aktualizujemy wagi naszego modelu zgodnie z założeniami metody gradientu prostego.

W przypadku funkcji straty $k$-OS WARP przyjmujemy $P(k) = 1$ i $P(m) = 0$, jeśli $m \neq k$ (por. [130]). Funkcja straty WARP jest osiągana, gdy $P(m) = 1$ dla $m \in \mathbb{N}$. W tym przypadku, aby uprościć obliczenia, możemy pominąć sortowanie przedmiotów ze zbioru $\mathcal{N}(u)$.

W naszych badaniach wszystkie funkcje straty (logistyczna, BPR, WARP i $k$-OS WARP) były traktowane jako hiperparametry modelu LightFM i optymalizowane podczas treningu.

### 4.1.3. SLIM

Model **SLIM** (Sparse Linear Methods for Top-N Recommender Systems; [95]) bazuje na regresji najbliższych sąsiadów opartej na przedmiotach [7]. W przypadku klasycznej metody $k$-najbliższych sąsiadów opartej na przedmiotach [37] liczba $k$ sąsiadów jest hiperparametrem modelu, zaś podobieństwo między przedmiotami wyznaczane jest za pomocą zadanej funkcji podobieństwa, najczęściej odległości cosinusowej. W odróżnieniu od tego podejścia, w modelu SLIM nie jest wybierana liczba najbliższych sąsiadów, a podobieństwo pomiędzy przedmiotami wyliczane jest na podstawie parametrów optymalizowanych podczas treningu. SLIM osiąga często lepsze wyniki od metody $k$-najbliższych sąsiadów opartych na przedmiotach [31, 95].

W modelu SLIM przybliżamy macierz interakcji $\boldsymbol{R}$ wykorzystując rzadką macierz podobieństwa przedmiot-przedmiot $\boldsymbol{W}$. Macierz $\boldsymbol{W}$ jest uczona poprzez minimalizację wyrażenia:

$$\frac{1}{2}\|\boldsymbol{R} - \boldsymbol{R}\boldsymbol{W}\|_F^2 + \frac{\beta}{2}\|\boldsymbol{W}\|_F^2 + \lambda\|\boldsymbol{W}\|_1,$$

gdzie $w_{ij} \geq 0$ i $w_{ii} = 0$ dla wszystkich $i, j, 1 \leq i, j \leq |\mathcal{I}|$,
$\|\cdot\|_F$ jest normą Frobeniusa macierzy:

$$\|\boldsymbol{W}\|_F = \sqrt{\sum_{i=1}^{|\mathcal{I}|} \sum_{j=1}^{|\mathcal{I}|} w_{ij}^2},$$

$\|\cdot\|_1$ jest normą $\ell_1$ macierzy:

$$|\boldsymbol{W}|_1 = \sum_{i=1}^{|\mathcal{I}|} \sum_{j=1}^{|\mathcal{I}|} |w_{ij}|,$$

a $\beta \geq 0$ i $\lambda \geq 0$ są parametrami regularyzacji. Zauważmy, że każda kolumna $\boldsymbol{W}$ może być uczona niezależnie, co poprawia skalowalność tego modelu [95].

W przypadku metod faktoryzacji macierzy, macierz interakcji $\boldsymbol{R}$ jest przybliżana przez iloczyn dwóch gęstych macierzy, w których wiersze lub kolumny oznaczają $d$-wymiarowe reprezentacje użytkowników. Wymiar $d$ wynosi zwykle od kilkudziesięciu do kilkuset [31, 75]. Jak zauważyli twórcy modelu SLIM [95], niska wymiarowość szukanych macierzy w przypadku modeli faktoryzacji macierzy może prowadzić do utraty pewnej części informacji podczas treningu. Zauważmy, że w przypadku modelu SLIM, macierz $\boldsymbol{R}$ przybliżana jest przez iloczyn macierzy $\boldsymbol{R}$ oraz $|\mathcal{I}| \times |\mathcal{I}|$-wymiarowej macierzy $\boldsymbol{W}$. Przedmioty są zatem reprezentowane poprzez $|\mathcal{I}|$-wymiarowe rzadkie wektory, gdzie liczba przedmiotów $|\mathcal{I}|$ jest zwykle o kilka rzędów wielkości większa od liczby wymiarów $d$. Zatem model SLIM jest potencjalnie w stanie przechowywać bogatsze reprezentacje przedmiotów i zapobiec wspomnianej utracie informacji.

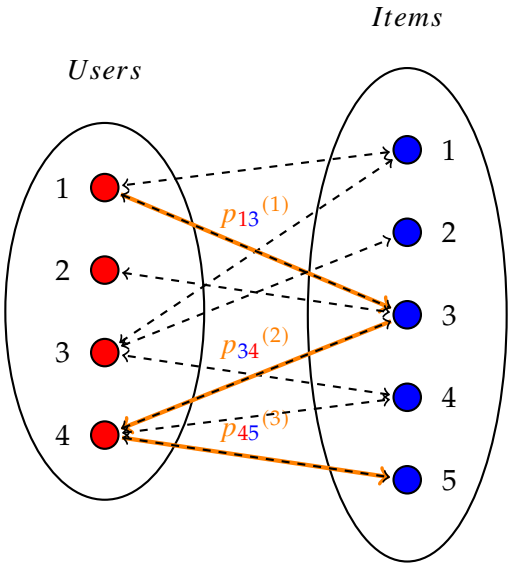
Wybraliśmy metodę SLIM ze względu na wymienione zalety w stosunku do klasycznej metody $k$-najbliższych sąsiadów opartych na przedmiotach oraz modeli faktoryzacji macierzy [95], a także osiągane wyniki w porównaniu z innymi modelami [31].

Pewnym ograniczeniem tej metody jest fakt, że nie generuje ona gęstych reprezentacji wektorowych użytkowników i przedmiotów, które mogłyby mieć także inne zastosowania (np. personalizacja wyników wyszukiwarki). Wprawdzie możemy traktować kolumny macierzy $\boldsymbol{W}$ jako reprezentacje przedmiotów, ale są to rzadkie wektory $|\mathcal{I}|$-wymiarowe, więc w wielu przypadkach nie mogą być bezpośrednio wykorzystane w innych modelach.

### 4.1.4. RP3Beta

**RP3Beta** [27] to podejście wspólnej filtracji oparte na grafie reprezentującym interakcje między użytkownikami a przedmiotami. Podobnie jak Dacrema i in. [31], traktujemy ten model jako uogólnienie modelu P3Alpha [28] i obliczamy dokładne wyniki zamiast przybliżeń za pomocą spacerów losowych.

Oznaczmy zbiór użytkowników przez $\mathcal{U}$, a zbiór przedmiotów przez $\mathcal{I}$. Reprezentujemy zbiór danych za pomocą grafu dwudzielnego, w którym wierzchołkami są




**Rysunek 4.1.** Ścieżka długości 3 z zaznaczonymi wartościami przypisanymi do krawędzi tworzących tę ścieżkę. Przerywane linie reprezentują interakcje pomiędzy użytkownikami a przedmiotami
Źródło: publikacja autora [78]

użytkownicy i przedmioty, a krawędzie reprezentują interakcje między nimi. Niech $\mathcal{N}(x)$ będzie zbiorem sąsiadów wierzchołka $x$. Model RP3Beta przewiduje ocenę $r_{ui}$ przedmiotu $i \in \mathcal{I}$ dla użytkownika $u \in \mathcal{U}$, jako sumę wartości przypisanych do ścieżek o długości 3 łączących wierzchołki reprezentujące użytkownika $u$ oraz przedmiot $i$:

$$\hat{r}_{ui} = \sum_{i' \in \mathcal{N}(u)} \sum_{u' \in \mathcal{N}(i')} p(u, i', u', i).$$

Wartość przypisana do danej ścieżki jest iloczynem wartości przypisanych do krawędzi tworzących daną ścieżkę:

$$p(u, i', u', i) = p_{ui'}^{(1)} p_{i'u'}^{(2)} p_{u'i}^{(3)},$$

gdzie $p_{ui'}^{(1)} = \frac{1}{|\mathcal{N}(u)|^{\alpha}}$, $p_{i'u'}^{(2)} = \frac{1}{|\mathcal{N}(i')|^{\alpha}}$ oraz $p_{u'i}^{(3)} = \frac{1}{|\mathcal{N}(u')|^{\alpha}|\mathcal{N}(i)|^{\beta}}$. Wartości krawędzi dla przykładowej ścieżki pokazane są na rysunku 4.1.

Zauważmy, że możemy wydajnie obliczyć predykcje preferencji wszystkich użytkowników poprzez mnożenie trzech rzadkich macierzy:

$$\hat{\boldsymbol{R}} = \boldsymbol{P}^{(1)} \boldsymbol{P}^{(2)} \boldsymbol{P}^{(3)}, \tag{4.2}$$

gdzie $\hat{\boldsymbol{R}} = (\hat{r}_{ui})$ oraz $\boldsymbol{P}^{(k)} = (p_{xy}^{(k)})$. Zauważmy, że macierze $\boldsymbol{P}^{(1)}$, $\boldsymbol{P}^{(3)}$ są wymiaru $|\mathcal{U}| \times |\mathcal{I}|$, zaś macierz $\boldsymbol{P}^{(2)}$ jest wymiaru $|\mathcal{I}| \times |\mathcal{U}|$.

Zdecydowaliśmy się wybrać ten model ze względu na jego prostotę, skalowalność i sprawdzoną dokładność [12,31,32]. Dacrema i in. [31,32] pokazali, że model RP3Beta,

w przypadku wielu zbiorów danych, daje lepsze wyniki od innych modeli nieneuronowych, a także niedawno proponowanych metod neuronowych. Model RP3Beta osiągnął najlepsze wyniki wśród porównywanych metod na zbiorze danych Pinterest [31, 50] oraz CiteULike-a [32, 121]. Ponadto, rzadkość naszego zbioru danych umożliwia bezpośrednie wyznaczenie i przechowywanie w pamięci macierzy podobieństwa pomiędzy przedmiotami, co znacznie ułatwiło wydajną implementację.

Model RP3Beta, podobnie do modelu SLIM, nie generuje reprezentacji użytkowników i przedmiotów. Innym ograniczeniem jest brak możliwości rekomendowania przedmiotów, których odległość na przedstawionym grafie dwudzielnym jest większa niż 3 od danego użytkownika. Mówiąc wprost, przedmiot $i$ może być rekomendowany użytkownikowi $u$ tylko wtedy, gdy istnieje co najmniej jeden użytkownik $u'$, który wszedł w interakcję z przedmiotem $i$ oraz z co najmniej jednym przedmiotem, z którym wszedł w interakcję użytkownik $u$. Dodatkowo, model RP3Beta nie ma parametrów optymalizowanych w procesie trenowania modelu, dlatego może nie być w stanie wykorzystać wszystkich informacji zawartych w danych.

### 4.1.5. Prod2Vec

Model **Prod2Vec** [15, 53] jest oparty na modelu Word2Vec [92] szeroko stosowanym w przetwarzaniu języka naturalnego. W przypadku modeli rekomendacyjnych, przedmioty pełnią rolę słów. Ciągi przedmiotów, z którymi użytkownik wszedł w interakcję rozumiane są jako zdania. Dzięki tej analogii, model Word2Vec może być niemalże bezpośrednio zastosowany jako model rekomendacyjny. W procesie uczenia, model znajduje reprezentacje wektorowe przedmiotów. Metoda ta została wybrana, ponieważ jest skalowalna i bardzo różni się od porównywanych metod. Ponadto, na modelu Word2Vec bazuje bardziej zaawansowany model Item2Vec wdrożony w serwisach Pracodawcy (por. 1.3.2), co jest przykładem skutecznego praktycznego zastosowania tego typu modeli.

Niech $\mathcal{S}$ oznacza zbiór ciągów przedmiotów (na przykład kolejnych interakcji użytkowników). Model Prod2Vec maksymalizuje podczas treningu następującą funkcję straty:

$$L = \sum_{s \in \mathcal{S}} \sum_{i_k \in s} \sum_{\substack{-c \leq l \leq c \\ l \neq 0}} \log \mathbb{P}(i_{k+l} | i_k),$$

gdzie $c$ jest ustaloną liczbą całkowitą dodatnią. Prawdopodobieństwo $\mathbb{P}(i_{k+l}|i_k)$ zaobserwowania przedmiotu $i_{k+l}$ pod warunkiem zaobserwowania przedmiotu $i_k$ zdefiniowane jest za pomocą funkcji softmax:

$$\mathbb{P}(i_{k+l} | i_k) = \frac{\exp(\boldsymbol{y}_{i_k}^\top \boldsymbol{y}'_{i_{k+l}})}{\sum_{i=1}^{|\mathcal{I}|} \exp(\boldsymbol{y}_{i_k}^\top \boldsymbol{y}'_i)},$$

gdzie $\boldsymbol{y}_i$ oraz $\boldsymbol{y}_i^{'}$ nazywamy odpowiednio wejściową i wyjściową reprezentacją wektorową przedmiotu $i$, zaś $|\mathcal{I}|$ jest łączną liczbą przedmiotów. W modelu Prod2Vec przedmioty, które często występują blisko siebie w ciągach należących do zbioru $\mathcal{S}$ uzyskają prawdopodobnie podobne reprezentacje wektorowe.

Przed rozpoczęciem generowania rekomendacji dokonujemy normalizacji reprezentacji przedmiotów (tj. każdy wektor reprezentujący przedmiot dzielimy przez jego długość). Operacja ta ma niewielki wpływ na jakość rekomendacji, a umożliwia szybsze ich generowanie. W celu wygenerowania rekomendacji dla zadanego użytkownika, reprezentujemy go jako średnią wejściowych reprezentacji wektorowych przedmiotów, z którymi wszedł w interakcje. Następnie rekomendujemy te przedmioty, których iloczyn skalarny wejściowej reprezentacji przedmiotu oraz reprezentacji użytkownika są największe (realizujemy zatem algorytm $k$-najbliższych sąsiadów).

## 4.2. Implementacja

Dokonaliśmy implementacji opisanych metod z wykorzystaniem istniejących bibliotek lub implementacji. Szczegółowe informacje na ten temat, wraz ze wskazaniem różnic względem publikacji naukowych przedstawiających dane metody, znajdują się w tabeli 4.1. Kod źródłowy pozwalający na odtworzenie prezentowanych wyników opublikowany został w repozytorium Github autora[1].

Modele zostały wytrenowane i zewaluowane na przedstawionym w rozdziale 3 zbiorze danych OLX Jobs Interactions. Podział interakcji na zbiór treningowy i testowy opisaliśmy w podrozdziale 3.4. Rozważane w rozdziale modele nie uwzględniają typu ani krotności interakcji. Z tego powodu zmodyfikowaliśmy zbiór treningowy poprzez ograniczenie się do pierwszej interakcji pomiędzy użytkownikiem a przedmiotem oraz pominięcie informacji o typie interakcji. Finalnie zbiór treningowy liczył około 38 milionów interakcji, zaś zbiór testowy około 6 milionów.

## 4.3. Optymalizacja hiperparametrów

W podrozdziale prezentujemy procedurę optymalizacji wykorzystaną do znalezienia hiperparametrów oraz uzyskane za jej pomocą wyniki. Jak podaje Dacrema i in. [31] przeszukiwanie przestrzeni hiperparametrów dla wszystkich porównywanych modeli jest kluczowe dla wiarygodnego porównania jakości modeli. Częstym błędem jest wykorzystywanie raportowanych przez innych badaczy hiperparametrów modeli optymalizowanych dla innych zbiorów danych bądź metod ewaluacji [31].

Podczas szukania optymalnych hiperparametrów kierujemy się maksymalizacją me-

[1] `https://github.com/rob-kwiec/olx-jobs-recommendations`, dostęp: 2023-12-02

**Tabela 4.1.** Szczegóły implementacji ewaluowanych metod

| Model | Źródła | Różnice względem publikacji wprowadzającej model |
|---|---|---|
| LightFM [75] | Implementacja oparta na bibliotece [76]. | Brak. |
| ALS [64, 113] | Implementacja oparta na bibliotece [43]. | Brak. |
| SLIM [95] | Implementacja autora inspirowana [30, 54]. | Brak. |
| RP3Beta [98] | Implementacja autora inspirowana [29]. | Wykonujemy mnożenie rzadkich macierzy zamiast aproksymacji za pomocą spacerów losowych, podobnie do [31]. |
| Prod2Vec [15, 53] | Implementacja wykorzystująca zaimplementowany algorytm Word2Vec [139]. | Ciągi interakcji posortowane względem czasu interakcji (zamiast losowo). Reprezentacja użytkownika uzyskana jako średnia reprezentacji wejściowych przedmiotów, z którymi użytkownik wszedł w interakcję. Zaimplementowana możliwość użycia algorytmu CBOW (Continuous Bag of Words) zamiast skip-gram. |

Źródło: publikacja autora [78]

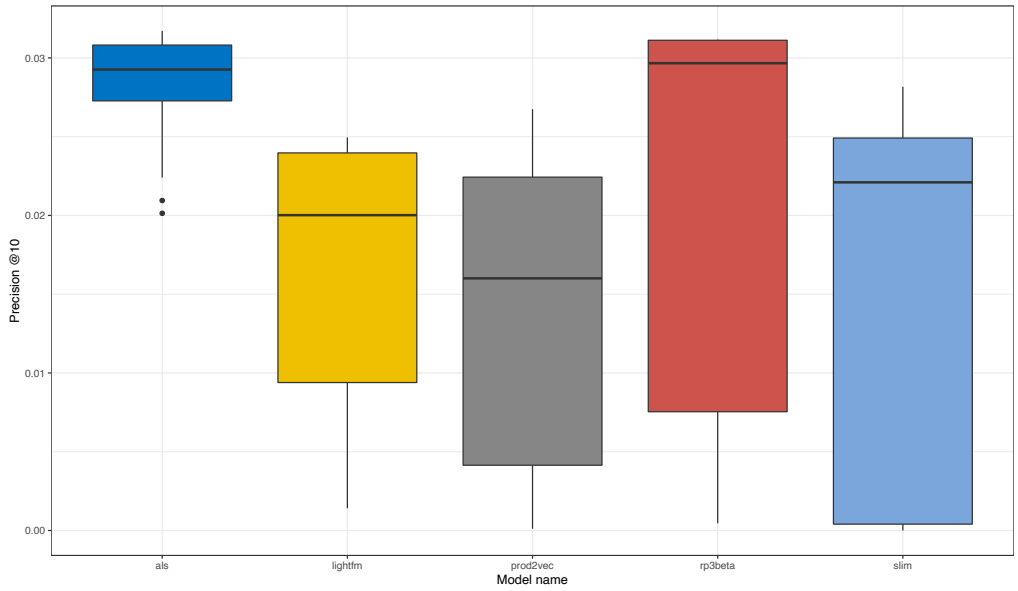
tryki Precision@$k$, która jak wspomnieliśmy wcześniej, jest także główną metryką ewaluacji modeli. Podczas ewaluacji online przesyłamy użytkownikom rekomendacje dziesięciu przedmiotów za pomocą powiadomień email. Z tego powodu przyjęliśmy $k = 10$.

Ze względu na złożoność obliczeniową, podczas optymalizacji hiperparametrów dokonaliśmy ograniczenia do losowo wybranych 20% użytkowników i 20% przedmiotów zbioru treningowego. Następnie podzieliliśmy tak ograniczony zbiór danych na zbiór treningowy i walidacyjny w ten sam sposób, w jaki dokonaliśmy podziału na zbiór treningowy i testowy. Dla każdego modelu testowaliśmy 100 zestawów hiperparametrów. Do ich wyboru zastosowaliśmy optymalizację bayesowską przy użyciu procesów gaussowskich. W tym celu wykorzystaliśmy bibliotekę ScikitLearn[2]. Plik konfiguracyjny definiujący możliwe wartości hiperparametrów jest dostępny w repozytorium Github autora[3]. Przykładowo, dla modelu RP3Beta ograniczyliśmy się do wyszukiwania hiperparametrów $\alpha$ i $\beta$ w przedziale [0,2], podobnie do [31].

Rekomendacje w procesie optymalizacji zostały przygotowane dla 30 000 użytkowników. Rysunek 4.2 pokazuje wartości metryki Precision@10 w zależności od wartości hiperparametrów. Widać, że poza modelem ALS, jakość modelu jest mocno zależna

[2] `https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html`, dostęp: 2023-11-18

[3] `https://github.com/rob-kwiec/olx-jobs-recommendations/blob/main/src/tuning/config.py`, dostęp: 2023-12-02




**Rysunek 4.2.** Wykres pudełkowy wartości metryki Precision@10 dla każdego modelu w zależności od wartości hiperparametrów
Źródło: publikacja autora [78]

**Tabela 4.2.** Wartości hiperparametrów, dla których wartość metryki Precision@10 jest najwyższa wśród testowanych możliwości

| Model | Hiperparametry modelu |
|---|---|
| **ALS** | {'factors': 357, 'regularization': 0,001, 'iterations': 20, 'event_weights_multiplier': 63} |
| **LightFM** | {'no_components': 512, 'k': 3, 'n': 20, 'learning_schedule': 'adadelta', 'loss': 'warp', 'max_sampled': 61, 'epochs': 11} |
| **Prod2Vec** | {'vector_size': 168, 'alpha': 0,028728, 'window': 20, 'min_count': 16, 'sample': 0,002690026, 'min_alpha': 0,0, 'sg': 1, 'hs': 1, 'negative': 200, 'ns_exponent': -0,16447846705441527, 'cbow_mean': 0, 'epochs': 22} |
| **RP3Beta** | {'alpha': 0,61447198, 'beta': 0,1443548} |
| **SLIM** | {'alpha': 0,00181289, 'l1_ratio': 0,0, 'iterations': 3} |

Źródło: publikacja autora [78]

od dobranych hiperparametrów. Wybrane w procesie optymalizacji hiperparametry przedstawiamy w tabeli 4.2.

## 4.4. Ewaluacja offline

Porównywane metody zostały wytrenowane przy użyciu hiperparametrów przedstawionych w tabeli 4.2. Podczas treningu wykorzystaliśmy pełny zbiór treningowy, a rekomendacje zostały wygenerowane dla wszystkich 619 389 użytkowników ze zbioru testowego. Dodatkowo, dla lepszego ukazania wartości przyjmowanych przez poszczególne metryki, przedstawiamy również wyniki ewaluacji następujących modeli:

- modelu **TopPop** rekomendującego przedmioty, z którymi weszło w interakcje najwięcej użytkowników,
- modelu **Random** rekomendującego losowe przedmioty ze zbioru treningowego (losowane niezależnie dla każdego użytkownika, z równym prawdopodobieństwem wylosowania każdego z przedmiotów).

Modele te będziemy nazywać niespersonalizowanymi, ponieważ generując rekomendacje dla zadanego użytkownika nie wykorzystują one żadnych jego cech ani informacji o jego interakcjach. Ponadto, we wszystkich przedstawionych podejściach rekomendowane są jedynie przedmioty, z którymi użytkownik nie wszedł jeszcze w interakcję.

### 4.4.1. Metryki dokładności

#### Wartości metryk dokładności

Wszystkie metryki przedstawione w tej sekcji uwzględniają pierwsze 10 rekomendacji, np. Precision@10, Recall@10. Wyższe wartości metryk wskazują na wyższą jakość rekomendacji. Wartości nie powinny być bezpośrednio porównywane z wynikami osiągniętymi na innych zbiorach danych, ponieważ metryki w dużym stopniu zależą od charakterystyki zbioru (na przykład rzadkości macierzy interakcji) i strategii podziału pomiędzy zbiór treningowy i testowy. Tabela 4.3 przedstawia wyniki przeprowadzonej ewaluacji. Ze względu na dużą liczbę ocenianych użytkowników, różnice dokładności prezentowanych modeli są statystycznie istotne dla większości par modeli. Szczegółową analizę istotności statystycznej różnic przedstawiamy w dalszej części podrozdziału.

Widzimy, że podejścia spersonalizowane znacznie przewyższają podejścia niespersonalizowane. Niska jakość modelu TopPop rekomendującego najpopularniejsze przedmioty związana jest prawdopodobnie ze specyfiką rekomendacji ofert pracy, gdzie użytkownicy często zainteresowani są ofertami jedynie z konkretnej lokalizacji i kategorii. Jakość modelu TopPop w stosunku do innych metod może być postrzegana jako jedna z cech zbioru danych. Przykładowo dla zbioru danych Epinions, metoda ta daje lepsze wyniki od wielu spersonalizowanych modeli [32]. W przypadku rekomendacji ofert pracy wdrożenie spersonalizowanych systemów rekomendacji wydaje się szczególnie istotne.

Pokazaliśmy, że model RP3Beta przewyższa inne podejścia pod względem wszystkich raportowanych miar dokładności. Uważamy, że jest to związane z dużą rzadkością

**Tabela 4.3.** Wartości metryk dokładności dla porównywanych metod

| Metryka | RP3Beta | SLIM | ALS | Prod2Vec | LightFM | TopPop | Random |
|---|---|---|---|---|---|---|---|
| Precision | **0,0484** | 0,0472 | 0,0434 | 0,0368 | 0,0359 | 0,0012 | 0,00006 |
| Recall | **0,0783** | 0,0736 | 0,0657 | 0,0580 | 0,0564 | 0,0012 | 0,00005 |
| NDCG | **0,0759** | 0,0721 | 0,0657 | 0,0567 | 0,0545 | 0,0016 | 0,00007 |
| MAP | **0,0393** | 0,0365 | 0,0329 | 0,0282 | 0,0264 | 0,0006 | 0,00002 |
| MRR | **0,1365** | 0,1314 | 0,1230 | 0,1065 | 0,1034 | 0,0038 | 0,00019 |
| LAUC | **0,5391** | 0,5368 | 0,5328 | 0,5289 | 0,5281 | 0,5006 | 0,49999 |
| HR | **0,3131** | 0,3066 | 0,2878 | 0,2537 | 0,2547 | 0,0112 | 0,00059 |

Źródło: publikacja autora [78]

naszego zbioru danych. RP3Beta oblicza rekomendacje w sposób deterministyczny (wykorzystując ścieżki o długości 3 na dwudzielnym grafie użytkownik-przedmiot). Wszystkie pozostałe podejścia wykorzystują techniki uczenia maszynowego do znajdowania reprezentacji użytkowników (ALS, LightFM), reprezentacji przedmiotów (ALS, LightFM, Prod2Vec) lub bezpośrednich podobieństw między przedmiotami (SLIM), co może stanowić wyzwanie w przypadku użytkowników lub przedmiotów z niewielką liczbą interakcji.

**Istotność statystyczna różnic**

Następnie zbadaliśmy występowanie statystycznie istotnych różnic między analizowanymi metodami dla naszej głównej metryki, Precision@10. Na początek testujemy hipotezę zerową, że wszystkie porównywane metody nie różnią się między sobą, a obserwowane różnice są jedynie losowe (test typu omnibus). Test Friedmana [44,45] z rozszerzeniem zaproponowanym przez Imana i Davenporta [67] jest prawdopodobnie najpopularniejszym testem typu omnibus i zwykle jest dobrym wyborem przy porównywaniu więcej niż pięciu różnych algorytmów [47,48]. Niech $R_{ij}$ będzie rangą $j$-tej z $K$ metod na $i$-tej z $N$ obserwacji. Ponadto, zdefiniujmy średnią rangę metody $j$ wzorem

$$R_j = \frac{1}{N} \sum_{i=1}^{N} R_{ij}.$$

Test porównuje średnie rangi metod i opiera się na statystyce

$$F_F = \frac{(N-1)\chi_F^2}{N(K-1) - \chi_F^2},$$

gdzie

$$\chi_F^2 = \frac{12N}{K(K+1)} \sum_{i=1}^{K} R_i^2 - 3N(K+1)$$

jest statystyką Friedmana, która ma rozkład $F$ Snedecora z $K-1$ i $(K-1)(N-1)$ stopniami swobody. W naszym przypadku $p$-wartość z tego testu jest równa 0, zatem możemy bezpiecznie odrzucić hipotezę zerową, głoszącą że porównywane metody nie różnią się między sobą. Możemy zatem przystąpić do testów post-hoc w celu wykrycia znaczących różnic pomiędzy parami rozpatrywanych metod. Demšar [35] proponuje użycie testu Nemenyi, który porównuje wszystkie algorytmy parami. Dla zadanego poziomu istotności $\alpha$ definiujemy *różnicę krytyczną* (ang. *critical difference*, CD) wzorem

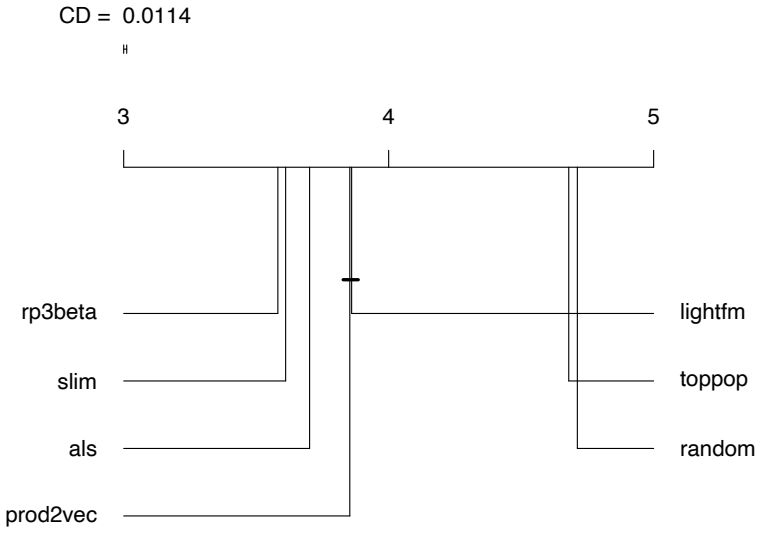
$$\mathrm{CD} = q_\alpha \sqrt{\frac{K(K+1)}{6N}},$$

gdzie wartość $q_\alpha$ jest kwantylem rzędu $\alpha$ rozkładu studentyzowanego rozstępu podzielonym przez $\sqrt{2}$. Jeśli różnica między średnimi rangami dwóch algorytmów jest większa niż różnica krytyczna, hipoteza zerowa, że algorytmy mają taką samą skuteczność, zostaje odrzucona. Demšar [35] zaproponował wykres ilustrujący różnice między poszczególnymi parami algorytmów. Dwa algorytmy można uznać za różne, jeśli nie są na tym wykresie połączone linią.

W naszym przypadku, przy poziomie istotności $\alpha = 0{,}05$, dowolne dwa algorytmy, których różnica między średnimi rangami jest większa niż 0,0114 będą traktowane jako różne. Na rysunku 4.3 możemy zauważyć, że hipoteza zerowa została odrzucona dla dowolnej pary algorytmów, poza parą (Prod2Vec, LightFM), dla której nie ma podstaw do odrzucenia hipotezy zerowej na zadanym poziomie istotności. W szczególności, możemy przyjąć, że model RP3Beta jest istotnie lepszy od pozostałych modeli.

**Dokładność w zależności od liczby interakcji użytkownika**

Jedyne informacje, jakie mamy o naszych użytkownikach, to ich interakcje. Podzieliliśmy użytkowników na 10 grup o podobnej liczności w zależności od liczby przedmiotów, z którymi weszli oni w interakcje w zbiorze treningowym. W każdej z tych grup policzyliśmy wartość metryki Precision@10 dla każdego z rozważanych algorytmów. Na rysunku 4.4 przedstawione są wyniki tej ewaluacji dla spersonalizowanych algorytmów (pominęliśmy niespersonalizowane modele TopPop i Random w celu zwiększenia czytelności wykresu). Zauważmy, że kolejność modeli posortowanych według metryki Precision@10 w większości przypadków nie zależy od liczby interakcji użytkowników. Wyjątkiem jest model SLIM, który wydaje się przewyższać model RP3Beta dla użytkowników z co najmniej 22 interakcjami. Różnica jest statystycznie istotna dla każdej grupy użytkowników (szczegóły analizy statystycznej przedstawione są w sekcji 4.4.1). Model RP3Beta może być postrzegany jako model najbliższych sąsiadów oparty na przedmiotach, w którym przyjąć



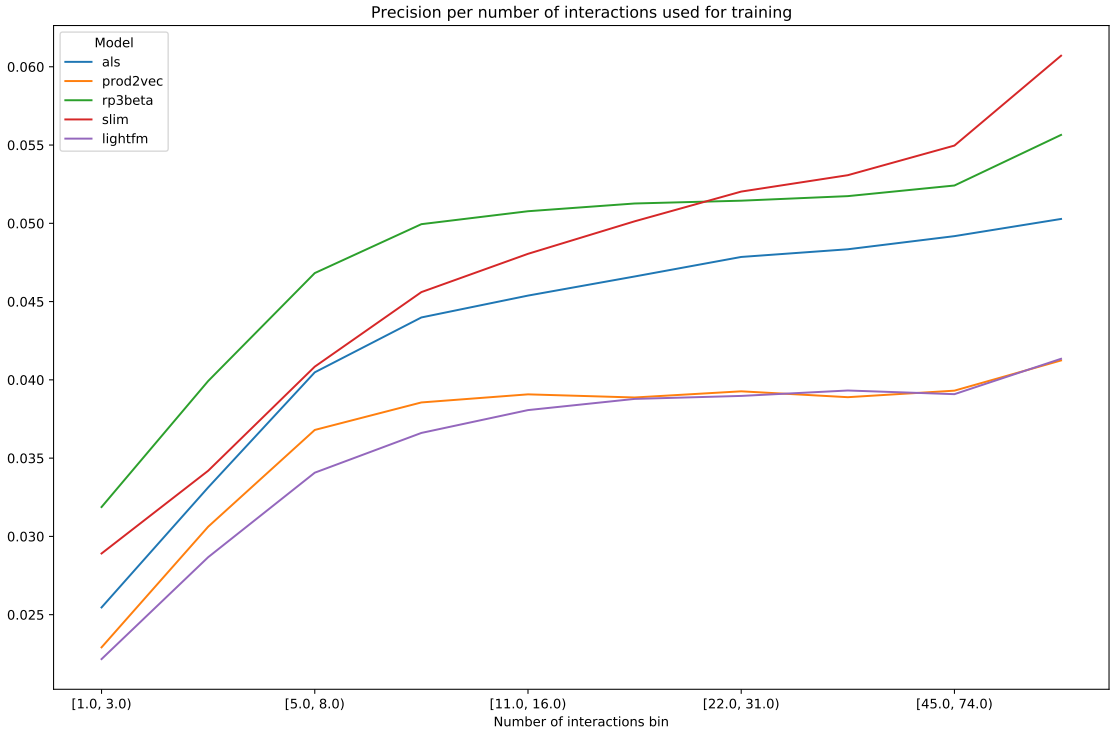

**Rysunek 4.3.** Wykres ilustrujący różnice między poszczególnymi parami algorytmów ze wskazaniem wartości różnicy krytycznej
Źródło: publikacja autora [78]

należy odpowiednią miarę podobieństwa między przedmiotami [32]. Podobieństwa między przedmiotami są bezpośrednio wyliczane, bez dostosowywania parametrów w procesie trenowania modelu. Model SLIM natomiast wykorzystuje metody uczenia maszynowego do uczenia się podobieństw między przedmiotami. Autor przypuszcza, że wyuczone w ten sposób podobieństwa przedmiotów są mniej obciążone, lecz mają większą wariancję, co prowadzi do lepszej skuteczności dla użytkowników z wieloma interakcjami, gdzie uśrednianie odbywa się na większej liczbie wyników.

**Istotność statystyczna różnic pomiędzy modelami RP3Beta i SLIM w zależności od liczby interakcji użytkownika**

W celu porównania dwóch metod dla wielu zbiorów danych, Demšar [35] zaleca stosowanie testu Wilcoxona dla par obserwacji (ang. *Wilcoxon signed-ranks test*, [131]). Test ten jest nieparametryczną alternatywą do testu t-Studenta dla prób zależnych. Pełen opis matematyczny znajduje się w pracy Demšara, w podrozdziale 3.1.3 [35].

W tabeli 4.4 możemy zaobserwować, że dla każdej grupy użytkowników różnica między modelami RP3Beta i SLIM jest statystycznie istotna.




**Rysunek 4.4.** Wartości metryki Precision@10 dla rozważanych modeli w zależności od liczby przedmiotów, z którymi użytkownik wszedł w interakcje w zbiorze treningowym
Źródło: publikacja autora [78]

**Tabela 4.4.** Wyniki testu Wilcoxona dla par obserwacji przeprowadzonego w celu porównania modeli RP3Beta i SLIM dla wszystkich rozważanych grup użytkowników

| **Liczba interakcji użytkowników** | ***p*-wartość** |
|---|---|
| [1,0, 3,0) | 0 |
| [3,0, 5,0) | 0 |
| [5,0, 8,0) | 0 |
| [8,0, 11,0) | 0 |
| [11,0, 16,0) | 0 |
| [16,0, 22,0) | 1,25e-6 |
| [22,0, 31,0) | 9,27e-7 |
| [31,0, 45,0) | 0 |
| [45,0, 74,0) | 0 |
| [74,0, 852,0) | 0 |

Źródło: publikacja autora [78]

**Tabela 4.5.** Wartości metryk pokrycia dla porównywanych metod

| Metryka | RP3Beta | SLIM | ALS | Prod2Vec | LightFM | TopPop | Random |
|---|---|---|---|---|---|---|---|
| Pokrycie zbioru testowego | 0,5725 | 0,5171 | 0,3038 | **0,7400** | 0,7031 | 0,0002 | 0,9778 |
| Entropia Shannona | 9,5271 | 9,6728 | 9,6270 | **10,4031** | 10,1385 | 2,3296 | 11,7267 |
| Indeks Giniego | 0,9083 | 0,9029 | 0,9120 | **0,7956** | 0,8397 | 0,9999 | 0,1159 |

Źródło: publikacja autora [78]

### 4.4.2. Metryki pokrycia

W serwisach ogłoszeniowych, zwłaszcza w przypadku ofert pracy, zwykle tylko jeden użytkownik jest potrzebny do zawarcia transakcji związanej z danym ogłoszeniem. W konsekwencji, wszyscy z wyjątkiem co najwyżej jednego wśród zainteresowanych użytkowników są odrzucani przez ogłoszeniodawcę. Z tego powodu staramy się unikać sytuacji, w której jedynie niewielka liczba przedmiotów jest rekomendowana użytkownikom. Przy podobnej wartości metryk dokładności, preferowane są metody dające większe pokrycie katalogu przedmiotów.

W celu oceny tego aspektu raportujemy wartości trzech miar: pokrycia zbioru testowego, entropii Shannona i indeksu Giniego, które zdefiniowaliśmy w sekcji 2.3.2. W tabeli 4.5 możemy zauważyć, że Prod2Vec i LightFM zapewniają największe pokrycie wśród spersonalizowanych metod. Pokrycie najdokładniejszych modeli, RP3Beta, SLIM i ALS, jest podobne, za wyjątkiem przypadku niższego pokrycia zbioru testowego dla modelu ALS.

### 4.4.3. Podobieństwo między rekomendacjami pochodzącymi z różnych modeli

Podczas ewaluacji offline, możemy czasami zaobserwować niewielką przewagę nowego modelu względem modelu wdrożonego wcześniej. Wdrożenie nowej metody wiąże się zwykle z koniecznością dostosowania i optymalizacji kodu oraz infrastruktury, a także przeprowadzeniem i analizą testu A/B pozwalającego na porównanie obu podejść. W związku z tymi kosztami, przed podjęciem decyzji, warto uwzględnić jak różne rekomendacje generują rozpatrywane metody. Jeśli nowa metoda generuje niemalże te same rekomendacje co już wdrożony model, możliwa do uzyskania poprawa jakości jest prawdopodobnie niewielka, a liczba użytkowników wymagana do uzyskania statystycznie istotnych wyników podczas testu A/B może być nieosiągalna. Może się jednak

**Tabela 4.6.** Wartości współczynników nakładania się dla analizowanych modeli

| Model | RP3Beta | SLIM | ALS | Prod2Vec | LightFM |
|---|---|---|---|---|---|
| RP3Beta | 100% | **73%** | 53% | 37% | 38% |
| SLIM | **73%** | 100% | 50% | 35% | 35% |
| ALS | 53% | 50% | 100% | 38% | 37% |
| Prod2Vec | 37% | 35% | 38% | 100% | 28% |
| LightFM | 38% | 35% | 37% | 28% | 100% |

Źródło: publikacja autora [78]

zdarzyć, że nowy model generuje wyraźnie odmienne rekomendacje, mimo że wartości rozważanych wcześniej metryk offline nie różnią się istotnie. W tej sytuacji możliwe, że nowy model zwraca uwagę na inne aspekty wiedzy, jaką posiadamy o użytkownikach i przedmiotach, co z kolei może trafniej oddawać rzeczywiste preferencje użytkowników. Możemy wtedy oczekiwać większych różnić pomiędzy rozważanymi modelami. Warto w tej sytuacji zbadać także jakość rekomendacji powstałych poprzez połączenie rekomendacji z obu modeli jednocześnie (na przykład budując listę rekomendacji poprzez wybieranie naprzemiennie rekomendacji generowanych z każdego z modeli).

W celu zbadania podobieństwa między rekomendacjami pochodzącymi z różnych modeli wykorzystamy współczynnik nakładania się (ang. *Overlap Coefficient* [119]). Współczynnik ten definiujemy wzorem:

$$\text{overlap}(\mathcal{R}_1, \mathcal{R}_2) = \frac{|\mathcal{R}_1 \cap \mathcal{R}_2|}{\min(|\mathcal{R}_1|, |\mathcal{R}_2|)},$$

gdzie $\mathcal{R}_1$ oraz $\mathcal{R}_2$ oznaczają dowolne zbiory. W naszym przypadku $\mathcal{R}_i$ oznacza zbiór par (użytkownik, przedmiot), dla których przedmiot został rekomendowany użytkownikowi przez $i$-ty model, $i \in \{1, 2\}$. Rozważane przez nas modele wygenerowały wszystkie $k = 10$ rekomendacji w przypadku co najmniej 99,9% użytkowników. Zatem dla dowolnych z rozważanych par modeli mamy $|\mathcal{R}_1| \approx |\mathcal{R}_2| \approx kn$, gdzie $n$ to liczba użytkowników.

Wartości współczynników nakładania się dla analizowanych modeli rekomendacji przedstawiliśmy w tabeli 4.6. Współczynnik ten jest stosunkowo wysoki dla pary modeli RP3Beta i SLIM. Wynika to prawdopodobnie z faktu, że obie te metody używają macierzy podobieństwa przedmiot-przedmiot, choć wyznaczanej w różny sposób. Nieco ponad połowa rekomendacji wygenerowanych przez model ALS, wygenerowana została również przez model RP3Beta. Najmniejszy współczynnik nakładania się z innymi modelami mają modele Prod2Vec i LightFM, które uzyskały również największe wartości metryk pokrycia. Niskie nakładanie się modeli ALS i LightFM pokazuje istotność wyboru funkcji straty w podejściach bazujących na faktoryzacji macierzy.

### 4.4.4. Skalowalność

Ostatni etap przeprowadzonej przez nas ewaluacji offline dotyczy skalowalności. W tym celu wszystkie modele trenowane były z wykorzystaniem serwisu AWS Sagemaker[4] z instancją ml.m5.4xlarge, 64 GB RAM, 16 vCPU Intel(R) Xeon(R) Platinum 8259CL CPU @ 2,50GHz w systemie operacyjnym Amazon Linux AMI 2018.03.

Opisane modele zostały poddane ocenie w odniesieniu do trzech operacji: wstępnego przetwarzania danych, trenowania modelu oraz generowania rekomendacji. W przypadku modelu RP3Beta, który nie posiada parametrów optymalizowanych w procesie uczenia, trenowanie rozumiemy jako proces znajdowania macierzy podobieństwa pomiędzy przedmiotami. Dla każdej operacji raportujemy maksymalne wykorzystanie pamięci, mierzone jako różnica pomiędzy największą zajmowaną pamięcią RAM podczas wykonywania operacji, a zajmowaną pamięcią RAM w momencie jej wywołania. Raportujemy także czas wykonania każdej operacji. Ewaluacja ta została przeprowadzona przy użyciu biblioteki tracemalloc[5]. Wyniki przedstawiliśmy na rysunku 4.5.

Zauważmy, że RP3Beta, ALS i LightFM mają znacznie niższy łączny czas wykonania wspomnianych operacji niż SLIM i Prod2Vec. W przypadku dwóch ostatnich, konieczna byłaby optymalizacja przed wdrożeniem w serwisach Pracodawcy, gdzie jak wspomnieliśmy wcześniej, modele wspólnej filtracji trenowane są nawet kilka razy dziennie. Model Prod2Vec ma najmniejsze wymagania odnośnie wykorzystania pamięci. Modele RP3Beta oraz SLIM wymagają około 30GB pamięci, co związane jest głównie z przechowywaniem macierzy podobieństw pomiędzy przedmiotami. Koszt trenowania modeli przy takiej pamięci jest akceptowalny (trening odbywa się na CPU).

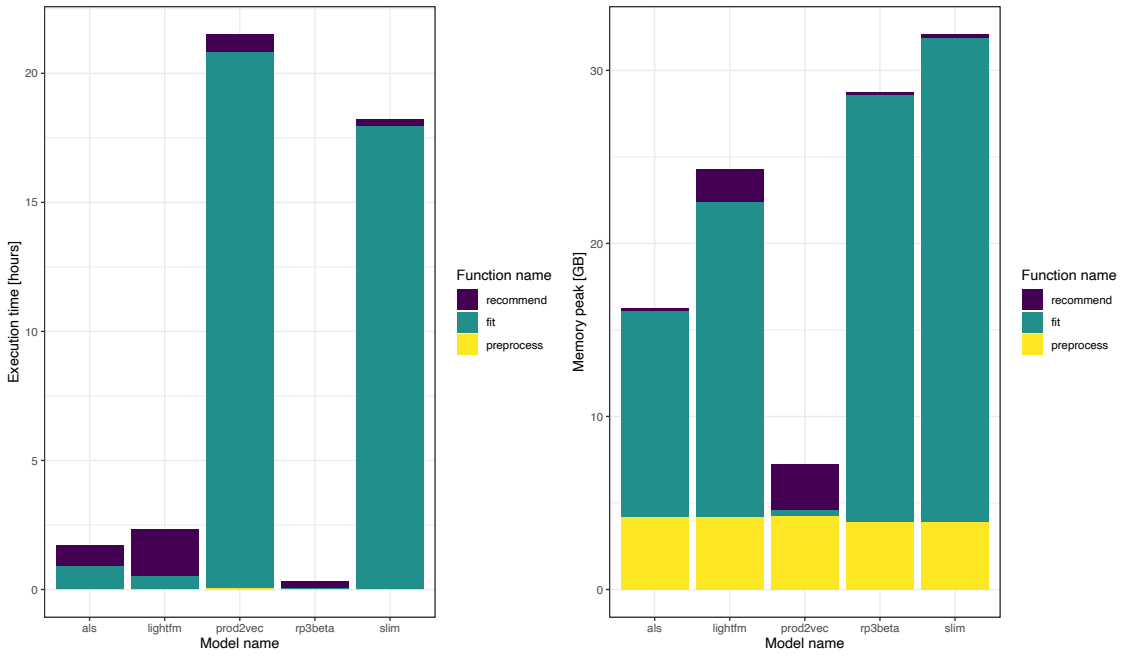
### 4.4.5. Modele wybrane do ewaluacji online

Na podstawie przeprowadzonej ewaluacji offline wybraliśmy modele do przetestowania w środowisku online z użytkownikami serwisów Pracodawcy. Pierwszym wybranym modelem był model RP3Beta, ponieważ uzyskał najwyższą wartość metryki Precision@10 wśród porównywanych metod. Ponadto, czas jego trenowania oraz generowania rekomendacji jest zdecydowanie niższy niż w przypadku pozostałych modeli. Wymagana pamięć jest stosunkowo duża, lecz dostatecznie mała, aby rozwiązanie mogło być wdrożone. Zbadanie wpływu metryk świadczących o dość niskim pokryciu tej metody może być interesującym kierunkiem przyszłych badań.

Drugim najlepszym modelem pod względem metryki Precision@10 jest SLIM. Niemniej jednak nie wybraliśmy tej metody do ewaluacji online ze względu na jej niską skalowalność i duży współczynnik nakładania się z już wybranym modelem RP3Beta. Poprawa wydajności implementacji SLIM mogłaby zmienić tę decyzję.

[4] `https://aws.amazon.com/pm/sagemaker/`, dostęp: 2023-11-18

[5] `https://docs.python.org/3/library/tracemalloc.html`, dostęp: 2023-11-18




**Rysunek 4.5.** Czas wykonania oraz maksymalne wykorzystanie pamięci dla operacji wstępnego przetwarzania danych (preprocess), trenowania modelu (fit) oraz generowania rekomendacji (recommend)
Źródło: publikacja autora [78]

Spośród innych modeli, najwyższą wartość metryki Precision@10 osiągnął model ALS. Ponadto, model ten został wdrożony przez nas w serwisach Pracodawcy jeszcze przed porównaniem go z innymi metodami. Zdecydowaliśmy się uwzględnić go w porównaniu online.

Mimo że Prod2Vec i LightFM generują najbardziej zróżnicowane rekomendacje, zdecydowaliśmy się nie testować ich online ze względu na znacznie gorszą wartość metryki Precision@10, a także wysoki czas wykonania w przypadku modelu Prod2Vec.

## 4.5. Ewaluacja online

Aby ocenić skuteczność wybranych metod, przeprowadziliśmy dwa testy A/B z użytkownikami serwisów Pracodawcy. Pierwszy etap naszego eksperymentu służył odpowiedzi na pytanie, czy wdrożenie wybranej metody rekomendacji zwiększa liczbę użytkowników odpowiadających na oferty pracy. Skupiliśmy się na użytkownikach, którzy niedawno wyświetlili ogłoszenia o pracę w serwisach Pracodawcy. Podzieliliśmy użytkowników na dwie grupy:

- **grupę kontrolną**, która nie otrzymywała rekomendacji,

**Tabela 4.7.** Liczba przekonwertowanych użytkowników dla poszczególnych wariantów testu A/B, w którym grupa kontrolna nie otrzymywała rekomendacji

| Wariant | Liczba użytkowników | % przekonwertowanych użytkowników |
|---|---|---|
| Grupa kontrolna | 129 308 | 15,98% |
| Grupa testowa | 1 170 262 | 16,83% |

Źródło: publikacja autora [78]

- **grupę testową**, której przesyłaliśmy rekomendacje wygenerowane przy użyciu modelu ALS.

Test został przeprowadzony przez 25 dni w marcu 2021 roku. Użytkownicy w grupie testowej otrzymywali wiadomości email z dziesięcioma rekomendowanymi ogłoszeniami o pracę, a gdy użytkownik zainstalował aplikację OLX, przesyłaliśmy również powiadomienia push z jednym rekomendowanym ogłoszeniem. Następnie obserwowaliśmy, czy użytkownik odpowiedział na jakąkolwiek ofertę pracy (niekoniecznie rekomendowaną) w ciągu następnych 48 godzin (użytkownik przekonwertowany). Wyniki przedstawiono w tabeli 4.7. Statystyczna istotność przewagi modelu ALS względem grupy kontrolnej została sprawdzona za pomocą testu chi-kwadrat, w którym uzyskaliśmy $p$-wartość równą 0. Eksperyment ten potwierdził naszą hipotezę, że rekomendacje wpływają na aktywność użytkowników. Odsetek przekonwertowanych użytkowników w grupie testowej był o $(16{,}83\% - 15{,}98\%)/15{,}98\% \approx 5{,}3\%$ wyższy niż w grupie kontrolnej. Zaobserwowany wzrost oznacza setki dodatkowych użytkowników odpowiadających na oferty pracy dziennie.

W drugim eksperymencie sprawdziliśmy, czy model RP3Beta, który osiągnął najlepsze wyniki podczas ewaluacji offline, osiągnie lepsze wyniki od modelu ALS podczas ewaluacji online. Podzieliliśmy naszych użytkowników na trzy grupy, otrzymujące rekomendacje z różnych systemów rekomendacji:

- ALS,
- RP3Beta,
- ALS+RP3Beta, w którym co druga rekomendacja wygenerowana została przez model ALS, co druga przez model RP3Beta.

Drugi eksperyment został przeprowadzony przez 28 dni w marcu i kwietniu 2021 roku. Wyniki przedstawiono w tabeli 4.8. Odsetek przekonwertowanych użytkowników jest tutaj obliczany w stosunku do wszystkich użytkowników, którym przesłane zostały rekomendacje. Zmiana systemu rekomendacji nie wpływa jednak na użytkowników, którzy nie otworzyli przesłanych przez nas wiadomości. Postanowiliśmy ich odfiltrować, ponieważ stanowią oni jedynie niepotrzebny szum w procesie wybrania najskutecz-

**Tabela 4.8.** Liczba przekonwertowanych użytkowników dla poszczególnych wariantów testu A/B porównującego modele ALS i RP3Beta

| Wariant | Liczba użytkowników | % przekonwertowanych użytkowników |
|---|---|---|
| ALS | 343 892 | 15,25% |
| RP3Beta | 345 273 | 15,40% |
| ALS+RP3Beta | 343 896 | 15,30% |

Źródło: publikacja autora [78]

**Tabela 4.9.** Liczba przekonwertowanych użytkowników dla poszczególnych wariantów testu A/B porównującego modele ALS i RP3Beta. Wyniki zostały ograniczone do użytkowników, którzy otworzyli wiadomości z rekomendacjami

| Wariant | Liczba użytkowników | % przekonwertowanych użytkowników |
|---|---|---|
| ALS | 44 775 | 19,66% |
| RP3Beta | 46 097 | 20,94% |
| ALS+RP3Beta | 45 469 | 20,59% |

Źródło: publikacja autora [78]

niejszego modelu rekomendacji. W tabeli 4.9 przedstawiamy wpływ rekomendacji na użytkowników, którzy otworzyli przesłane przez nas wiadomości. Przewaga modelu RP3Beta nad modelem ALS, zbadana testem chi-kwadrat, jest statystycznie istotna ($p$-wartość wynosi około $10^{-6}$). Różnica między wariantami RP3Beta i ALS+RP3Beta nie jest istotna statystycznie ($p$-wartość wynosi 0,19).

Wśród użytkowników, którzy otworzyli wiadomość, obserwujemy o 1,28 p.p. (punktu procentowego) więcej przekonwertowanych użytkowników w przypadku modelu RP3Beta niż dla modelu ALS. Odsetek osób otwierających wiadomości wynosił 13,2%. Zatem obserwowany wzrost liczby przekonwertowanych użytkowników przekłada się na wzrost o około $1{,}28 \cdot 0{,}132 \approx 0{,}17$ pp względem wszystkich testowanych użytkowników. W oparciu o wyniki pierwszego eksperymentu, ALS przynosi wzrost o około 0,85 p.p. w stosunku do grupy kontrolnej. Oznacza to, że zastąpienie modelu ALS przez model RP3Beta zwiększa nasz wpływ na wszystkich docelowych użytkowników o około $0{,}17/0{,}85 = 20\%$.

## 4.6. Podsumowanie

W rozdziale przeprowadziliśmy ewaluację offline oraz online wybranych metod rekomendacji. Zarówno w wyborze metod jak i ich ewaluacji zwracaliśmy szczególną uwagę na możliwość ich skutecznego wykorzystania w przypadku serwisów ogłoszeniowych, w szczególności w kategorii praca.

Podczas ewaluacji offline analizowaliśmy metryki dokładności, pokrycia oraz skalowalności. Ponadto, sprawdziliśmy podobieństwo między rekomendacjami pochodzącymi z różnych źródeł. Poniżej wymieniamy najważniejsze wnioski.

- W przypadku czterech z pięciu modeli optymalizacja hiperparametrów była kluczowa dla maksymalizacji metryki Precision@10.
- Model RP3Beta uzyskał najwyższe wartości wszystkich rozpatrywanych metryk dokładności.
- Model rekomendujący najpopularniejsze ogłoszenia uzyskał około 40 razy niższą wartość metryki Precision@10 od modelu RP3Beta.
- Poprzez analizę statystyczną pokazaliśmy występowanie istotnych różnic rozpatrywanych modeli względem metryki Precision@10 dla wszystkich rozpatrywanych par modeli, z wyjątkiem pary Prod2Vec i LightFM.
- Model SLIM jest istotnie lepszy od modelu RP3Beta względem metryki Precision@10 dla użytkowników posiadających co najmniej 22 interakcje.
- Najwyższe pokrycie uzyskały modele Prod2Vec oraz LightFM.
- W przypadku generowania zbioru 10 rekomendacji dla każdego użytkownika, średnio 73% rekomendacji generowanych przez modele RP3Beta oraz SLIM dotyczyło tych samych przedmiotów. Dla pozostałych par modeli współczynnik nakładania się był niższy.
- Model RP3Beta uzyskał najniższy czas trenowania modelu i generowania rekomendacji, lecz proces trenowania wymagał stosunkowo dużo pamięci.

Na podstawie wyników ewaluacji offline wybraliśmy modele RP3Beta oraz ALS do przeprowadzenie ewaluacji online z użytkownikami serwisów Pracodawcy. Przeprowadziliśmy dwa testy A/B. W każdym z nich uczestniczył ponad milion użytkowników otrzymujących rekomendacje poprzez powiadomienia email oraz push. W pierwszym teście pokazaliśmy, że wysyłanie rekomendacji ofert pracy z wykorzystaniem modelu ALS zwiększa liczbę użytkowników odpowiadających na ogłoszenia o pracę o ponad 5%. W drugim teście wykazaliśmy, że zastąpienie w tym zadaniu modelu ALS modelem RP3Beta pozwala nam uzyskać o około 20% większy wpływ na liczbę osób odpowiadających na oferty pracy. Wydajność modelu RP3Beta umożliwia trenowanie go wiele razy dziennie przy bardzo niskich kosztach. Dzięki temu jest on skutecznie stosowany w serwisach Pracodawcy. W kolejnych rozdziałach prezentujemy dwa sposoby rozwinięcia tego modelu.

Przedstawiona ewaluacja istniejących metod wypełnia lukę badawczą polegającą na braku kompleksowego porównania klasycznych modeli wspólnej filtracji w domenie ofert pracy. Autor opublikował zarówno kod źródłowy, jak i zbiór danych potrzebne do odtworzenia uzyskanych wyników. Umożliwia to innym badaczom pracę nad rozwojem modeli rekomendacji w domenie serwisów ogłoszeniowych.

W rozdziale zrealizowany został drugi cel pomocniczy rozprawy: *zaprezentowanie sposobu doboru i ewaluacji modeli rekomendacyjnych w przypadku serwisów ogłoszeniowych oraz przedstawienie wyników takiej ewaluacji dla metod opisanych w literaturze*.

Rozdział 5

# Generowanie rekomendacji w czasie rzeczywistym

Systemy wspólnej filtracji są zwykle zaprojektowane do tworzenia spersonalizowanych rekomendacji dla dużej liczby użytkowników w tym samym czasie [120]. Rozwiązania takie wydają się wystarczające dla niektórych praktycznych zastosowań, takich jak na przykład przedstawione w poprzednim rozdziale przesyłanie wiadomości email z rekomendacjami ofert pracy. Jednak w przypadku rekomendacji wyświetlanych na stronie internetowej, gdzie użytkownicy spędzają średnio kilkanaście minut, bardziej zasadne jest generowanie rekomendacji w czasie rzeczywistym uwzględniając niedawne interakcje użytkowników. W szczególności użytkownicy, którzy pierwszy raz odwiedzają daną stronę internetową, również mogą otrzymać wtedy spersonalizowane rekomendacje.

W rozdziale przedstawiamy model *RP3Beta real-time*, który jest zaproponowaną przez autora modyfikacją omawianego wcześniej modelu RP3Beta. Model ten pozwala na generowanie rekomendacji wykorzystujących informacje o interakcjach użytkowników wykonanych zaledwie kilka sekund wcześniej. Przedstawiamy zarówno matematyczny opis modelu jak i infrastruktury wykorzystanej do jego wdrożenia w serwisach Pracodawcy.

Zaproponowane podejście i infrastruktura mogą być wykorzystane przez wiele innych systemów rekomendacji. Szczególnie interesującym przypadkiem są modele, których rekomendacje dla użytkowników mogą być uzyskane na podstawie podobieństw pomiędzy przedmiotami. Wtedy rekomendacje modelu bazowego i jego odpowiednika produkującego rekomendacje w czasie rzeczywistym są jednakowe dla użytkownika posiadającego zadany zbiór interakcji. Rekomendacje w czasie rzeczywistym mogą być zatem lepsze od rekomendacji generowanych przez model bazowy ze względu na możliwość uwzględnienia dodatkowych interakcji użytkownika. Pokażemy, że opisywany przypadek obejmuje wiele popularnych modeli rekomendacji.

Przedstawiamy również wyniki testów A/B porównujących klasyczny model RP3Beta z modelem RP3Beta real-time. Zostały one przeprowadzone w serwisach Pracodawcy z udziałem prawie 200 000 użytkowników.

Wiele wyników prezentowanych w tym rozdziale pochodzi z publikacji autora [80].

## 5.1. Rekomendacje generowane w trybie wsadowym lub w czasie rzeczywistym

Proces umożliwiający dostarczanie użytkownikom rekomendacji możemy zwykle podzielić na dwa etapy: trenowanie modelu i generowanie rekomendacji [21]. Możemy wyróżnić trzy typy systemów ze względu na ich zachowanie zaraz po tym jak użytkownik wchodzi w interakcję z przedmiotem (np. wyświetla ogłoszenie lub ocenia przedmiot):

1. parametry modelu **nie są** aktualizowane, a interakcja **nie ma** wpływu na rekomendacje użytkownika,
2. parametry modelu **nie są** aktualizowane, ale interakcja **ma** wpływ na rekomendacje użytkownika,
3. parametry modelu **są** aktualizowane, a interakcja **ma** wpływ na rekomendacje użytkownika.

W pierwszym przypadku mówimy, że rekomendacje generowane są w **trybie wsadowym** (ang. *batch mode* [120]). Rozwiązanie to jest najtańsze i najprostsze do wdrożenia. Jest ono wystarczające, gdy nie jest istotne uwzględnienie najnowszych interakcji użytkownika oraz niedawno dodanych przedmiotów (jak w przypadku wcześniej wspomnianych rekomendacji ofert pracy za pomocą wiadomości email). Wcześniej wyliczone rekomendacje mogą być prezentowane użytkownikom na stronach internetowych, lecz ich preferencje mogą być inne niż w momencie generowania rekomendacji. Przykładowo, użytkownik mógł kupić wcześniej szukany przedmiot i poszukiwać przedmiotów z zupełnie innych kategorii. Ponadto anonimowi użytkownicy bez wcześniejszej historii interakcji nie otrzymają spersonalizowanych rekomendacji.

W drugim przypadku mówimy, że rekomendacje generowane są **w czasie rzeczywistym** (ang. *real-time* [88]). W tym przypadku najnowsze interakcje użytkownika wpływają na generowane dla niego rekomendacje, lecz nie wpływają na rekomendacje generowane dla innych użytkowników (ponieważ parametry modelu nie są modyfikowane). Przykładowo, model trenowany raz dziennie może generować reprezentacje wektorowe wszystkich przedmiotów, zaś reprezentacja wektorowa danego użytkownika może być obliczana w czasie rzeczywistym jako średnia z reprezentacji wektorowych przedmiotów, z którymi użytkownik ten wszedł w interakcje. W przeciwieństwie do rekomendacji generowanych w trybie wsadowym, podejście to adresuje problem zmiennych preferencji użytkownika w czasie (ang. *concept drift*). Możliwe jest także generowanie rekomendacji dla nowych użytkowników niemalże natychmiast po wykonaniu przez nich pierwszej interakcji z przedmiotami. Metody te nie są jednak w stanie rekomendować przedmiotów, które nie posiadały żadnych interakcji podczas ostatniego trenowania modelu (problem zimnego startu dla przedmiotów).

W trzecim przypadku parametry modelu aktualizowane są z każdą interakcją dowolnego użytkownika (ang. *stream-based recommender systems* [9, 21]). Podejście to jest najbardziej kosztowne do wdrożenia i utrzymywania, lecz może potencjalnie dawać najlepsze wyniki. Nawet w przypadku modeli wspólnej filtracji, metody te są w stanie rekomendować przedmioty niemalże natychmiast po pierwszej interakcji użytkownika [72].

W pracy porównujemy model generujący rekomendacje w trybie wsadowym z jego odpowiednikiem generującym rekomendacje w czasie rzeczywistym. Nie bierzemy pod uwagę modeli aktualizujących swoje parametry wraz z każdą interakcją ze względu na ich wysokie koszty wdrożenia w stosunku do oczekiwanej poprawy. Zdolność takich modeli do adresowania problemu zimnego startu dla przedmiotów byłaby dla nas użyteczna, lecz nie jest kluczowa (jak przykładowo w przypadku rekomendacji najnowszych wiadomości na portalach informacyjnych [72]).

### 5.1.1. Przegląd literatury

Viniski i in. [120] zauważyli, że modele rekomendacji trenowane są zwykle w trybie wsadowym i zbadali kilka popularnych podejść, takich jak SVD [97], BPRMF [104] oraz NeuMF [63]. Sharma i in. [108] opisali rozwój metod rekomendacji w serwisie Twitter oraz proces przejścia z modeli generujących rekomendacje w trybie wsadowym, do rozwiązań generujących rekomendacje w czasie rzeczywistym. Wspomnieli oni, że około roku 2012 *niemalże wszystkie rekomendacje w serwisie Twitter pochodzące z modeli grafowych generowane były w trybie wsadowym w przybliżeniu z dzienną częstotliwością*. Zaobserwowali oni wyższą skuteczność rekomendacji krótko po ich wygenerowaniu, co potwierdziło ich przypuszczenia, że *rekomendacje generowane raz dziennie nie wykorzystują w pełni możliwości serwisu Twitter*.

Linden, Smith i York [88] zaproponowali skalowalny model *item-to-item collaborative filtering* pozwalający na generowanie rekomendacji w czasie rzeczywistym i wdrożyli rozwiązanie w serwisie Amazon. Algorytm ten oblicza w trybie wsadowym dla każdego przedmiotu listę przedmiotów podobnych. Spersonalizowane rekomendacje generowane są poprzez agregację list przedmiotów podobnych do przedmiotów, z którymi zadany użytkownik wszedł w interakcje. Podejście to jest szeroko stosowane [21], ponieważ wyznaczanie przedmiotów podobnych do zadanego jest kluczowym elementem wielu popularnych modeli rekomendacji [15, 95, 98]. Metoda opisywana w rozdziale również oparta jest na tym podejściu. W pracy opisujemy jednak szczegółowo możliwość zastosowania go dla modelu RP3Beta, a także szerszej klasy wskazanych modeli. Ponadto, prezentujemy szczegóły implementacji, infrastruktury oraz wyniki testów online.

Znane są badania porównujące modele generujące rekomendacje w trybie wsadowym lub w czasie rzeczywistym z modelami aktualizującymi parametry modelu z każdą interakcją dowolnego użytkownika [120, 124]. Według naszej wiedzy nie ma badań

porównujących modele generujące rekomendacje w trybie wsadowym z analogicznymi modelami generującymi rekomendacje w czasie rzeczywistym. W pracy wypełniamy tę lukę.

## 5.2. Modele RP3Beta oraz RP3Beta real-time

W sekcji prezentujemy model RP3Beta generujący rekomendacje w trybie wsadowym oraz proponowany model RP3Beta real-time generujący rekomendacje w czasie rzeczywistym. W rozdziale 4.1.4 przedstawiliśmy model RP3Beta. W szczególności zauważyliśmy, że możemy obliczyć predykcje preferencji wszystkich użytkowników poprzez mnożenie trzech rzadkich macierzy:

$$\hat{\boldsymbol{R}} = \boldsymbol{P}^{(1)}\boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}, \tag{5.1}$$

gdzie macierze $\boldsymbol{P}^{(1)}$, $\boldsymbol{P}^{(3)}$ są wymiaru $|\mathcal{U}| \times |\mathcal{I}|$, zaś macierz $\boldsymbol{P}^{(2)}$ jest wymiaru $|\mathcal{I}| \times |\mathcal{U}|$. Powyższe równanie jest podstawą obu przedstawionych w kolejnych podrozdziałach podejść.

### 5.2.1. Model RP3Beta

W przypadku, gdy rekomendacje generowane są w trybie wsadowym, obliczamy jednocześnie rekomendacje dla wszystkich użytkowników za pomocą równania (5.1). Obliczone rekomendacje są zapisywane i mogą być użyte w dowolnym momencie w przyszłości. Rozwiązanie to zostało wdrożone w serwisach Pracodawcy w celu przesyłania rekomendacji za pomocą wiadomości email. Rekomendacje są w tym przypadku przesyłane dużej liczbie użytkowników natychmiast po ich wygenerowaniu.

Zdecydowaliśmy się wykorzystywać otrzymywane w ten sposób rekomendacje także w innych kanałach dostępu. W tym celu zapisujemy rekomendacje w bazie klucz-wartość (ang. *key-value store*), gdzie kluczami są identyfikatory użytkowników, zaś wartościami są rekomendacje. Model trenowany jest kilka razy dziennie, aby uwzględnić nowe przedmioty oraz interakcje użytkowników. Natychmiast po jego wytrenowaniu, generowane są rekomendacje dla użytkowników, którymi nadpisujemy rekomendacje wygenerowane wcześniej. Rekomendacje te mogą nie uwzględniać najnowszych interakcji użytkownika, a jedynie te wykonane przed ostatnim przeliczeniem modelu. Największą zaletą tego podejścia jest łatwość wdrożenia i niski koszt utrzymania.

### 5.2.2. Model RP3Beta real-time

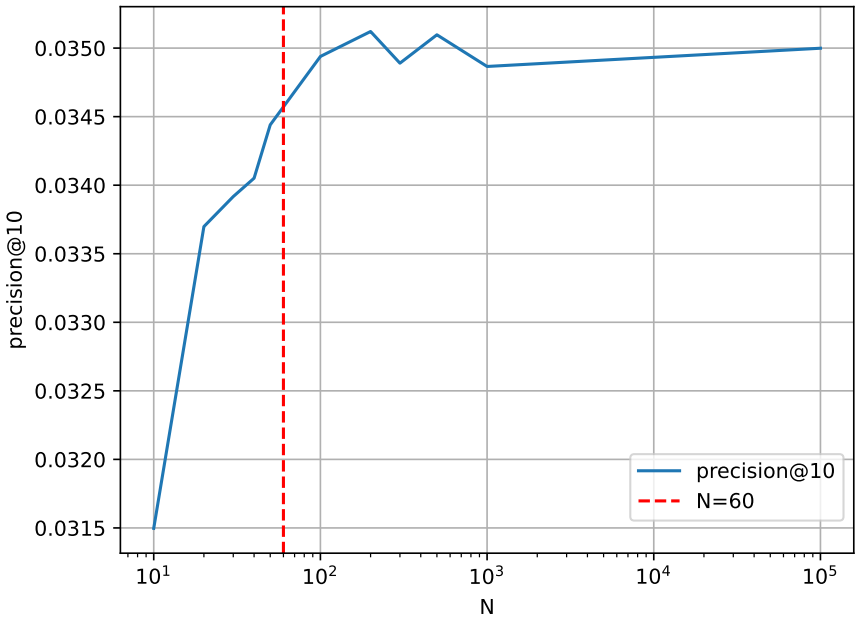
W sytuacji, w której chcielibyśmy w dowolnym momencie mieć możliwość uzyskania rekomendacji dla zadanego użytkownika uwzględniając wszystkie jego interakcje w procesie ich generowania, obliczenie iloczynu macierzy zgodnie z równaniem (5.1) jest

zbyt kosztowne obliczeniowo. Zauważmy, że w celu wygenerowania rekomendacji dla zadanego użytkownika $u$, wystarczające jest obliczenie iloczynu macierzy $\boldsymbol{P}^{(1)}(u)\boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$, gdzie $\boldsymbol{P}^{(1)}(u)$ jest wierszem macierzy $\boldsymbol{P}^{(1)}$ odpowiadającym użytkownikowi $u$. Część obliczeń może być wykonywana w trybie wsadowym. Iloczyn macierzy przedstawiony w równaniu (5.1) można obliczyć w następujących dwóch etapach.

1. Obliczenie macierzy $\boldsymbol{A} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$, która jest macierzą rzadką o wymiarach $|\mathcal{I}| \times |\mathcal{I}|$. Odbywa się to w trybie wsadowym - kilka razy dziennie.
2. Generowanie rekomendacji dla zadanego użytkownika poprzez iloczyn reprezentacji użytkownika $\boldsymbol{P}^{(1)}(u)$ oraz wcześniej obliczonej macierzy $\boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$. Operacja ta jest wykonywana w momencie, gdy potrzebne są rekomendacje dla zadanego użytkownika (na przykład gdy odwiedzi zadaną podstronę).

Ponieważ macierz $\boldsymbol{A}$ jest wyznaczana w trybie wsadowym, nie jesteśmy w stanie uwzględnić informacji o przedmiotach, z którymi wszystkie interakcje wykonane były po ostatnim wyliczeniu macierzy $\boldsymbol{A}$. Przedmioty takie nie będą rekomendowane żadnemu użytkownikowi. Ponadto, model nie jest w stanie wygenerować rekomendacji dla użytkowników, którzy weszli w interakcje wyłącznie z takimi przedmiotami. Z drugiej strony, w przypadku rekomendacji ofert pracy, prawdopodobnie niewielka liczba użytkowników weszła w interakcje z tymi przedmiotami w ciągu kilku godzin pomiędzy kolejnymi procesami obliczania macierzy $\boldsymbol{A}$. Ponadto, reprezentacje tych przedmiotów (tj. odpowiadające im wiersze i kolumny macierzy $\boldsymbol{A}$) mogą być mniej dokładne niż reprezentacje bardziej popularnych przedmiotów. Dlatego uważamy, że wyznaczenie macierzy $\boldsymbol{A}$ w równaniu (4.2) w trybie wsadowym zamiast w czasie rzeczywistym ma niewielki wpływ na ogólną jakość systemu rekomendacji w przypadku rekomendacji ofert pracy. Może to być jednak istotne ograniczenie w domenach, gdzie uwaga użytkowników jest bardziej skupiona na najnowszych przedmiotach (np. w przypadku wiadomości lub wpisów na portalach społecznościowych).

Liczba niezerowych elementów macierzy $\boldsymbol{A}$ jest dwa razy większa niż liczba różnych par przedmiotów odwiedzonych przez tego samego użytkownika. W rzeczywistych zastosowaniach rząd wielkości tej liczby może osiągać miliardy, czego konsekwencją jest duża pamięć wymagana do trenowania modelu RP3Beta (co obserwowaliśmy w rozdziale 4.4.4). Rozmiar macierzy $\boldsymbol{A}$ wpływa także na czas i koszt generowania rekomendacji. Dlatego też zdecydowaliśmy się ograniczyć każdą kolumnę macierzy $\boldsymbol{A}$ do $N$ największych wartości, zaś pozostałe wartości zastąpić zerami. Wybierając $N \geq |\mathcal{I}|$ nie wprowadzamy żadnych modyfikacji do macierzy $\boldsymbol{A}$. Na rysunku 5.1 możemy zaobserwować wpływ wyboru liczby $N$ na wartość metryki Precision@10 w przypadku wykorzystywanego przez nas zbioru danych zawierającego około $10^5$ przedmiotów. Dla $N = 100$ wartość tej metryki wynosi 0,03494, zaś zwiększenie wartości liczby $N$ ma niewielki wpływ na wartość metryki. Zaobserwowaliśmy podobną zależność od $N$ wartości innych metryk offline.

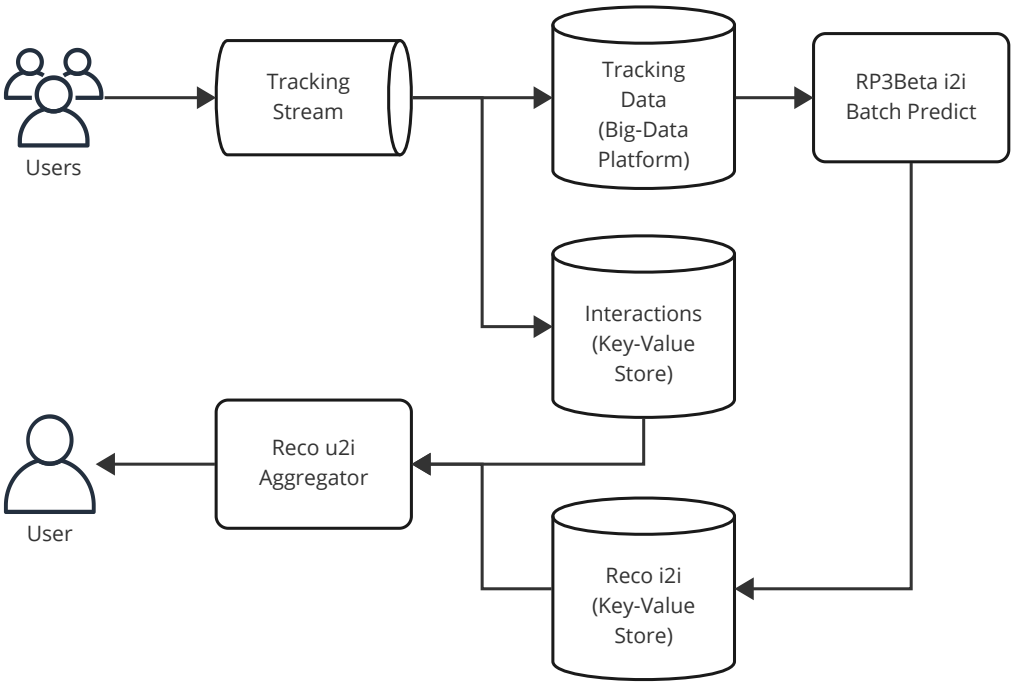



**Rysunek 5.1.** Wartości metryki Precision@10 po zastąpieniu zerami wszystkich z wyjątkiem $N$ największych wartości w każdej kolumnie macierzy $\boldsymbol{A}$, gdzie $N \in \{10, 20, 30, 40, 50, 100, 200, 300, 500, 1000, 10^5\}$
Źródło: publikacja autora [80]

W celu ograniczenia kosztów wdrażanego rozwiązania, zdecydowaliśmy się wybrać $N = 60$, dla którego wartości metryk są nieznacznie niższe niż dla $N \geq 100$.

## 5.3. Architektura umożliwiająca generowanie rekomendacji w czasie rzeczywistym

W sekcji podajemy szczegółowy opis zastosowanej infrastruktury pozwalającej na generowanie rekomendacji w czasie rzeczywistym. Ponadto, wskazujemy możliwość jej zastosowania dla wielu znanych modeli rekomendacji. Wskazujemy też warunki jakie musi spełniać model, aby rekomendacje generowane w czasie rzeczywistym były dokładnie takie same jak rekomendacje generowane w trybie wsadowym w sytuacjach, gdy zbiór interakcji użytkownika jest jednakowy.




**Rysunek 5.2.** Schemat architektury pozwalającej na dostarczanie użytkownikom rekomendacji w czasie rzeczywistym
Źródło: publikacja autora [80]

### 5.3.1. Opis architektury

Architektura pozwalająca na dostarczanie użytkownikom spersonalizowanych rekomendacji w czasie rzeczywistym przedstawiona jest na rysunku 5.2.

Interakcje użytkowników są rejestrowane i natychmiast przesyłane, a później zapisywane w dwóch różnych systemach: na platformie big data i w szybkiej bazie klucz-wartość (ang. *Key-Value Store*) zawierającej ostatnie interakcje użytkownika z przedmiotami. Przy odpowiedniej implementacji, baza klucz-wartość może odzwierciadlać najnowsze interakcje użytkowników niemalże natychmiast po wykonaniu przez nich interakcji. Platforma big data jest używana do przechowywania i odpytywania dużej ilości danych wykorzystywanych jako dane wejściowe do trenowania modelu (w przypadku modelu RP3Beta real-time trening ten oznacza znalezienie macierzy $\boldsymbol{A} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$). Po wytrenowaniu modelu, dla każdego przedmiotu generowana jest lista zawierająca zadaną liczbę $N$ przedmiotów podobnych wraz z wartościami miary podobieństwa. Listy te są zapisywane i przechowywane w szybkiej bazie klucz-wartość.

Generowanie rekomendacji odbywa się na żądanie użytkownika za pomocą komponentu nazwanego *agregatorem*, którego działanie przedstawiamy w Algorytmie 1.

**Algorytm 1.** Obliczenia wykonywane w procesie generowania rekomendacji w czasie rzeczywistym

```
R = słownik()
for i ∈ N_M(u) do
  for i', s_ii' ∈ S_N(i) do
    dołącz s_ii' do R[i']
  end for
end for
for i, s ∈ R do
  R[i] = suma(s)
end for
```

Agregator pobiera, z bazy klucz-wartość zawierającej dane o interakcjach użytkowników, zbiór $\mathcal{N}_M(u)$ ostatnich $M$ interakcji zadanego użytkownika. Następnie dla uzyskanych przedmiotów, z którymi użytkownik wszedł w interakcje, pobierane są zbiory przedmiotów podobnych z drugiej bazy klucz-wartość:

$$S(\mathcal{N}_M(u)) = \{\{(i', s_{ii'}) : i' \in \mathcal{S}_N(i)\} : i \ \in \mathcal{N}_M(u)\},$$

gdzie $s_{ij}$ oznacza wartość miary podobieństwa pomiędzy przedmiotami $i$ oraz $j$, zaś $\mathcal{S}_N(i)$ jest zbiorem $N$ przedmiotów najbardziej podobnych do przedmiotu $i$ (wyłączając przedmiot $i$). Zauważmy, że przedmiot $i'$ może pojawić się wielokrotnie, jako podobny do różnych przedmiotów z listy $\mathcal{N}_M(u)$. Predykcja preferencji użytkownika względem przedmiotu obliczana jest wtedy jako suma tych wyników. Zauważmy, że zamiast sumy możemy użyć innej funkcji agregującej, na przykład maksimum, średniej lub minimum. Moglibyśmy również uwzględnić liczbę lub rodzaj interakcji pomiędzy użytkownikiem a przedmiotem.

Wykorzystanie przedstawionej infrastruktury umożliwia generowanie rekomendacji wykonanych w ciągu kilku sekund od interakcji użytkownika przy zachowaniu niskich kosztów utrzymania infrastruktury (w stosunku do rozwiązań wymagających bardziej zaawansowanych obliczeń w czasie rzeczywistym).

### 5.3.2. Zastosowania opracowanej architektury

Wiele modeli rekomendacji umożliwia generowanie listy przedmiotów podobnych do zadanego. Przykładowo rekomendacje generowane dla użytkownika posiadającego interakcje tylko z jednym, zadanym przedmiotem, mogą być traktowane jako przedmioty podobne do niego. Możliwe jest zatem wygenerowanie listy przedmiotów podobnych dla wszystkich istniejących przedmiotów, a następnie wykorzystanie przedstawionej infrastruktury w celu generowania rekomendacji w czasie rzeczywistym. Dzięki temu

złożoność użytego modelu nie ma wpływu na koszt etapu generowania rekomendacji dla użytkowników. Ponadto, rekomendacje generowane w czasie rzeczywistym mogą być bardziej trafne od rekomendacji generowanych w trybie wsadowym ze względu na możliwość uwzględnienia najnowszych interakcji danego użytkownika.

Skuteczność proponowanego przez nas rozwiązania zależy w dużej mierze od jakości wyliczonych podobieństw między przedmiotami. Jedyną informacją specyficzną dla użytkownika braną pod uwagę podczas generowania rekomendacji są interakcje użytkownika. W rezultacie nasze podejście może dawać słabe wyniki, gdy cechy użytkownika są ważniejsze niż jego interakcje. W modelach wspólnej filtracji ryzyko to nie istnieje, ponieważ cechy użytkowników nie są wykorzystywane.

Istnieje też ryzyko pogorszenia jakości rekomendacji nawet w przypadku modeli, które nie wykorzystują cech użytkowników. W przedstawionym przez nas rozwiązaniu proces generowania rekomendacji przebiega w dwóch etapach: generowanie podobnych przedmiotów, a następnie agregacja tych wyników z użyciem najnowszych interakcji użytkownika. Jak pokazaliśmy, rozbicie procesu generowania rekomendacji na dwa etapy jest możliwe w przypadku modelu RP3Beta (co wynika z równania (5.1)). Przy zadanym zbiorze interakcji danego użytkownika, modele RP3Beta i RP3Beta real-time zwrócą dokładnie te same rekomendacje. Nie jest to jednak prawdą dla modeli faktoryzacji macierzy [74], gdzie rekomendacje generowane są na podstawie reprezentacji użytkowników znajdowanych w procesie trenowania modelu. Dwaj użytkownicy z dokładnie takim samym zbiorem interakcji mogą uzyskać w procesie trenowania różne reprezentacje, a w konsekwencji różne rekomendacje. Rekomendacje w czasie rzeczywistym wygenerowane za pomocą opisanej przez nas metody byłyby identyczne dla obu tych użytkowników. Wpływ różnicy pomiędzy rekomendacjami pochodzącymi z modelu bazowego a jego odpowiednika generującego rekomendacje w czasie rzeczywistym zależy od wykorzystywanego modelu bazowego. Zalecamy zatem przeprowadzenie ewaluacji offline porównującej oba podejścia przed podjęciem decyzji o wdrożeniu opisanej przez nas infrastruktury.

Pokażemy jednak, że dla szerokiej klasy modeli, przy zadanym zbiorze interakcji danego użytkownika, model bazowy (trenowany w trybie wsadowym) i jego odpowiednik generujący rekomendacje w czasie rzeczywistym zwrócą dokładnie te same rekomendacje (przy dostatecznie dużych wartościach $M$ i $N$ opisanych w rozdziale 5.3.1). Wśród modeli przedstawionych w sekcji 4.1, modele RP3Beta, SLIM oraz Prod2Vec spełniają tę własność. W przypadku modeli RP3Beta oraz SLIM, wynika to bezpośrednio z faktu, że predykcje ocen użytkownika wyliczane są jako iloczyn wektora reprezentującego jego interakcje oraz macierzy podobieństwa przedmiotów. Pokażemy, że model Prod2Vec spełnia zadaną własność poprzez wykazanie, że spełnia ją szersza klasa modeli, do której należy.

Rozważmy modele, dla których spełnione są warunki:

- predykcja oceny użytkownika $u$ względem przedmiotu $i$ obliczana jest jako **iloczyn**

**skalarny** reprezentacji wektorowej użytkownika i reprezentacji wektorowej przedmiotu,

- reprezentacja użytkownika jest średnią ważoną reprezentacji przedmiotów, z którymi użytkownik wszedł w interakcje.

Przyjmijmy, że reprezentacja użytkownika $u$ zadana jest wzorem:

$$\boldsymbol{x}_j = \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j,$$

gdzie: $\boldsymbol{y}_j$ oznacza reprezentację wektorową przedmiotu $j$, $w_{uj}$ oznacza wagę przypisaną zadanej parze $(u, j)$, zaś $\mathcal{N}(u)$ oznacza zbiór interakcji użytkownika $u$. Wtedy predykcję oceny zadanego przedmiotu $i$ przez użytkownika $u$ można wyrazić następująco:

$$\hat{r}_{ui} = \left( \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \right) \cdot \boldsymbol{y}_i = \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \cdot \boldsymbol{y}_i = \sum_{j \in \mathcal{N}(u)} w_{uj} s_{ji},$$

gdzie $s_{ji}$ oznacza miarę podobieństwa pomiędzy przedmiotami $j$ oraz $i$, zaś $\cdot$ oznacza iloczyn skalarny. Zatem rekomendacje mogą być obliczone za pomocą podobieństw pomiędzy parami przedmiotów, gdzie miara podobieństwa jest w tym przypadku iloczynem skalarnym reprezentacji przedmiotów. Sytuacja taka ma miejsce w opisywanym w sekcji 4.1.5 modelu Prod2Vec, gdzie przyjęlibyśmy $w_{uj} = 1$ dla wszystkich par $u \in \mathcal{U}$, $j \in \mathcal{I}$.

Często wykorzystywaną miarą podobieństwa jest odległość cosinusowa [37], czyli wartość cosinusa kąta pomiędzy reprezentacją użytkownika a przedmiotu. W tym przypadku możliwe jest obliczanie rekomendacji za pomocą podobieństw pomiędzy przedmiotami w następujący sposób:

$$\begin{aligned}
\hat{r}_{ui} &= \text{cosine}\Big( \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j, \boldsymbol{y}_i \Big) \\
&= \frac{\sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \cdot \boldsymbol{y}_i}{\| \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \| \| \boldsymbol{y}_i \|} \\
&= \frac{1}{\| \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \|} \sum_{j \in \mathcal{N}(u)} w_{uj} \| \boldsymbol{y}_j \| \frac{\boldsymbol{y}_j \cdot \boldsymbol{y}_i}{\| \boldsymbol{y}_j \| \| \boldsymbol{y}_i \|} \\
&= K_u \sum_{j \in \mathcal{N}(u)} w_{uj} \| \boldsymbol{y}_j \| s_{ji},
\end{aligned}$$

gdzie $K_u = \frac{1}{\| \sum_{j \in \mathcal{N}(u)} w_{uj} \boldsymbol{y}_j \|}$ oraz $s_{ji} = \text{cosine}(\boldsymbol{y}_j, \boldsymbol{y}_i)$. Zauważmy, że $K_u$ jest liczbą dodatnią, która nie zależy od przedmiotu $i$, dla którego ocenę estymujemy. Zatem nie ma ona wpływu na kolejność rekomendacji i może zostać pominięta. Zauważmy ponadto, że w kalkulacji potrzebna jest też długość wektora $\boldsymbol{y}_j$ reprezentacji każdego z przedmiotów, z którym użytkownik wszedł w interakcję. Wartość ta może być traktowana jako część wagi $w_{uj}$. Niestety przyjęcie wartości wag zależnych od cech przedmiotów (w tym przypadku długości wektora reprezentacji każdego przedmiotu) wymaga dodatkowego źródła

danych. Takim źródłem mogłaby być kolejna baza danych klucz-wartość, gdzie kluczami byłyby przedmioty, zaś wartościami ich cechy. Zwiększa to jednak złożoność systemu. Taka modyfikacja nie jest konieczna, jeśli reprezentacje przedmiotów są znormalizowane (tzn. $\|\boldsymbol{y}_j\| = 1$ dla każdego przedmiotu).

Pokazaliśmy zatem możliwość zastosowania proponowanej przez nas infrastruktury w przypadku wielu istniejących podejść generujących rekomendacje w trybie wsadowym.

## 5.4. Ewaluacja online

Przeprowadziliśmy testy A/B w celu porównania wdrożonego wcześniej modelu RP3Beta z zaproponowanym modelem RP3Beta real-time. Rekomendacje były prezentowane w zakładce profilu kandydata wspomnianego w rozdziale 1.3.1. Użytkownicy otrzymywali tam 30 spersonalizowanych rekomendacji ofert pracy. Rekomendacje pochodziły z dwóch rodzajów modeli: modelu wspólnej filtracji (model RP3Beta lub RP3Beta real-time) i modelu filtrowania opartego na treści (model Elasticsearch). Ostateczna lista rekomendacji tworzona była na podstawie tych dwóch źródeł. Dla uproszczenia możemy założyć, że znaczenie tych źródeł było podobne. Testowane były następujące warianty:

- **A (RP3Beta)**: rekomendacje pochodziły z modeli RP3Beta oraz Elasticsearch,
- **B (RP3Beta real-time)**: rekomendacje pochodziły z modeli RP3Beta real-time oraz Elasticsearch.

Każdy użytkownik został niezależnie przypisany do jednego z wariantów w sposób losowy z równym prawdopodobieństwem przypisania do każdego z nich. Użytkownik nie mógł zmienić wariantu podczas trwania eksperymentu. Test trwał dwanaście dni.

Generowanie rekomendacji modelu RP3Beta, a także wyznaczenie macierzy $\boldsymbol{A}$ w modelu RP3Beta real-time były wykonywane kilka razy dziennie na podstawie ostatnich siedmiu dni interakcji użytkowników. Eksperyment został przeprowadzony jednocześnie w serwisach OLX działających na różnych rynkach (w różnych krajach). Modele były trenowane dla każdego rynku osobno. Na największym rynku zbiory danych treningowych składały się z około 35 milionów interakcji wykonanych przez około 2 miliony użytkowników w odniesieniu do około 160 tysięcy ogłoszeń o pracę. Opublikowany przez nas zbiór danych opisywany w rozdziale 3 jest analogiczny do zbiorów wykorzystywanych podczas trenowania modeli.

Wyniki eksperymentu prezentujemy w tabeli 5.1. Przekonwertowany użytkownik to użytkownik, który odpowiedział na co najmniej jedno ogłoszenie o pracę wskutek rekomendacji. Obserwujemy około 10,14% wyższy odsetek przekonwertowanych użytkowników w wariancie B. Przewagę te sprawdziliśmy za pomocą testu chi-kwadrat, w którym otrzymaliśmy $p$-wartość wynoszącą 0,00001. Zatem możemy uznać, iż wariant $B$ jest istotnie lepszy od wariantu $A$ względem rozważanej metryki. Zauważmy, że zmieniony został jedynie model wspólnej filtracji, zaś wpływ zmiany zmierzony został

**Tabela 5.1.** Wyniki testu A/B porównującego modele RP3Beta oraz RP3Beta real-time. Różnica prezentowana jest jako procentowa przewaga wartości danej metryki dla wariantu B ponad wartość tej metryki dla wariantu A

| Wariant | Liczba użytkowników | % przekonwertowanych użytkowników |
|---|---|---|
| A (RP3Beta) | 92,835 | 4,02% |
| B (RP3Beta real-time) | 92,194 | 4,43% |
| Różnica | -0,69% | 10,14% |

Źródło: publikacja autora [80]

na całym systemie. Uzyskana przewaga była dostatecznie duża, abyśmy zdecydowali się na zastąpienie modelu RP3Beta przez model RP3Beta real-time.

Warto także wspomnieć o dwóch kwestiach wymienionych poniżej.

1. W systemie rekomendacji RP3Beta dodatkowo zmieniliśmy kolejność rekomendacji na podstawie dopasowania profilu użytkownika do rekomendowanych ogłoszeń. W przeszłości zaobserwowaliśmy, że zmiana kolejności zwiększa liczbę przekonwertowanych użytkowników o około 2,5%, więc zdecydowaliśmy się na wdrożenie takiego rozwiązania. Przeprowadzenie testu porównującego model RP3Beta bez zmiany kolejności z modelem RP3Beta real-time byłoby bardziej wartościowe naukowo. Niestety taki test zmuszałby nas do prezentowania części użytkownikom rekomendacji z modelu RP3Beta, o którym wiedzieliśmy, że jest gorszej jakości od modelu RP3Beta z przesortowanymi rekomendacjami. W modelu RP3Beta real-time rekomendacje nie były przesortowane.
2. Podczas przeprowadzania eksperymentu popełniliśmy błąd w implementacji rekomendacji w czasie rzeczywistym, a mianowicie przypadkowo transponowaliśmy macierz $\boldsymbol{A}$. Nie było możliwe ponowne przeprowadzenie eksperymentu porównującego rekomendacje generowane w trybie wsadowym z rekomendacjami generowanymi w czasie rzeczywistym, ponieważ zostało już udowodnione, że model RP3Beta real-time, nawet z błędem w implementacji, jest lepszy od modelu RP3Beta. W związku z tym przeprowadziliśmy kolejny eksperyment porównujący błędną i poprawną implementację modelu RP3Beta real-time. Eksperyment trwał 10 dni. Zaobserwowaliśmy o 4,7% więcej przekonwertowanych użytkowników w poprawionym wariancie modelu.

W związku z tym przewaga modelu RP3Beta real-time generującego rekomendacje w czasie rzeczywistym nad modelem RP3Beta generującym rekomendacje w trybie wsadowym jest prawdopodobnie większa niż podano w tabeli 5.1.

## 5.5. Podsumowanie

W rozdziale przedstawiliśmy szczegółowy opis infrastruktury pozwalającej na generowanie rekomendacji w czasie rzeczywistym. Rozwiązanie to zostało wdrożone w serwisach Pracodawcy działających na różnych rynkach.

Pokazaliśmy również sposób wykorzystania wdrożonej infrastruktury dla uzyskania rekomendacji w czasie rzeczywistym w przypadku modelu RP3Beta. Przedstawiliśmy wyniki testów A/B przeprowadzonych z użytkownikami serwisów Pracodawcy. Zastąpienie modelu RP3Beta generującego rekomendacje w trybie wsadowym przez model RP3Beta generujący rekomendacje w czasie rzeczywistym, zwiększyło liczbę osób odpowiadających na rekomendowane oferty pracy o ponad 10%. Autorowi nie jest znana praca prezentująca podobne porównanie.

Za pomocą zaproponowanej infrastruktury możliwe jest wdrożenie wielu modeli rekomendacji. W szczególności, wskazaliśmy warunki, przy których model bazowy oraz jego odpowiednik generujący rekomendacje w czasie rzeczywistym zwrócą dokładnie te same rekomendacje (przy zadanym zbiorze interakcji użytkownika).

Opublikowane przez nas wyniki mogą pomóc innym organizacjom w podjęciu świadomej decyzji w kwestii wdrożenia systemu generującego rekomendacje w czasie rzeczywistym.

W rozdziale zrealizowany został trzeci cel pomocniczy rozprawy: *opisanie procesu skutecznego wdrożenia modeli rekomendacyjnych w serwisach ogłoszeniowych Pracodawcy*. Ponadto, częściowo zrealizowany został główny cel rozprawy: *przedstawienie nowych metod rekomendacji opracowanych dla serwisów ogłoszeniowych oraz wykazanie ich przewagi względem metod opisanych w literaturze*.

Rozdział 6

# Model P3LTR — uogólnienie modelu RP3Beta

W rozdziale 4 pokazaliśmy przewagę modelu RP3Beta nad pozostałymi wybranymi metodami. W wyniku tych badań model RP3Beta został wdrożony w serwisach Pracodawcy we wszystkich kanałach dostępu (co opisaliśmy w podrozdziale 1.3.2). Kierunkiem dalszych prac stało się rozwinięcie tego modelu w celu uzyskania lepszych wyników. Możemy wyróżnić trzy główne typy działań, które w tym celu podjęliśmy.

1. **Przesortowanie rekomendacji** - chcieliśmy poprawić jakość rekomendacji poprzez zmianę kolejności rekomendacji generowanych przez zadany model (w tym przypadku model RP3Beta). Szczegóły opisaliśmy w podrozdziale 1.3.2.
2. **Generowanie rekomendacji w czasie rzeczywistym** - w rozdziale 5 zaproponowaliśmy odpowiednie w tym celu podejście oraz wykazaliśmy jego skuteczność.
3. **Uogólnienie modelu RP3Beta** - w rozdziale opisujemy stworzony przez nas model uczenia maszynowego, który adresuje ograniczenia modelu RP3Beta. W podrozdziale 7.3.5 zaproponujemy rozwiązanie wykorzystujące grafowe sieci neuronowe.

Proponowany model uczenia maszynowego, **P3 Learning to Rank (P3LTR**), w przeciwieństwie do modelu RP3Beta, pozwala na **uwzględnienie cech interakcji**, a także umożliwia **optymalizację parametrów w procesie treningu**.

W praktycznych zastosowaniach interakcje pomiędzy użytkownikiem a przedmiotem zawierają często więcej informacji niż jedynie fakt ich wystąpienia. Przykładowo w przedstawionym w rozdziale 3 zbiorze danych użytkownik może wejść w interakcje z tym samym przedmiotem wielokrotnie, znany jest czas tych interakcji oraz ich typ. Wykorzystanie tych informacji pozwala nam ocenić, które z przedmiotów, z którymi użytkownik wszedł w interakcje, preferuje on najbardziej. Dzięki temu możemy zwiększyć wpływ takich przedmiotów na generowane rekomendacje.

Model RP3Beta nie posiada parametrów, które znajdowane są w procesie uczenia. Posiada on jedynie dwa hiperparametry ($\alpha$ i $\beta$), których optymalny wybór może nie być wystarczający do pełnego wykorzystania wiedzy zawartej w danych. Ponadto uwzględnienie cech interakcji wymagałoby prawdopodobnie zwiększenia liczby hiperparametrów. Przeszukiwanie wielowymiarowej przestrzeni hiperparametrów może być kosztowne,

ponieważ sprawdzenie każdej konfiguracji wymaga trenowania modelu od nowa. Z tego powodu, proponujemy uogólnienie modelu RP3Beta, w którym model trenowany jest za pomocą technik uczenia maszynowego. Prezentujemy także sposób uwzględnienia przez ten model cech interakcji pomiędzy użytkownikami a przedmiotami.

W rozdziale przedstawiamy opis matematyczny proponowanego modelu, sposób jego trenowania, a także odpowiednie w tym celu funkcje straty. Przedstawiamy także wyniki ewaluacji offline, w której porównujemy model RP3Beta z modelem P3LTR na prezentowanym w rozdziale 3 zbiorze danych OLX Jobs Interactions.

Wiele wyników prezentowanych w tym rozdziale pochodzi z publikacji autora [79].

## 6.1. Opis modelu P3LTR

W podrozdziale prezentujemy opis matematyczny proponowanego modelu P3LTR, parametry modelu oraz sposób jego trenowania.

### 6.1.1. Opis modelu

Część opisu proponowanej przez nas metody P3LTR jest identyczna jak w przypadku modelu RP3Beta prezentowanego w sekcji 4.1.4. Dla kompletności jednak przedstawiamy pełny opis.

Niech $\mathcal{U}$ będzie zbiorem użytkowników, a $\mathcal{I}$ zbiorem przedmiotów. Przez $\hat{r}_{ui}$ oznaczamy predykcję oceny przedmiotu $i \in \mathcal{I}$ przez użytkownika $u \in \mathcal{U}$. Macierz wszystkich predykcji oznaczamy przez $\hat{\boldsymbol{R}}$. Model rekomenduje przedmioty, dla których $\hat{r}_{ui}$ osiąga największe wartości, z wyłączeniem przedmiotów, z którymi użytkownik wszedł już w interakcje.

Reprezentujemy zbiór danych za pomocą grafu dwudzielnego, w którym wierzchołkami są użytkownicy i przedmioty, a krawędzie reprezentują interakcje między nimi. Niech $\mathcal{N}(x)$ oznacza zbiór wierzchołków połączonych z wierzchołkiem $x$.

Predykcja oceny $\hat{r}_{ui}$ to suma wartości przypisanych do ścieżek długości 3 łączących danego użytkownika $u$ oraz przedmiot $i$:

$$\hat{r}_{ui} = \sum_{i' \in \mathcal{N}(u)} \sum_{u' \in \mathcal{N}(i')} p(u, i', u', u),$$

gdzie $p(u, i', u', u)$ to wartość przypisana do danej ścieżki. Podobnie jak w modelu RP3Beta, przedstawiamy tę wartość jako iloczyn wartości przypisanych do krawędzi, z których złożona jest zadana ścieżka:

$$p(u, i', u', u) = p^{(1)}_{ui'} p^{(2)}_{i'u'} p^{(3)}_{u'i},$$

gdzie $p_{xy}^{(k)}$ jest wartością przypisaną do krawędzi łączącej wierzchołki $x$ oraz $y$ w $k$-tej warstwie, $k = 1, 2, 3$. Wartości krawędzi przykładowej ścieżki pokazane są na rysunku 4.1 w podrozdziale 4.1.4, gdzie zdefiniowaliśmy model RP3Beta.

Obliczenie preferencji ocen dla wszystkich użytkowników może być wykonane poprzez mnożenie macierzy

$$\boldsymbol{R} = \boldsymbol{P}^{(1)}\boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}, \tag{6.1}$$

gdzie $\boldsymbol{P}^{(k)} = (p_{xy}^{(k)})$.

W modelu P3LTR obliczamy wartości krawędzi jako funkcje cech wierzchołków oraz cech krawędzi:

$$p_{xy}^{(k)} = \phi^{(k)}(f_x^n, f_y^n, f_{xy}^e),$$

gdzie $f_x^n$ jest wektorem cech wierzchołka $x$, $f_{xy}^e$ jest wektorem cech krawędzi łączącej wierzchołki $x$ oraz $y$, zaś $\phi^{(k)}$ jest dowolną funkcją rzeczywistą (np. siecią neuronową). Funkcje $\phi^{(k)}$ dla $k = 1, 2, 3$ nazywać będziemy **koderami cech**. Poniżej zaproponujemy konkretną postać koderów cech, którą wykorzystamy później do trenowania modelu P3LTR na zbiorze danych OLX Jobs Interactions opisanym w rozdziale 3.

Załóżmy, że zbiór danych składa się z następujących informacji na temat interakcji: użytkownik, przedmiot, typ interakcji, znacznik czasowy. Ponadto pomiędzy zadaną parą (użytkownik, przedmiot) może wystąpić wiele interakcji. Niech $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$ będzie zbiorem wszystkich możliwych typów interakcji (np. wyświetlenie przedmiotu, zakup przedmiotu, zapisanie przedmiotu do ulubionych). Dla takiego zbioru danych definiujemy następujące cechy:

- $|\mathcal{N}(x)|$ – stopień wierzchołka $x$ (liczba różnych wierzchołków połączonych z wierzchołkiem $x$),
- $\mathrm{rec}(x, y)$ – liczba dni, jaka minęła od najnowszej interakcji użytkownika ($x$ lub $y$) z danym przedmiotem ($y$ lub $x$) do najnowszej interakcji tego użytkownika z jakimkolwiek przedmiotem,
- $\mathrm{ev}(e_i, x, y)$ – liczba interakcji typu $e_i$ pomiędzy wierzchołkami $x$ oraz $y$,
- $\mathrm{ev}(x, y)$ – liczba interakcji pomiędzy wierzchołkami $x$ oraz $y$.

Wtedy wartość przypisana do zadanej krawędzi łączącej wierzchołek $x$ z wierzchołkiem $y$ obliczana jest następująco:

$$\begin{aligned} p_{xy}^{(k)} =& |\mathcal{N}(y)|^{-d^{(k)}} \\ & \cdot e^{-\mathrm{rec}(x,y)r^{(k)}} \\ & \cdot \sigma\left(\sum_{i \in |\mathcal{E}|} \frac{\mathrm{ev}(e_i, x, y)}{\mathrm{ev}(x, y)} e_i^{(k)} + b_e^{(k)}\right) \\ & \cdot \sigma(\mathrm{ev}(x, y)e^{(k)} + b^{(k)}), \end{aligned} \tag{6.2}$$

gdzie $\sigma(x) = \frac{1}{1+e^{-x}}$ oraz $d^{(k)}$, $r^{(k)}$, $e_i^{(k)}$, $b_e^{(k)}$, $e^{(k)}$, $b^{(k)}$ są parametrami modelu przyjmującymi wartości rzeczywiste.

### 6.1.2. Parametry modelu

Pokażemy, że model RP3Beta jest szczególnym przypadkiem modelu P3LTR. Przyjmijmy $d^{(1)} = d^{(2)} = \alpha$, $d^{(3)} = \beta$ oraz wartość 0 dla wszystkich pozostałych parametrów. Wtedy równanie (6.2) przyjmie postać

$$p_{xy}^{(k)} = |\mathcal{N}(y)|^{-d^{(k)}} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{|\mathcal{N}(y)|^{-d^{(k)}}}{4}.$$

Zatem:

$$p(u, i', u', u) = p_{ui'}^{(1)} p_{i'u'}^{(2)} p_{u'i}^{(3)} = \frac{1}{4^3} \cdot \frac{1}{|\mathcal{N}(i')|^{\alpha}} \cdot \frac{1}{|\mathcal{N}(u')|^{\alpha}} \cdot \frac{1}{|\mathcal{N}(i)|^{\beta}}. \tag{6.3}$$

Przypomnijmy, że w przypadku modelu RP3Beta równanie to ma postać:

$$p(u, i', u', i) = \frac{1}{|\mathcal{N}(u)|^{\alpha}} \cdot \frac{1}{|\mathcal{N}(i')|^{\alpha}} \cdot \frac{1}{|\mathcal{N}(u')|^{\alpha}|} \cdot \frac{1}{\mathcal{N}(i)|^{\beta}}. \tag{6.4}$$

Zauważmy, że wartość ścieżki dla modelu RP3Beta jest iloczynem wartości ścieżki dla modelu P3LTR oraz liczby $C = \frac{4^3}{|\mathcal{N}(u)|^{\alpha}}$. Liczba ta jest dodatnia i zależy wyłącznie od użytkownika $u$, niezależnie od przedmiotu $i$, dla którego obliczamy predykcję. W przypadku obu modeli predykcja oceny użytkownika względem danego przedmiotu jest sumą wartości przypisanych do ścieżek długości 3 łączących użytkownika z przedmiotem. Zatem predykcje ocen danego użytkownika względem wszystkich przedmiotów dla modelu RP3Beta są iloczynem predykcji ocen przedmiotów dla modelu P3LTR pomnożonych przez stałą $C$. Wynika z tego, iż modele te zwracają dokładnie te same rekomendacje przy tak ustalonych wartościach parametrów modelu P3LTR. Inicjalizując parametry modelu P3LTR w zadany sposób i pomijając proces ich optymalizowania podczas trenowania modelu, model P3LTR redukuje się do modelu RP3Beta. Celem optymalizacji parametrów podczas treningu jest poprawa jakości modelu, więc model P3LTR może potencjalnie uzyskać lepsze wyniki od modelu RP3Beta.

Model RP3Beta jest uogólnieniem modelu P3Alpha [28] (dla $\beta = 0$), P3 [28] (dla $\alpha = 1$, $\beta = 0$) oraz #3-Paths [28] (dla $\alpha = 0$, $\beta = 0$). Zatem model P3LTR jest także uogólnieniem tych metod.

Łączna liczba parametrów modelu P3LTR przy zadanych koderach cech wynosi $3 \cdot (5 + |\mathcal{E}|)$. Wartości parametrów wpływają na wartości przypisane do poszczególnych krawędzi, a w konsekwencji wartości przypisane do poszczególnych ścieżek. Niewielka liczba parametrów umożliwia ich interpretowanie poprzez bezpośrednie identyfikowanie **wpływu poszczególnych cech** na wartości przypisane do krawędzi.

Parametry $d^{(k)}$ określają **wpływ stopni wierzchołków** tworzących daną krawędź na wartość przypisaną do tej krawędzi. W szczególności, $d^{(k)} = 0$ oznacza, że traktujemy

wszystkie wierzchołki jednakowo, $d^{(k)} > 0$ oznacza, że zmniejszamy wpływ wierzchołków o większym stopniu, zaś $d^{(k)} < 0$ oznacza, że zwiększamy wpływ wierzchołków o większym stopniu. Zauważmy, że bardziej popularne przedmioty mogą występować w większej liczbie ścieżek długości 3 łączących użytkownika z przedmiotem. Zatem zmniejszenie wartości przypisanych do tych ścieżek poprzez przyjęcie $d^{(k)} > 0$ może zniwelować większy wpływ popularnych przedmiotów na estymacje ocen. W szczególności, im większą wartość parametru $d^{(3)}$ przyjmiemy, tym mniej popularne przedmioty będą rekomendowane.

Parametry $r^{(k)}$ określają **wpływ czasu wykonywanych interakcji** na wartość przypisaną do danej krawędzi. W szczególności, dla $r^{(k)} > 0$ nowsze interakcje mają większe znaczenie niż starsze interakcje, dla $r^{(k)} = 0$ czas interakcji nie ma wpływu na wartość krawędzi, zaś przy $r^{(k)} < 0$ starsze interakcje mają większe znaczenie niż nowsze.

Parametry $e_i^{(k)}$, $b_e^{(k)}$ określają **wpływ typu interakcji** pomiędzy użytkownikiem a przedmiotem na wartość przypisaną do danej krawędzi. Spodziewamy się, że interakcje wymagające większego zaangażowania użytkownika (np. odpowiedź na ogłoszenie) powinny mieć większy wpływ na wartość krawędzi od interakcji wymagających mniejszego zaangażowania (np. wyświetlenie ogłoszenia). W celu oddzielenia wpływu częstotliwości interakcji od wpływu ich typu, zastosowaliśmy średnią ważoną, gdzie wagi określają częstość występowania danego typu interakcji względem wszystkich interakcji pomiędzy zadanymi wierzchołkami.

Parametry $e^{(k)}$, $b^{(k)}$ określają **wpływ częstotliwości interakcji** pomiędzy użytkownikiem a przedmiotem na wartość przypisaną do danej krawędzi. Przy $e^{(k)} > 0$ przedmioty, z którymi użytkownik wszedł w interakcje wielokrotnie, mają większy wpływ na wartość krawędzi od przedmiotów z mniejszą liczbą interakcji, dla $e^{(k)} = 0$ krotność interakcji nie jest uwzględniana, zaś dla $e^{(k)} < 0$ większa liczba interakcji zmniejsza wpływ na wartość krawędzi.

### 6.1.3. Trenowanie modelu

Celem uczenia modelu jest optymalizacja wartości parametrów koderów cech $\phi^{(k)}$. Przedstawiamy opis tego procesu wyodrębniając trzy elementy: krok w przód (ang. *forward pass*) dla pojedynczego użytkownika, pętlę treningową i funkcję straty.

Kod źródłowy zawierający implementację modelu P3LTR opublikowany został w repozytorium Github autora[1].

#### Krok w przód dla pojedynczego użytkownika

Przedstawiamy nasz model wykorzystując techniki przekazywania wiadomości (ang. *message passing*) stosowane w grafowych sieciach neuronowych, które omawiamy w roz-

[1] `https://github.com/rob-kwiec/olx-jobs-recommendations`, dostęp: 2023-12-02

dziale 7. Przyjęliśmy takie podejście, ponieważ w trakcie opracowywania modelu P3LTR autorowi nie była znana biblioteka umożliwiająca wydajne mnożenie macierzy rzadkich wraz z możliwością automatycznego obliczenia gradientów potrzebnych do aktualizacji parametrów modelu. Zatem w procesie uczenia nie było możliwe bezpośrednie wykorzystanie reprezentacji macierzowej prezentowanej w równaniu (6.1). Równanie to wykorzystujemy jednak w procesie generowania rekomendacji.

Rozważmy graf dwudzielny, w którym wierzchołki reprezentują użytkowników oraz przedmioty. Krawędź pomiędzy parą wierzchołków istnieje wtedy i tylko wtedy, gdy jeden z wierzchołków reprezentuje użytkownika, zaś drugi wierzchołek reprezentuje przedmiot, z którym ten użytkownik wszedł w co najmniej jedną interakcję.

Niech dany będzie użytkownik $u$, dla którego rekomendacje chcemy obliczyć. Przyjmijmy następujące wartości początkowe przypisane wszystkim wierzchołkom:

- $r_u^{(0)} = 1$,
- $r_x^{(0)} = 0$, dla pozostałych wierzchołków.

Reprezentacje wierzchołków w kolejnych krokach $k = 1, 2, 3$ obliczane są za pomocą równania:

$$r_x^{(k)} = \sum_{y \in N(x)} r_y^{(k-1)} \phi^{(k)}(f_x^n, f_y^n, f_{xy}^e). \tag{6.5}$$

Proces ten możemy zinterpretować jako rozprzestrzenianie się wartości po grafie. W momencie $k = 0$ cała wartość skupiona jest w jednym wierzchołku. W pierwszym kroku wierzchołek ten przesyła przypisaną sobie wartość do sąsiadujących wierzchołków, przy czym przekazywana wartość zależy od funkcji $\phi^{(1)}$ (suma przesyłanych wartości może być większa od wartości w wierzchołku początkowym). Następnie każdy z wierzchołków posiadających niezerową wartość, przesyła ją do swoich sąsiadów w zależności od funkcji $\phi^{(2)}$. Proces ten mógłby być kontynuowany, lecz ze względu na wydajność, podobnie jak w modelu RP3Beta, ograniczamy się do trzech kroków. Predykcja $\hat{r}_{ui}$ oceny przedmiotu $i$ przez użytkownika $u$ dana jest wzorem $\hat{r}_{ui} = r_i^{(3)}$. Ponadto, przyjmujemy $\hat{r}_{ui} = 0$, jeśli użytkownik $u$ wszedł w interakcję z przedmiotem $i$, aby podobnie jak w podejściach opisywanych w rozdziale 4, nie rekomendować przedmiotów, z którymi użytkownik wszedł już w interakcję.

### Pętla treningowa

Pętla treningowa modelu P3LTR opisana jest przez Algorytm 2. Wykorzystuje ona opisany wyżej krok w przód dla pojedynczego użytkownika. W kolejnej sekcji omówimy szczegółowo wspomnianą w algorytmie funkcję straty.

### Funkcja straty

W celu trenowania modelu P3LTR autor proponuje własne funkcje straty. Zanim je przedstawimy wyjaśnimy dlaczego standardowe podejście mogłoby być nieskuteczne w przypadku tego modelu. W standardowym podejściu model optymalizowany jest

**Algorytm 2.** Pętla treningowa modelu P3LTR

**for** iteracja = 1, 2, . . . , liczba_iteracji **do**
  Oblicz wartości przypisane do krawędzi grafu na podstawie koderów cech.
  Ustaw zerową wartość straty.
  **for** i = 1, 2, . . . , rozmiar_partii **do**
    Wybierz losowego użytkownika, który wszedł w interakcję z minimum dwoma przedmiotami.
    Oznacz przedmiot, z którym użytkownik wszedł w ostatnią wykonaną interakcję jako **wierzchołek walidacyjny** i usuń go ze zbioru interakcji użytkownika.
    Wykonaj *krok w przód dla tego użytkownika*. Zapisz wartości predykcji ocen $k$ przedmiotów, dla których są one najwyższe. Zapisz pozycję i predykcję oceny wierzchołka walidacyjnego.
    Oblicz wartość *funkcji straty* dla danego użytkownika, a następnie dodaj ją do obecnej straty.
  **end for**
  Wykonaj wsteczną propagację błędu i zaktualizuj wartości koderów cech.
**end for**

w taki sposób, aby predykcje ocen przedmiotów, z którymi użytkownik wszedł w interakcję były wyższe od predykcji przedmiotów, z którymi użytkownik nie wszedł w interakcję. Przykładem są przedstawione w podrozdziale 4.1.2 funkcje straty oparte na parach: BPR [104], WARP [129] oraz k-OS WARP [130]. Innym przykładem jest zastosowana w modelu ALS przedstawionym w rozdziale 4.1.1 funkcja straty opisana równaniem (4.1). W przypadku modelu P3LTR predykcja oceny przedmiotu $i$ przez użytkownika $u$ wyliczana jest jako suma wartości przypisanych do ścieżek długości 3. Ścieżki długości 3 łączące użytkownika $u$ z przedmiotem $i$, z którym użytkownik ten wszedł w interakcję możemy podzielić na cztery typy:

1. ścieżkę $(u, i, u, i)$,
2. ścieżki $(u, i', u, i)$, gdzie $i' \neq i$, których jest $|\mathcal{N}(u)| - 1$,
3. ścieżki $(u, i, u', i)$, gdzie $u' \neq u$, których jest $|\mathcal{N}(i)| - 1$,
4. ścieżki $(u, i', u', i)$, gdzie $i' \neq i$ oraz $u' \neq u$.

Zauważmy, że w przypadku ścieżek łączących użytkownika $u$ z przedmiotem, z którym użytkownik ten nie wszedł w interakcję, istnieją jedynie ścieżki czwartego typu. Zatem przedmioty, z którymi użytkownik wszedł w interakcję, prawdopodobnie posiadają więcej ścieżek długości 3 łączących je z użytkownikiem $u$ od przedmiotów, z którymi użytkownik ten nie wszedł w interakcje. W konsekwencji, optymalizowanie różnicy pomiędzy predykcjami dla przedmiotów, z którymi użytkownik wszedł w interakcję względem pozostałych przedmiotów, mogłoby faworyzować te zestawy parametrów,

w których liczba ścieżek ma duży wpływ na predykcje (przykładowo, gdy wszystkie parametry przyjmą wartość 0, każda ścieżka ma jednakową wartość). Postępowanie takie niekoniecznie prowadziłoby do poprawy jakości rekomendacji.

W celu rozwiązania opisanych problemów standardowego podejścia zdecydowaliśmy się zignorować krawędź pomiędzy zadanym użytkownikiem a przedmiotem, z którym wszedł on w ostatnią interakcję. Wierzchołek ten nazwaliśmy wierzchołkiem walidacyjnym, jak przedstawiono w Algorytmie 2. Rola tego wierzchołka w procesie predykcji jest analogiczna do roli przedmiotów, z którymi użytkownik nie wszedł w interakcję, tzn. prowadzą do niego wyłącznie ścieżki czwartego typu.

Zaproponowane przez nas funkcje straty mają na celu uzyskanie możliwie najwyższej pozycji wierzchołka walidacyjnego na liście rekomendowanych przedmiotów. Zdefiniujmy

$$q = \frac{\text{średnia predykcja oceny dla } k \text{ najwyższych predykcji}}{\text{predykcja oceny dla wierzchołka walidacyjnego}}.$$

Dodatkowo, dla uzyskania szybszej zbieżności modelu, zapobiegamy uzyskaniu przez parametry zbyt dużych wartości poprzez wprowadzenie regularyzacji. Niech reg oznacza sumę kwadratów parametrów pomnożoną przez stały hiperparametr regularyzacji. Ponadto niech poz_wal oznacza pozycję przedmiotu walidacyjnego na liście rekomendowanych przedmiotów.

Proponujemy trzy funkcje straty:

1. **iloraz predykcji**: $q + \text{reg}$,
2. **logarytm ilorazu predykcji**: $\log(q) + \text{reg}$,
3. **wzmocniony logarytm ilorazu predykcji**: $\log(q) \cdot \text{poz\_wal} + \text{reg}$.

Inspiracją do przekształcenia wartości *ilorazu predykcji* przez funkcję logarytmiczną była postać powszechnie stosowanej funkcji BPR. Zaobserwowaliśmy poprawę jakości modelu po wykonaniu tego przekształcenia. W przypadku funkcji *wzmocnionego logarytmu ilorazu predykcji* chcemy szybciej aktualizować parametry modelu, gdy predykcja jest daleka od oczekiwanej, tj. gdy pozycja przedmiotu walidacyjnego jest wysoka. Podobne rozumowanie wykorzystują przedstawione wcześniej funkcje WARP i k-OS WARP.

Przedstawione funkcje straty traktowane są jako hiperparametry modelu P3LTR. Funkcja straty dająca najlepsze wyniki zostanie wybrana w procesie ich optymalizacji.

Zauważmy, że zaproponowane podejście może zostać zastosowane także dla innych modeli, co może być tematem dalszych badań.

## 6.2. Zalety modelu P3LTR

Poniżej wymieniamy zalety proponowanego modelu.

1. Model P3LTR jest **uogólnieniem modelu RP3Beta** dającego dobre wyniki w wielu zastosowaniach [31, 32].

2. Model P3LTR wykorzystuje informacje na temat **typu, częstotliwości i czasu interakcji**, dopuszczając także możliwość wystąpienia wielokrotnych interakcji pomiędzy zadanym użytkownikiem a przedmiotem. W przypadku ocen jawnych (np. w skali od 1 do 5) nasz model mógłby traktować każdą z nich jako osobny typ interakcji, a następnie nauczyć się wpływu jaki interakcja danego typu powinna mieć na generowane rekomendacje.
3. Model P3LTR może wykorzystywać **cechy użytkowników i przedmiotów**. W pracy koncentrujemy się na modelach wspólnej filtracji, dlatego zaproponowane przez nas kodery cech uwzględniają jedynie cechy, które mogą być obliczone na podstawie interakcji (stopnie wierzchołków). Moglibyśmy jednak zdefiniować kodery cech w oparciu o inne cechy, czyniąc jednocześnie model P3LTR modelem hybrydowym.
4. Predykcje modelu P3LTR są **wyjaśnialne** z dwóch powodów. Po pierwsze, podobnie jak w modelu RP3Beta, możemy wskazać przedmioty mające największy wpływ na estymacje oceny zadanej rekomendacji spośród przedmiotów, z którymi użytkownik wszedł w interakcje. Po drugie, poprzez analizę wartości koderów cech, możemy wyjaśnić dlaczego niektóre z interakcji użytkownika mają większy od innych wpływ na wygenerowane rekomendacje.
5. Model P3LTR wykorzystuje **relacje sąsiedztwa w procesie generowania predykcji**. Jak wspomnieliśmy w podrozdziale 4.4.1, takie podejście może dawać lepsze wyniki niż metody oparte na znajdowaniu reprezentacji wektorowych użytkowników i przedmiotów, szczególnie w przypadku użytkowników z niewielką liczbą interakcji.
6. Opisany proces trenowania modelu w celu znalezienia optymalnych parametrów może odbywać się **sporadycznie**, o ile definiując kodery cech nie wykorzystamy identyfikatorów użytkowników ani przedmiotów (np. stosując *one-hot encoding* [59]). Zatem w przypadku zaproponowanych przez nas postaci koderów cech nie musimy trenować modelu przed każdą predykcją.
7. Generowanie rekomendacji jest niemalże tak **wydajne** jak w przypadku modelu RP3Beta. Różnica znajduję się jedynie w procesie wstępnego przetwarzania danych, gdzie w przypadku modelu P3LTR musimy obliczyć wartości cech wierzchołków i interakcji, a następnie, za pomocą wytrenowanych wcześniej koderów cech, obliczyć wartości przypisane do każdej krawędzi. Następnie obliczanie predykcji ocen dla przedmiotów w przypadku obu modeli może być wykonane za pomocą równania (6.1).
8. Infrastruktura zaproponowana w rozdziale 5 może być wykorzystana do generowania rekomendacji **w czasie rzeczywistym** za pomocą modelu P3LTR. Podobnie jak w modelu RP3Beta, generowanie rekomendacji dla użytkownika $u$ za pomocą równania (6.1) może być wykonane w dwóch etapach: obliczenia macierzy $\boldsymbol{A} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$ w trybie wsadowym, a następnie obliczenia iloczynu $\boldsymbol{P}^{(1)}(u)\boldsymbol{A}$ w czasie rzeczywistym. W tym przypadku powinniśmy jednak zwrócić uwagę na możliwość szybkiego uzyskania

cech potrzebnych do wyliczenia wektora $\boldsymbol{P}^{(1)}(u)$ (wektora wag interakcji użytkownika). Możliwe w tym celu jest stworzenie baz danych klucz-wartość, umożliwiających uzyskanie cech użytkownika, przedmiotu oraz interakcji, co jednak generuje dodatkowe koszty. Zauważmy, że przyjmując postać kodera $\phi^{(1)} = 1$ dodatkowe cechy nie są potrzebne. Zatem przed podjęciem decyzji o wdrożeniu rozwiązania, zalecamy sprawdzić wpływ przyjęcia takiej postaci kodera na metryki offline. Oczywiście, możemy też przyjąć postać kodera, która wykorzystuje jedynie te cechy, które jesteśmy w stanie łatwo uzyskać w czasie rzeczywistym.

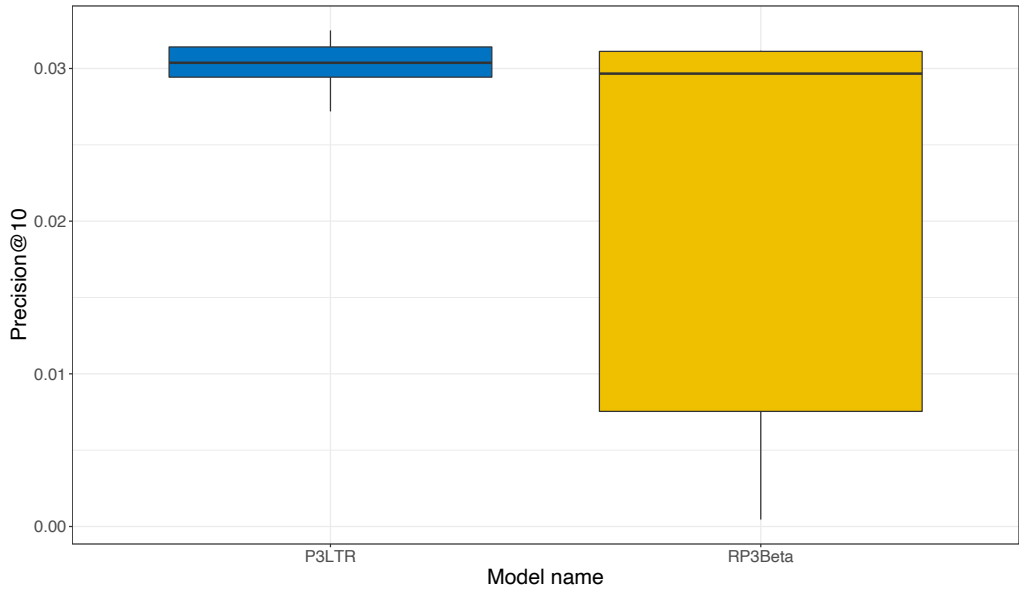
## 6.3. Ewaluacja

W kolejnych sekcjach przedstawiamy wyniki ewaluacji offline modelu P3LTR na opublikowanym przez autora zbiorze danych przedstawionym w rozdziale 3. Przyjęliśmy ten sam podział na zbiór testowy i treningowy co w rozdziale 4, dzięki czemu wyniki mogą być bezpośrednio porównane z wynikami modeli prezentowanymi w rozdziale 4.

### 6.3.1. Optymalizacja hiperparametrów

Wykorzystaliśmy metodę przeszukiwania przestrzeni hiperparametrów przedstawioną w podrozdziale 4.3. Plik konfiguracyjny definiujący możliwe wartości hiperparametrów jest dostępny w repozytorium Github autora[2]. Rysunek 6.1 przedstawia wyniki przedstawionej optymalizacji. Możemy zaobserwować, że wybór nieoptymalnych hiperparametrów dla modelu RP3Beta może doprowadzić do bardzo słabej jakości modelu mierzonej metryką Precision@10. W przypadku modelu P3LTR wybór ten jest znacznie mniej istotny. Jednym z powodów tego zjawiska jest fakt, że rolę hiperparametrów modelu RP3Beta ($\alpha$ oraz $\beta$) przejmują w modelu P3LTR parametry $d^{(k)}$ optymalizowane podczas treningu. Właściwe przeszukiwanie przestrzeni hiperparametrów jest kosztowne i czasami nie jest przeprowadzane zarówno w praktycznych zastosowaniach, jak i w pracach naukowych [31]. Zatem dobre wyniki modelu P3LTR uzyskiwane dla większości wybranych hiperparametrów są dodatkową zaletą tego modelu. Optymalne hiperparametry dla obu modeli raportujemy w tabeli 6.1.

[2] `https://github.com/rob-kwiec/olx-jobs-recommendations/blob/main/src/tuning/config.py`, dostęp: 2023-12-02




**Rysunek 6.1.** Wykres pudełkowy wartości metryki Precision@10 dla modeli RP3Beta oraz P3LTR w zależności od wybranych wartości hiperparametrów
Źródło: publikacja autora [79]

**Tabela 6.1.** Wartości hiperparametrów modeli P3LTR oraz RP3Beta, dla których wartość metryki Precision@10 jest najwyższa wśród testowanych możliwości

| Model | Hiperparametry modelu |
|---|---|
| **RP3beta** | {'alpha': 0,61447198, 'beta': 0,1443548} |
| **P3LTR** | {'regularization': 0,001, 'learning_rate': 0,02, 'batch_size': 153, 'iterations': 80, 'top_k': 205, 'loss': 'log_ratio'} |

Źródło: publikacja autora [79]

### 6.3.2. Porównywane metody

Wytrenowaliśmy modele P3LTR oraz RP3Beta na pełnym zbiorze danych wykorzystując hiperparametry przedstawione w tabeli 6.1. Następnie wygenerowaliśmy rekomendacje dla wszystkich 619 389 użytkowników w zbiorze testowym. Ponadto dodaliśmy do porównania dwie metody będące szczególnymi przypadkami modelu RP3Beta:

- model **P3** [28], do którego redukuje się model PR3Beta dla $\alpha = 1$ oraz $\beta = 0$,
- model **#3-Paths** [28], do którego redukuje się model PR3Beta dla $\alpha = 0$ oraz $\beta = 0$.

Przyjęliśmy początkową wartość równą zero dla wszystkich parametrów modelu

**Tabela 6.2.** Wartość metryk dokładności dla porównywanych metod

| Metryka | **P3LTR** | RP3Beta | P3 | #3-Paths |
|---|---|---|---|---|
| Precision | **0,0515** | 0,0484 | 0,0481 | 0,0391 |
| Recall | **0,0817** | 0,0783 | 0,0782 | 0,0611 |
| NDCG | **0,0798** | 0,0759 | 0,0755 | 0,0599 |
| MAP | **0,0414** | 0,0393 | 0,0390 | 0,0302 |
| MRR | **0,1423** | 0,1365 | 0,1363 | 0,1107 |
| LAUC | **0,5408** | 0,5391 | 0,5391 | 0,5305 |
| HR | **0,3242** | 0,3131 | 0,3147 | 0,2605 |

Źródło: publikacja autora [79]

P3LTR. Zatem model #3-Paths jest równoważny modelowi P3LTR przed rozpoczęciem procesu uczenia.

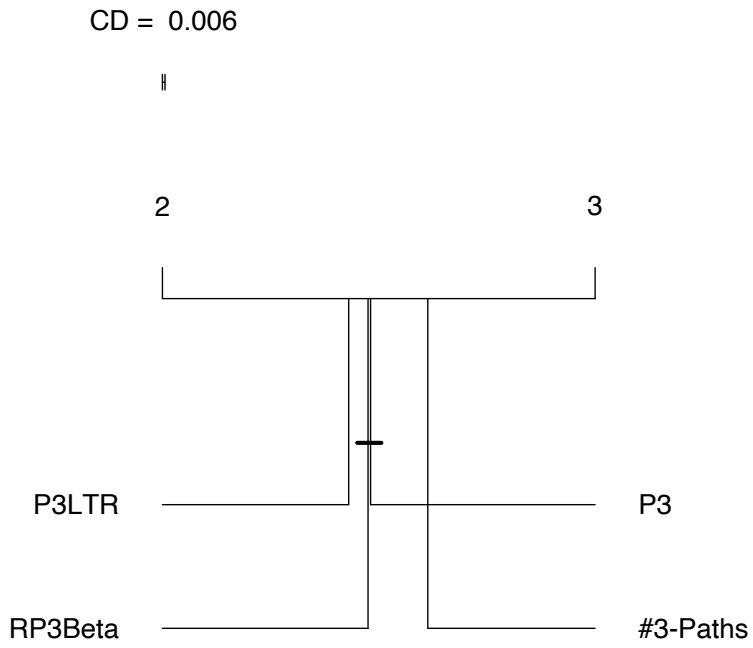
W kolejnych podrozdziałach prezentujemy wyniki przeprowadzonej ewaluacji.

### 6.3.3. Metryki dokładności

W tabeli 6.2 raportujemy wyniki metryk dokładności dla $k = 10$ rekomendacji. Możemy zauważyć, że proponowana przez nas metoda P3LTR uzyskuje lepsze wyniki od modelu RP3Beta dla wszystkich raportowanych metryk. Zwróćmy także uwagę na niskie wartości metryk w przypadku modelu #3-Paths. Przed rozpoczęciem treningu model P3LTR był równoważny temu modelowi. Dzięki optymalizacji parametrów podczas treningu, model P3LTR był w stanie nauczyć się prawidłowości pozwalających na uzyskanie znacznie lepszych wyników.

Podobnie jak w podrozdziale 4.4.1, przedstawiamy analizę statystyczną w celu rozstrzygnięcia czy metoda P3LTR daje istotnie lepsze wyniki ze względu na wartość metryki Precision@10 od metody RP3Beta.

W celu stwierdzenia istnienia różnic pomiędzy rozpatrywanymi metodami, testujemy hipotezę zerową o braku różnic pomiędzy nimi. W tym celu przeprowadzamy test Friedmana [44, 45] z rozszerzeniem zaproponowanym przez Imana oraz Davenporta [67]. W teście tym uzyskaliśmy $p$-wartość równą zero co pozwala nam odrzucić hipotezę zerową. Dzięki temu możemy przejść do przeprowadzenie testów post-hoc dla wszystkich par rozpatrywanych modeli. Wykorzystujemy w tym celu test Nemenyi. Wyniki wizualizujemy za pomocą zaproponowanego przez Demšara [35] wykresu. Przypomnijmy, że na wykresie tym, algorytmy, które nie są połączone linią traktowane są jako różne. W naszym przypadku, przyjmując poziom istotności $\alpha = 0{,}05$, dowolne dwa algorytmy, których różnica pomiędzy średnimi rangami jest większa od 0,006 traktowane są jako




**Rysunek 6.2.** Wykres ilustrujący różnice między poszczególnymi parami algorytmów ze wskazaniem wartości różnicy krytycznej
Źródło: publikacja autora [79]

różne. Na podstawie rysunku 6.2 możemy zauważyć, że model P3LTR jest istotnie lepszy od pozostałych modeli, zaś model #3-Paths jest istotnie gorszy. Na przyjętym poziomie istotności nie możemy natomiast odrzucić hipotezy zerowej o braku różnic pomiędzy modelami RP3Beta oraz P3.

### 6.3.4. Metryki pokrycia

Jak wspomnieliśmy w rozdziale 4.4.2, w przypadku rekomendacji ofert pracy staramy się unikać rekomendowania tego samego ogłoszenia dużej liczbie użytkowników. W tabeli 6.3 raportujemy w tym celu wartości metryk pokrycia dla rozpatrywanych metod. Model P3LTR charakteryzuje się największym pokryciem ze względu na wszystkie rozpatrywane metryki. Zauważmy ponadto, że wartość pokrycia zbioru testowego jest wyższa niż w przypadku wszystkich spersonalizowanych modeli rozpatrywanych w tabeli 4.5 w rozdziale 4.4.2. Jedną z wad modelu RP3Beta jest niższe pokrycie względem modeli Prod2Vec i LightFM. Możemy uznać, że model P3LTR pozbawiony jest tej wady. Dzięki temu może on uzyskać potencjalnie lepszy wpływ na metryki biznesowe od modelu RP3Beta.

Możemy również zauważyć, że metoda #3-Paths charakteryzuje się najniższym pokryciem ze względu na analizowane metryki. Metoda ta rekomenduje przedmioty na

**Tabela 6.3.** Wartości metryk pokrycia dla ewaluowanych modeli

| Metryka | **P3LTR** | RP3Beta | P3 | #3-Paths |
|---|---|---|---|---|
| Pokrycie zbioru testowego | **0,757** | 0,573 | 0,617 | 0,374 |
| Entropia Shannona | **10,090** | 9,527 | 9,631 | 8,712 |
| Indeks Giniego | **0,844** | 0,908 | 0,898 | 0,957 |

Źródło: publikacja autora [79]

**Tabela 6.4.** Wartości współczynników nakładania się dla analizowanych modeli

| Metryka | **P3LTR** | RP3Beta | P3 | #3-Paths |
|---|---|---|---|---|
| P3LTR | 100% | 70% | 64% | 50% |
| RP3Beta | **70%** | 100% | 84% | 68% |
| P3 | **64%** | 84% | 100% | 60% |
| #3-Paths | **50%** | 68% | 60% | 100% |

Źródło: publikacja autora [79]

podstawie liczby ścieżek o długości 3 łączących danego użytkownika i przedmiot, więc najpopularniejsze przedmioty mogą być znacznie częściej rekomendowane, co z kolei wpływa na niskie wartości metryk.

### 6.3.5. Podobieństwo między rekomendacjami pochodzącymi z różnych modeli

W celu podjęcia decyzji o wdrożeniu nowego modelu rekomendacji raportujemy także wartość współczynnika nakładania się zdefiniowanego w podrozdziale 4.4.3. W tabeli 6.4 przedstawiamy wartości tego współczynnika dla wszystkich par modeli. Zauważmy, że 70% rekomendacji wygenerowanych przez model RP3Beta, zostało również wygenerowanych przez model P3LTR. Wydaje nam się zatem, że zastąpienie modelu RP3Beta modelem P3LTR nie byłoby radykalną, ale zauważalną zmianą dla użytkowników serwisu.

Zauważmy też, że modele RP3Beta oraz P3 zwracają dość podobne rekomendacje (współczynnik nakładania się wynosi dla nich 84%). Zatem wprowadzenie parametrów $\alpha$ i $\beta$ w celu uogólnienia modelu P3 do modelu RP3Beta powoduje niewielkie zmiany jakości modelu w przypadku analizowanego zbioru danych.

**Tabela 6.5.** Wartości wybranych parametrów modelu P3LTR

| Parametr | $k = 1$ | $k = 2$ | $k = 3$ |
|---|---|---|---|
| $d^{(k)}$ | 0,631 | 0,163 | 0,433 |
| $r^{(k)}$ | 0,081 | 0,008 | 0,028 |

Źródło: publikacja autora [79]

### 6.3.6. Parametry modelu P3LTR

Jak wspomnieliśmy, parametry modelu P3LTR można łatwo zinterpretować. W tabeli 6.5 przedstawiamy wartości parametrów $d^{(k)}$ (związanych ze stopniami wierzchołków) oraz $r^{(k)}$ (związanych z czasem wystąpienia interakcji).

W poprzednich pracach [31, 98] dotyczących modelu RP3Beta zwykle wybierano dodatnie wartości parametrów $\alpha$ i $\beta$, aby zmniejszyć wpływ najbardziej popularnych przedmiotów na generowane rekomendacje. Okazuje się, że w modelu P3LTR parametry $d^{(1)}, d^{(2)}, d^{(3)}$, pełniące podobną do nich rolę, uzyskały wskutek procesu trenowania dodatnie wartości. Zatem w tym przypadku, założenia przyjęte we wspomnianych pracach wydają się słuszne.

Zauważmy, że model P3LTR redukuje się do modelu RP3Beta dla $d^{(1)} = d^{(2)} = \alpha$ oraz ustalając odpowiednie wartości dla pozostałych parametrów. W tabeli 6.5 widzimy jednak, że wartości obrane dla parametrów $d^{(1)}$ oraz $d^{(2)}$ mocno różnią się od siebie. Może być to zatem źródło obserwowanych różnic wartości metryk ewaluacji pomiędzy modelami RP3Beta oraz P3LTR.

Zauważmy także, że parametry $r^{(1)}, r^{(2)}, r^{(3)}$ przyjęły dodatnie wartości. Oznacza to, że według modelu, nowsze interakcje użytkowników powinny mieć większy wpływ na generowane rekomendacje.

Nie omawiamy parametrów związanych z typem oraz krotnością interakcji, ponieważ ich wartości nie ustabilizowały się podczas wybranej przez nas liczby iteracji. Prawdopodobnie moglibyśmy uzyskać zbieżność tych parametrów przyjmując wyższe wartości hiperparametru *iterations* określającego liczbę iteracji oraz hiperparametru *batch_size* określającego rozmiar partii. Niestety mogłoby to znacznie wydłużyć czas trenowania modelu.

## 6.4. Podsumowanie

W rozdziale wprowadziliśmy nową, grafową metodę rekomendacji, P3 Learning to Rank (P3LTR). Pokazaliśmy, że metoda ta przy odpowiednim doborze parametrów, redukuje się do przedstawionego wcześniej modelu RP3Beta. Zaproponowana metoda, w przeciwieństwie do modelu RP3Beta, posiada parametry optymalizowane podczas trenowania modelu. Parametry te pozwalają na uwzględnienie cech interakcji, a także cech użytkowników i przedmiotów.

Przedstawiliśmy metodę trenowania modelu P3LTR oraz zdefiniowaliśmy odpowiednie do tego celu funkcje straty. Wymieniliśmy również liczne zalety proponowanego podejścia, między innymi wyjaśnialność rekomendacji, wydajność ich generowania, a także możliwość uzyskania rekomendacji w czasie rzeczywistym poprzez zastosowanie proponowanej wcześniej infrastruktury.

Przedstawiliśmy szczególny przypadek tej metody, w którym uwzględniane są cechy interakcji dostępne w opublikowanym przez nas zbiorze danych. Pokazaliśmy przewagę modelu P3LTR nad modelem RP3Beta pod względem metryk dokładności oraz pokrycia. Przedstawiliśmy także współczynniki nakładania się pomiędzy rozważanymi modelami. Model P3LTR jest zatem jedną z metod wspomnianych w celu głównym rozprawy: *przedstawienie nowych metod rekomendacji opracowanych dla serwisów ogłoszeniowych oraz wykazanie ich przewagi względem metod opisanych w literaturze*.

Proponowana metoda może poprawić jakość rekomendacji generowanych obecnie przy użyciu modelu RP3Beta w serwisach Pracodawcy. Wdrożenie jej zostało jednak wstrzymane kosztem prowadzenia badań nad opracowaniem metody wykorzystującej grafowe sieci neuronowe. Wyniki tych badań przedstawione są w kolejnym rozdziale.

Rozdział 7

# Grafowe sieci neuronowe

W rozdziale przedstawiamy wyniki badań dotyczących wykorzystania metod grafowych sieci neuronowych w celu rekomendacji ofert pracy w serwisach Pracodawcy. W szczególności:

- przedstawiamy wyniki ewaluacji istniejących metod rekomendacji opartych o grafowe sieci neuronowe,
- proponujemy i ewaluujemy skuteczność nowego modelu rekomendacji P3GNN wykorzystującego grafowe sieci neuronowe,
- porównujemy skuteczność popularnych funkcji straty,
- proponujemy i ewaluujemy skuteczność nowych funkcji straty,
- analizujemy wpływ wprowadzenia trudniejszych przykładów negatywnych podczas uczenia modelu.

Poniżej przedstawiamy powody rozpoczęcia badań nad grafowymi sieciami neuronowymi.

- **Skuteczność modeli grafowych** - wyniki uzyskane w rozdziałach 4, 5 oraz 6 pokazują skuteczność metod grafowych w celu rekomendowania ofert pracy w serwisach Pracodawcy.
- **Uzyskanie reprezentacji użytkowników i przedmiotów** - wcześniej przedstawione modele RP3Beta oraz P3LTR, w przeciwieństwie do przedstawianych w tym rozdziale metod, generują rekomendacje bez konieczności znajdowania reprezentacji użytkowników i przedmiotów we wspólnej, wysokowymiarowej przestrzeni. Takie reprezentacje mogłyby zostać wykorzystane w wielu innych zadaniach. Przykładowo umożliwiłyby one wydajne przesortowanie wyników wyszukiwarki, gdzie użytkownicy często wybierają jedynie kategorię lub lokalizację interesujących ich ofert pracy. Mogłyby one zostać także wykorzystane w celu znajdowania najbardziej dopasowanych użytkowników do zadanego ogłoszenia (na przykład kandydatów do zadanej oferty pracy), bez konieczności trenowania odrębnego modelu dla tego zadania.

- **Wykorzystanie cech użytkowników i przedmiotów** - jak wspomnieliśmy, rekomendacje dostarczane użytkownikom serwisów Pracodawcy pochodzą z dwóch modeli: modelu wspólnej filtracji (RP3Beta real-time) oraz modelu opartego wyłącznie o podobieństwo między profilem użytkownika a ogłoszeniem (Elasticsearch). Sądzimy, że model wykorzystujący jednocześnie informacje o interakcjach, jak i cechach użytkowników i przedmiotów może poprawić jakość rekomendacji. Liczba profili użytkowników znacznie wzrosła od momentu rozpoczęcia badań opisywanych w rozprawie. W związku z tym wzrósł także potencjalny wpływ ich właściwego wykorzystania w modelach rekomendacyjnych. Wiele podejść opartych o grafowe sieci neuronowe umożliwia wykorzystanie cech użytkowników i przedmiotów. Zauważmy ponadto, że w przypadku wielu cech użytkownika, w szczególności wyrażanych w profilu użytkownika preferencji (np. oczekiwanej płacy, preferowanej lokalizacji czy rodzaju umowy o pracę), możliwe jest wskazanie odpowiadających im cech przedmiotów. Wartym rozważenia jest zatem reprezentowanie tych cech za pomocą wierzchołków. Dzięki temu obiecującym kierunkiem przyszłych badań jest wykorzystanie metod opartych o grafy wiedzy (ang. *Knowledge Graphs* [56]).
- **Bezpośrednie przekazywanie informacji sąsiednim wierzchołkom** - w grafowych sieciach neuronowych informacje są bezpośrednio przekazywane do sąsiednich wierzchołków w procesie generowania rekomendacji. Wiele klasycznych modeli (np. faktoryzacji macierzy [64, 74]) wykorzystuje informacje na temat interakcji pomiędzy użytkownikami a przedmiotami jedynie poprzez funkcje straty w procesie treningu. Ponadto, nie wykorzystują one informacji o relacji łączącej wierzchołki znajdujące się w odległości większej niż jeden na grafie dwudzielnym reprezentującym użytkowników i przedmioty (ang. *high-order connectivity* [46]). Pokazano, że wykorzystanie tych relacji przez grafowe sieci neuronowe pozwala uzyskać lepsze wyniki [62, 126, 133].
- **Dostępność bibliotek dedykowanych grafowym sieciom neuronowym** - implementację nowych metod wykorzystujących grafowe sieci neuronowe ułatwiają dedykowane w tym celu biblioteki takie jak PyTorch Geometric (PyG) [42], Deep Graph Library (DGL) [122], Graph-Learn (wcześniej AliGraph) [137] czy TensorFlow GNN [41]. Wyniki prezentowane w rozdziale uzyskane zostały za pomocą biblioteki DGL, którą wybraliśmy ze względu na jej popularność oraz bogatą dokumentację.
- **Popularność metod** - w ostatnich latach opublikowane zostało wiele prac pokazujących wysoką jakość modeli rekomendacyjnych opartych na grafowych sieciach neuronowych. Prace te zostały sklasyfikowane w licznych pracach przeglądowych [46, 123, 133]. Raportowane są także skuteczne wdrożenia takich metod w przedsiębiorstwach [135, 137].

Podjęliśmy próby wykorzystania cech przedmiotów w kilku wybranych modelach opartych o grafowe sieci neuronowe. Niestety uzyskaliśmy istotnie gorsze wyniki względem metryki Precision@10 niż w przypadku grafowych modeli neuronowych wspólnej

filtracji. Z tego powodu postanowiliśmy najpierw skoncentrować się na znalezieniu modelu wspólnej filtracji dającego co najmniej tak dobre wyniki jak model RP3Beta, a następnie wzbogacić model o cechy użytkowników i przedmiotów. W rozdziale szczegółowo przedstawimy jedynie wyniki uzyskane przez modele wspólnej filtracji, co jest spójne z tematyką rozprawy.

## 7.1. Rekomendacje na grafach

Interakcje użytkowników z przedmiotami pełnią kluczową rolę w budowie modeli rekomendacyjnych. Reprezentacja interakcji za pomocą grafu umożliwia eksplorację licznych zależności występujących między poszczególnymi użytkownikami i przedmiotami. Z tego powodu opracowano wiele metod rekomendacji bazujących na grafach [36, 123].

Omawiany przez nas wcześniej model RP3Beta został początkowo przedstawiony [98] jako metoda bazująca na spacerach losowych [123]. W uproszczeniu, w metodach tych obliczane jest prawdopodobieństwo przejścia z danego wierzchołka na wierzchołki sąsiednie. Następnie symulowane są spacery losowe, najczęściej rozpoczynające się od wierzchołka reprezentującego danego użytkownika. Rekomendacje generowane są na podstawie analizy wierzchołków, w których spacery te się kończą (np. rekomendowane są przedmioty reprezentujące wierzchołki, na których zakończyła się największa liczba spacerów). Istotnym dla nas ograniczeniem tej grupy metod jest brak możliwości uzyskania reprezentacji wektorowej użytkowników i przedmiotów.

Istnieje jednak wiele metod grafowych pozwalających na generowanie reprezentacji wektorowych użytkowników i przedmiotów [123]. Przykładowo modele Node2vec [55] oraz DeepWalk [101] generują spacery losowe jedynie w celu uzyskania sekwencji kolejnych przedmiotów, które stanowią dane treningowe dla modelu Skip-gram [92]. W ostatnich latach popularność zyskały grafowe sieci neuronowe [46, 123, 133]. Podejścia te bazują na agregowaniu informacji pochodzących z sąsiednich wierzchołków. Powtórzenie tej operacji $K$-krotnie (w $K$ warstwach) pozwala na przepływ informacji pomiędzy wierzchołkami znajdującymi się w odległości co najwyżej $K$. Idea ta została wcześniej wykorzystana w konwolucyjnych sieciach neuronowych (ang. *Convolutional Neural Networks* [86]), których skuteczność przyczyniła się do uogólnienia stosowanych tam mechanizmów dla grafów [46].

### 7.1.1. Typy grafów

Przed rozpoczęciem tworzenia grafowej sieci neuronowej konieczne jest przedstawienie danych w postaci grafu. Często można to uczynić na wiele sposobów. Poniżej przedstawiamy cechy grafu, które mają istotny wpływ na działanie modelu rekomendacji w przypadku modeli wspólnej filtracji.

Zdaniem Wu i in. [133] wierzchołki najczęściej reprezentują użytkowników oraz przedmioty. Podejście takie wykorzystywane jest przez 9 z 10 klasyfikowanych pod tym względem metod przez Gao i in. [46]. Jedno z podejść, PinSage [135], reprezentuje jako wierzchołki jedynie przedmioty. W naszych badaniach **wierzchołki reprezentują użytkowników i przedmioty** ze względu na większą popularność takich metod, a także plany wykorzystania cech użytkowników w przyszłych badaniach. Rozważany przez nas graf jest zatem **grafem heterogenicznym** (ang. *heterogeneous graphs*) [133] czyli grafem, w którym istnieje więcej niż jeden typ wierzchołków lub krawędzi.

Istotne jest także określenie czy informacje pomiędzy połączonymi wierzchołkami zawsze mogą przepływać w dwóch kierunkach. Przykładowo aplikacja użytkownika na ofertę pracy może być oznaczona poprzez krawędź łączącą wierzchołek $u$ reprezentujący tego użytkownika z wierzchołkiem $v$ reprezentującym tę ofertę pracy. W naszych badaniach **interakcja użytkownika automatycznie generuje także krawędź przeciwną** (od wierzchołka $v$ do wierzchołka $u$). Alternatywnie, krawędź od wierzchołka $v$ do wierzchołka $u$ mogłaby reprezentować zainteresowanie pracodawcy danym kandydatem (jeśli dane takie byłyby dostępne).

Kolejną decyzją jest rozróżnianie typów krawędzi. Przyjęcie większej liczby typów zwykle zwiększa złożoność modelu, lecz może prowadzić do lepszych wyników (o czym wspominaliśmy w rozdziale 6). Model RP3Beta nie rozróżnia jednak typów interakcji, a początkowym celem prowadzonych przez autora badań nad sieciami grafowymi jest uzyskanie modelu dającego wyniki przynajmniej tak dobre jak model RP3Beta. Uznaliśmy zatem, że cel ten powinien zostać osiągnięty **bez rozróżniania typów interakcji**, co upraszcza konstrukcję modelu.

Każda warstwa grafowych sieci neuronowych wykorzystuje reprezentacje wierzchołków w celu uzyskania nowych reprezentacji. W przypadku modeli wspólnej filtracji, popularnym podejściem jest losowa inicjalizacji początkowych reprezentacji wierzchołków i traktowanie ich jako parametry optymalizowane w procesie uczenia [62, 126, 128]. W szczególności, istnieją modele, które nie posiadają żadnych parametrów poza nimi (np. LightGCN [62]). Traktowanie początkowych reprezentacji wierzchołków jako parametrów nie jest jednak koniecznością, nawet w przypadku modeli wspólnej filtracji. Przykładowo możemy uzyskać początkowe reprezentacje za pomocą innego modelu (np. modelu ALS zdefiniowanego w podrozdziale 4.1.1). Potencjalnie najlepsze wyniki mogłoby dać trenowanie modelu od początku do końca (ang. *end-to-end*), lecz w celu zwiększenia wydajności, traktowanie reprezentacji uzyskanych z modelu bazowego jako początkowych

cech wierzchołków również może pozwolić na uzyskanie lepszych wyników od modelu bazowego. W przedstawionych w rozprawie badaniach, przyjmujemy, że **początkowe reprezentacje wierzchołków są parametrami optymalizowanymi w trakcie procesu uczenia**.

## 7.2. Grafowe sieci neuronowe

Grafowe sieci neuronowe często definiowane są za pomocą **paradygmatu przekazywania wiadomości** (ang. *message passing paradigm*) [128, 133] i taką definicję przedstawiamy w pracy. Czasami przyjmuje się, że pojęcie grafowej sieci neuronowej obejmuje szerszą klasę modeli, lecz wiele z tych metod może zostać przedstawionych za pomocą paradygmatu przekazywania wiadomości [122], czasem jednak muszą one być zastosowane do odpowiednio zmodyfikowanego grafu [118].

Przyjmijmy następujące oznaczenia:

- $\mathcal{E}$ : zbiór trójek $(u, v, e)$, gdzie $e$ jest krawędzią łączącą wierzchołek $u$ z wierzchołkiem $v$,
- $\mathbf{x}_v^{(l)}$ : reprezentacja wektorowa wierzchołka $v$ w $l$-tej warstwie sieci,
- $\mathbf{z}_e^{(l)}$ : reprezentacja wektorowa krawędzi $e$ w $l$-tej warstwie sieci.

Paradygmatem przekazywania wiadomości nazywamy następujące operacje pozwalające na uzyskanie reprezentacji wierzchołków w $(l+1)$-szej warstwie na podstawie reprezentacji wierzchołków i krawędzi w warstwie $l$-tej [122]:

$$\mathbf{m}_e^{(l+1)} = \phi\left(\mathbf{x}_v^{(l)}, \mathbf{x}_u^{(l)}, \mathbf{z}_e^{(l)}\right), (u, v, e) \in \mathcal{E}, \tag{7.1}$$

$$\mathbf{x}_v^{(l+1)} = \psi\left(\mathbf{x}_v^{(l)}, \rho\left(\left\{\mathbf{m}_e^{(l+1)} : (u, v, e) \in \mathcal{E}\right\}\right)\right), \tag{7.2}$$

gdzie: $\phi$ nazywamy funkcją **wiadomości**, $\psi$ funkcją **aktualizacji**, zaś $\rho$ funkcją **redukcji**.

W równaniu (7.1) dla każdej krawędzi $(u, v, e)$ obliczana jest wartość $\mathbf{m}_e^{(l+1)}$, którą nazywać będziemy wiadomością przekazywaną z wierzchołka $u$ do wierzchołka $v$. Następnie w równaniu (7.2) dla każdego wierzchołka $v$ wszystkie przekazane do niego wiadomości są agregowane za pomocą funkcji redukcji $\rho$, która pozwala na uzyskanie wektora ustalonej długości reprezentującego wszystkie te wiadomości. Przykładowe funkcje redukcji to średnia, suma, maksimum, minimum oraz LSTM [58]. Finalnie, funkcja aktualizacji $\psi$ pozwala na uwzględnienie poprzedniej reprezentacji wierzchołka $v$ przy ustalaniu jego nowej reprezentacji.

Zauważmy, iż w przedstawionym paradygmacie przekazywania wiadomości, podobnie jak autorzy biblioteki DGL [122], nie opisaliśmy metody uzyskiwania reprezentacji krawędzi w kolejnych warstwach sieci. W analizowanych w tym rozdziale modelach nie wykorzystujemy cech krawędzi. Przedstawiamy jednak model P3LTR z punktu widzenia paradygmatu przekazywania wiadomości, gdzie cechy krawędzi w kolejnych warstwach nie zmieniają się, tzn. $\mathbf{z}_e^{(l)} = \mathbf{z}_e^{(0)}$, dla dowolnej liczby całkowitej $l$.

Powtarzając opisany proces możemy uzyskać reprezentacje wierzchołków odpowiednio w warstwach $1, 2, \ldots, L$, gdzie $L$ to liczba warstw sieci. Zauważmy, że na uzyskaną reprezentację w $L$-tej warstwie wpływ mają wierzchołki znajdujące się w odległości co najwyżej $L$. W procesie agregacji pewne informacje mogą jednak zostać utracone, więc reprezentacje wierzchołków w niższych warstwach również mogą być użyteczne. Z tego powodu finalna reprezentacja wierzchołków zdefiniowana jest wzorem:

$$\mathbf{x}_u = g\left(\mathbf{x}_u^{(0)}, \mathbf{x}_u^{(1)}, \cdots, \mathbf{x}_u^{(L)}\right). \tag{7.3}$$

Funkcja $g$, w zależności od podejścia, definiowana jest jako suma, średnia, konkatenacja, bądź wartość reprezentacji z ostatniej warstwy [128].

W przypadku grafów heterogenicznych wierzchołki bądź krawędzie należące do różnych typów wyrażają inne informacje i mogą być opisane przez różne typy cech. Naturalnym wydaje się zatem, aby istniały parametry funkcji wiadomości, aktualizacji i redukcji odpowiadające każdemu z występujących typów (szczególnie, gdy wymiarowość cech różni się w zależności od typu). Wykorzystując bibliotekę DGL funkcje te definiujemy niezależnie dla każdej trójki typów złożonej z typu wierzchołka początkowego, typu krawędzi oraz typu wierzchołka końcowego. Jeśli do pewnego wierzchołka końcowego prowadzą krawędzie reprezentowane przez różne trójki typów konieczne jest dodatkowo zdefiniowanie metody agregacji w celu uzyskania finalnej reprezentacji wierzchołka końcowego. W grafie rozważanym w naszych badaniach sytuacja ta nie ma miejsca, ponieważ rozważamy jedynie dwie trójki typów: (użytkownik, interakcja, przedmiot) oraz (przedmiot, interakcja, użytkownik). Moglibyśmy rozróżnić rodzaj interakcji (taki jak wyświetlenie przedmiotu, aplikacja na ofertę pracy) poprzez utworzenie większej liczby trójek typów, co jak widzimy skomplikowałoby złożoność sieci. Alternatywnie, moglibyśmy potraktować rodzaj interakcji jako cechę krawędzi. Jak wspomnieliśmy jednak, na tym etapie badań zdecydowaliśmy się pominąć rozróżnianie rodzajów interakcji.

### 7.2.1. Grafowe sieci neuronowe a model P3LTR

W podrozdziale 6.1.3 przestawiliśmy metodę trenowania modelu P3LTR. Wspomnieliśmy, że wykorzystuje ona przedstawioną ideę przekazywania wiadomości opisaną równaniami (7.1) oraz (7.2). Istnieje jednak zasadnicza różnica między modelem P3LTR a klasycznymi grafowymi sieciami neuronowymi. W przypadku modelu P3LTR przekazywanie wiadomości wykonywane jest dla każdego użytkownika z osobna za pomocą opisanej poniżej procedury.

1. Wybieramy użytkownika, dla którego chcemy obliczyć rekomendacje. Niech $v_0$ oznacza wierzchołek grafu reprezentujący tego użytkownika.
2. Przyjmujemy $\mathbf{x}_{v_0}^{(0)} = \text{concat}(1, f_{v_0}^n)$ oraz $\mathbf{x}_u^{(0)} = \text{concat}(0, f_u^n)$ dla wierzchołków $u \neq v_0$, zaś $\mathbf{z}_e^{(l)} = f_{vu}^e$ dla $l = 0, 1, 2$, gdzie $f_x^n$ jest wektorem cech wierzchołka $x$, zaś $f_{xy}^e$ jest wektorem cech krawędzi łączącej wierzchołki $x$ oraz $y$ (jak w równaniu (6.5)).

3. Równania (7.1) oraz (7.2) przybierają wtedy postać:

$$\mathbf{m}_e^{(l+1)} = \mathbf{x}_u^{(l)}[0]\phi^{(l+1)}\left(\mathbf{x}_v^{(l)}[1:], \mathbf{x}_u^{(l)}[1:], \mathbf{z}_e^{(l)}\right), \text{ dla } (u,v,e) \in \mathcal{E},$$

$$\mathbf{x}_v^{(l+1)} = \text{concat}\left(\sum_{(u,v,e)\in\mathcal{E}} \mathbf{m}_e^{(l+1)}, \mathbf{x}_v^{(l)}[1:]\right),$$

   gdzie $\phi^{(l+1)}$ jest koderem cech (jak w równaniu (6.5)), $\mathbf{x}[0]$ oznaczą pierwszą współrzędną wektora $\mathbf{x}$, zaś $\mathbf{x}[1:]$ oznacza wektor złożony ze wszystkich poza pierwszą współrzędnych wektora $\mathbf{x}$.
4. Wartość $\mathbf{x}_v^{(3)}[0]$ reprezentuje podobieństwo wierzchołka $v$ względem wierzchołka $v_0$. Rekomendowane są przedmioty, dla których wartości tak określonego podobieństwa są największe.

Zauważmy, że w punkcie drugim tej procedury przypisujemy cechy wszystkim wierzchołkom w celu uzyskania odpowiedniego grafu względem wybranego w punkcie pierwszym użytkownika. Wadą takiego postępowania jest długi czas trenowania modelu P3LTR. Przyjęte w rozdziale 6 kodery cech posiadają niewielką liczbę parametrów. Dzięki temu do trenowania modelu P3LTR, jak raportowaliśmy w tabeli 6.1, wystarczającym było wykonanie opisanej procedury jedynie $\text{iterations} \cdot \text{batch\_size} = 80 * 153 = 12240$ razy. Niestety w przypadku koderów cech posiadających większą liczbę parametrów konieczne mogłoby się okazać wykonanie tej procedury znacznie większą liczbę razy, co byłoby czasochłonne.

Zauważmy również, że w modelu P3LTR nie uzyskujemy użytecznej reprezentacji wektorowej wierzchołków. Wszystkie poza pierwszą współrzędną wektora $\mathbf{x}_v^{(3)}$ reprezentują jedynie pierwotne cechy tego wierzchołka. Pierwsza współrzędna reprezentuje podobieństwo wierzchołka $v$ do zadanego wierzchołka $v_0$, a więc wartość tej reprezentacji zależna jest od $v_0$. Reprezentacja wierzchołka $v$ jako wektora podobieństw do wszystkich możliwych wierzchołków $v_0$ prawdopodobnie posiadałaby zbyt duży wymiar by użyć jej w praktycznych zastosowaniach. Możliwe byłoby wykorzystanie technik redukcji wymiarowości, bądź zbudowanie reprezentacji za pomocą podobieństw do wierzchołków z pewnego szczególnego podzbioru. Takie operacje wymagałyby jednak przeprowadzenia dodatkowych badań i zwiększałyby dodatkowo złożoność modelu.

Zaletą modelu P3LTR jest jednak uzyskanie uogólnienia modelu RP3Beta dającego lepsze wyniki od modelu RP3Beta. W podrozdziale 7.3.5 przedstawimy inne uogólnienie modelu RP3Beta, w którym nie będzie konieczne konstruowanie grafu dla każdego użytkownika z osobna, a przy ograniczeniu jego wymiarowości, pozwoli ono na uzyskanie reprezentacji wektorowej wierzchołków.

### 7.2.2. Elementy budowy grafowych sieci neuronowych

Paradygmat przekazywania wiadomości opisuje główną ideę grafowych sieci neuronowych. Istnieje jednak wiele innych elementów, które musimy określić w procesie trenowania i generowania rekomendacji. Opisujemy je w tym podrozdziale.

Wang, Zhao i Shi [128] wyszczególnili elementy budowy grafowych sieci neuronowych stosowanych w modelach wspólnej filtracji. Elementy te przedstawiamy poniżej.

- **Wymiar wejściowej reprezentacji wektorowej wierzchołków.**
- **Funkcje wiadomości, redukcji i aktualizacji** - autorzy zapisali równanie (7.2) za pomocą funkcji agregacji i aktywacji zamiast redukcji i aktualizacji. Możemy zatem przyjąć, że istotnymi elementami są funkcje wiadomości, redukcji i aktualizacji.
- **Liczba warstw.**
- **Funkcja agregacji reprezentacji z różnych warstw**, którą opisuje równanie (7.3).
- **Liczba komponentów** - możliwe jest trenowanie różnych grafowych sieci neuronowych w celu uzyskania różnych reprezentacji wierzchołków.
- **Funkcja agregacji reprezentacji z różnych komponentów.**
- **Funkcja interakcji** - odpowiada za predykcję wartości dla zadanej pary wierzchołków na podstawie ich reprezentacji wektorowych. W celu generowania rekomendacji dla zadanego użytkownika wybierane są te przedmioty, dla których funkcja ta osiąga najwyższe wartości. Częstym wyborem funkcji interakcji jest iloczyn skalarny.

Poniżej wymieniamy także inne elementy, których wyboru dokonaliśmy podczas budowy przedstawianych w tym rozdziale modeli.

Pierwszym z nich jest **funkcja straty**. Autorzy cytowanej wyżej pracy wspomnieli, iż pozostawiają badanie wpływu zastosowania różnych funkcji straty jako potencjalne kierunki przyszłych badań. W rozprawie częściowo adresujemy ten problem przedstawiając skuteczność omawianych modeli dla różnych funkcji strat. Często stosowanymi funkcjami straty są entropia krzyżowa oraz BPR [46].

W wielu praktycznych zastosowaniach, wierzchołki mogą posiadać ogromną liczbę sąsiadów. Jak podaliśmy w tabeli 3.5, ponad 10% interakcji w zbiorze danych OLX Jobs Interactions dotyczy przedmiotów posiadających co najmniej 2000 sąsiadów. W przypadku rekomendacji filmów czy muzyki liczby te mogą osiągać znacznie większe wartości. W konsekwencji wyzwaniem jest utrzymanie odpowiedniej wydajności modelu [133]. Rozwiązaniem tego problemu jest **agregowanie wiadomości przekazywanych przez podzbiór sąsiednich wierzchołków**. Najprostszym przykładem takiego postępowania jest losowanie ustalonej liczby sąsiadów danego wierzchołka. Bardziej zaawansowane rozwiązanie zastosowane zostało w modelu PinSage [135], gdzie agregowane są wiadomości przekazywane przez wierzchołki, w których kończy się największa liczba spacerów losowych rozpoczynających się w zadanym wierzchołku.

Kolejnym elementem jest **wybór algorytmu losowania negatywnych przykładów**. Uczenie modelu polega na dostosowaniu parametrów w taki sposób, aby krawędzie pozytywne uzyskiwały wyższe wartości funkcji interakcji od przykładów negatywnych. W zbiorach danych z niejawnymi ocenami posiadamy najczęściej jedynie pozytywne przykłady. Często przyjmuje się, że negatywnymi krawędziami dla danego użytkownika jest pewna liczba losowo wygenerowanych krawędzi łączących tego użytkownika z przedmiotami, z którymi nie wszedł on w interakcje [46]. Losowo wygenerowane przykłady negatywne mogą jednak być na tyle różne od przykładów pozytywnych, że model szybko nauczy się trafnie je klasyfikować, mimo braku zdolności do trafnego klasyfikowania przykładów trudniejszych. W celu zaadresowania tego problemu stosowane są bardziej zaawansowane metody losowania przykładów negatywnych [25, 135]. W dalszej części rozdziału przedstawimy skuteczność zaimplementowanej przez nas metody losowania przykładów negatywnych opartej o spacery losowe.

Jak wspomina Chen i in. [46] wykonanie obliczeń na całym grafie podczas wykonywania kroku w przód w procesie uczenia nie jest możliwe w praktycznych zastosowaniach, gdzie zbiór treningowy liczy miliony wierzchołków i krawędzi. Z tego powodu krok w przód wykonywany jest na **partiach danych**, a po każdej partii wykonywana jest propagacja wsteczna błędu. W obliczaniu reprezentacji danego wierzchołka w $L$-warstwowej sieci grafowej biorą udział wierzchołki znajdujące się w odległości $L$ od niego. Zatem w celu obliczenia finalnej reprezentacji zadanej liczby wierzchołków wymagana jest zwykle wielokrotnie większa liczba wierzchołków. Szczegóły dotyczące metody konstruowania takich podgrafów znajdują się w pracy przedstawiającej model PinSage [135]. Podgrafy te mogą być konstruowane za pomocą biblioteki DGL, gdzie określić należy rozmiar partii, a także metody generowania przykładów pozytywnych i negatywnych. Podczas badań autor zauważył, że w generowanych podgrafach nie powinny się znajdować zarówno krawędzie negatywne jak i pozytywne. W przypadku braku jedynie krawędzi negatywnych, model mógłby błędnie nauczyć się przyjmowania takich reprezentacji wierzchołków, w których wartość funkcja interakcji osiągałaby duże wartości jedynie dla par wierzchołków sąsiadujących ze sobą. Autor znalazł błąd w metodzie wykluczania obserwacji pozytywnych z podgrafów zaimplementowanej w bibliotece DGL, wskutek czego autorzy biblioteki naprawili go[1]. Autor sprawdził jednak, iż wykluczenie wierzchołków pozytywnych nie wpłynęło istotnie na jakość ewaluowanych modeli.

Przedstawione elementy budowy możemy traktować jako hiperparametry modelu. Duża ich liczba stanowi wyzwanie, szczególnie w przypadku dużych zbiorów danych. W naszych badaniach, wiele z tych elementów dobraliśmy w procesie optymalizacji hiperparametrów, co szczegółowo opiszemy w dalszej części rozdziału.

[1] `https://github.com/dmlc/dgl/pull/5532`, dostęp: 2023-11-18

## 7.3. Porównywane metody

W podrozdziale przedstawiamy porównywane przez nas modele wspólnej filtracji oparte na grafowych sieciach neuronowych. Wszystkie one stosowane są dla grafu dwudzielnego reprezentującego użytkowników i przedmioty. Początkowe reprezentacje wierzchołków stanowią parametry optymalizowane podczas trenowania modeli. Jako funkcję interakcji przyjęliśmy iloczyn skalarny.

### 7.3.1. LightGCN

Model LightGCN [62] jest prostym modelem nieposiadającym parametrów. Zatem w procesie trenowania optymalizowane są jedynie parametry pochodzące z początkowych reprezentacji wierzchołków. W modelu tym przesyłanie informacji określone jest równaniem:

$$\mathbf{x}_v^{(l+1)} = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(v)|}} \mathbf{x}_u^{(l)},$$

gdzie $N(x)$ oznacza zbiór wierzchołków sąsiadujących z wierzchołkiem $x$.

Finalna reprezentacja uzyskiwana jest jako średnia reprezentacji wszystkich warstw, tzn.

$$\mathbf{x}_v = \frac{1}{L+1} \sum_{l=0}^{L} \mathbf{x}_v^{(l)}.$$

Zdaniem autorów tego podejścia, powszechnie stosowane w grafowych sieciach neuronowych nieliniowe funkcje aktywacji oraz przekształcanie macierzy cech, nie poprawiają wyników w przypadku modeli wspólnej filtracji. Przeprowadzili oni eksperymenty pokazujące, że usunięcie tych elementów z powszechnie stosowanego modelu NGCF [126] poprawia jego wyniki. Model NGCF uzyskuje słabsze wyniki od modelu LightGCN zarówno w badaniach autorów, jak i badaniach innych badaczy [73, 89]. Z tego powodu zdecydowaliśmy się nie umieszczać modelu NGCF w porównaniu.

### 7.3.2. Average

Autor zainspirowany prostotą i skutecznością modelu LightGCN zaimplementował uproszczoną wersję modelu, którą oznaczać będziemy przez Average (rozważaną także w eksperymentach autorów modelu LightGCN [62]). Przekazywanie wiadomości w tym modelu zadane jest równaniem:

$$\mathbf{x}_v^{(l+1)} = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{x}_u^{(l)}.$$

Jako finalne reprezentacje wierzchołków przyjmujemy ich reprezentacje z ostatniej warstwy lub, podobnie jak w modelu LightGCN, średnią reprezentacji ze wszystkich warstw (traktujemy ten wybór jako hiperparametr modelu).

### 7.3.3. GraphSAGE

Model GraphSAGE [58] (Graph SAmple and aggreGatE) to pierwsza metoda wykorzystująca jedynie podzbiór wierzchołków w procesie agregacji informacji z sąsiednich wierzchołków [46]. Proces przekazywania wiadomości opisany jest równaniami:

$$\begin{aligned}\mathbf{m}_e^{(l+1)} &= \text{aggregate}^{(l)}\left(\left\{\mathbf{x}_u^{(l)} : (u,v,e) \in \mathcal{E}\right\}\right), \\ \mathbf{x}_v^{(l+1)} &= \sigma\left(\boldsymbol{W}^{(l)} \cdot \text{concat}\left(\mathbf{x}_v^{(l)}, \mathbf{m}_e^{(l+1)}\right)\right).\end{aligned},$$

gdzie $\sigma$ oznacza funkcję aktywacji, $\boldsymbol{W}^{(l)}$ jest macierzą parametrów, zaś $\text{aggregate}^{(l)}$ funkcją agregującą. Funkcję agregującą traktujemy jako hiperparametr modelu, który może przyjąć następujące wartości: *mean*, *gcn* oraz *pool*. Testowaliśmy również proponowaną przez autorów wartość *lstm*, lecz we wstępnych testach nie uzyskaliśmy istotnie lepszych rezultatów, a czas trenowania modelu był wielokrotnie dłuższy niż przy użyciu innych metod. Funkcję aktywacji $\sigma$ również traktujemy jako hiperparametr i rozważamy następujące możliwości: funkcja sigmoidalna ($\sigma(x) = \frac{1}{1+e^{-x}}$), funkcja ReLU ($\sigma(x) = \max(0,x)$) oraz funkcja identycznościowa ($\sigma(x) = x$; pominięcie aktywacji).

Ponadto, w naszych badaniach rozważamy jako hiperparametr możliwość i sposób normalizacji wartości reprezentacji wierzchołków w każdej warstwie. Poza zaproponowaną przez autorów normalizacją l2, rozważamy normalizację l1 oraz pominięcie normalizacji.

Model GraphSAGE był inspiracją do stworzenia wspomnianego wcześniej modelu PinSAGE [135] - modelu rekomendacyjnego wytrenowanego na grafie z 3 miliardami wierzchołków i 18 miliardami krawędzi.

### 7.3.4. GAT

W modelu GAT [117] (Graph Attention Network) wpływ wierzchołków przekazujących informacje zależy od ich podobieństwa do wierzchołka docelowego. Podobieństwo to jest wyliczane na podstawie cech tych wierzchołków oraz optymalizowanych podczas treningu parametrów. Proces przekazywania wiadomości opisany jest równaniami:

$$\begin{aligned}\mathbf{m}_e^{(l+1)} &= \alpha_{vu}^{(l)} \boldsymbol{W}^{(l)} \mathbf{x}_u^{(l)} : (u,v,e) \in \mathcal{E}, \\ \alpha_{vu}^{(l)} &= \frac{\exp\left(\text{LeakyReLU}\left((\mathbf{a}^{(l)})^\top \left[\boldsymbol{W}^{(l)}\mathbf{x}_v^{(l)} \| \boldsymbol{W}^{(l)}\mathbf{x}_u^{(l)}\right]\right)\right)}{\sum_{s\in\mathcal{N}(v)} \exp\left(\text{LeakyReLU}\left((\mathbf{a}^{(l)})^\top \left[\boldsymbol{W}^{(l)}\mathbf{x}_v^{(l)} \| \boldsymbol{W}^{(l)}\mathbf{x}_s^{(l)}\right]\right)\right)}, \\ \text{LeakyReLU}(x) &= \begin{cases} x, & \text{gdy } x \geq 0 \\ px, & \text{gdy } x < 0 \end{cases}, \\ \mathbf{x}_v^{(l+1)} &= \sum_{(u,v,e)\in\mathcal{E}} \mathbf{m}_e^{(l+1)},\end{aligned}$$

gdzie $\boldsymbol{W}^{(l)}$ jest macierzą wykorzystywaną do transformacji cech wierzchołków, $\mathbf{a}^{(l)}$ jest wektorem parametrów, $\|$ oznacza konkatenację, zaś wartość $p$ oznacza współczynnik nachylenia funkcji LeakyReLU. Autorzy sugerowali przyjęcie $p = 0{,}2$ jednak w naszych badaniach traktujemy tę liczbę jako hiperparametr, którego wartość będziemy optymalizować. Mówimy, że model ten wykorzystuje mechanizm atencji, ponieważ wpływ poszczególnych wierzchołków na wartości reprezentacji zadanego wierzchołka w kolejnej warstwie wyznaczany jest w procesie uczenia.

Autorzy modelu GAT sugerują, iż korzystne może być wykorzystanie mechanizmu atencji $K$-krotnie przy użyciu niezależnych zestawów parametrów (ang. *multi-head attention*). Niech $\mathbf{x}_{v,k}^{(l+1)}$ oznacza reprezentację wierzchołka $v$ w warstwie $(l+1)$-szej uzyskaną za pomocą $k$-tego mechanizmu atencji, $1 \leq k \leq K$. Wtedy reprezentację wierzchołka w warstwie kolejnej uzyskujemy za pomocą równania:

$$\sigma\left(\frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_{v,k}^{(l+1)}\right),$$

gdzie $\sigma$ oznacza funkcję aktywacji. Autorzy rozważali także uzyskanie reprezentacji za pomocą konkatenacji reprezentacji wierzchołków pochodzących z różnych mechanizmów atencji. Taka operacja może jednak istotnie zwiększyć wymiarowość reprezentacji, dlatego w naszych badaniach zdecydowaliśmy się jej nie rozważać.

Autorzy wspominają, iż $\mathcal{N}(v)$ oznacza *pewne sąsiedztwo* wierzchołka $v$, co niekoniecznie oznaczać musi zbiór jego bezpośrednich sąsiadów (tzn. wierzchołków bezpośrednio połączonych z zadanym wierzchołkiem $v$). W swoich eksperymentach autorzy przyjęli, iż $\mathcal{N}(v)$ oznacza zbiór bezpośrednich sąsiadów wierzchołka $v$ wraz z wierzchołkiem $v$. W naszym przypadku jednak nie wliczamy wierzchołka $v$ do zbioru $\mathcal{N}(v)$ ze względu na heterogoniczną strukturę naszego grafu. Jak wspomnieliśmy w sekcji 7.2, w przypadku grafów heterogenicznych, przekazywanie wiadomości odbywa się osobno dla każdej trójki złożonej z typu wierzchołka początkowego, typu krawędzi i typu wierzchołka końcowego, tzn. trójki (użytkownik, interakcja, przedmiot) oraz trójki (przedmiot, interakcja, użytkownik). Przy różnej liczbie wymiarów cech użytkowników i cech przedmiotów nie jest możliwe zastosowanie tej samej macierzy $\boldsymbol{W}$ w celu transformacji cech wierzchołków reprezentujących użytkowników i przedmioty. Ponadto, nawet przy równej liczbie cech wierzchołków, cechy wierzchołków różnych typów mogą reprezentować odmienne informacje, przez co wykonanie transformacji przez tę samą macierz $\boldsymbol{W}$ mogłoby być mało skuteczne.

### 7.3.5. P3GNN

W podrozdziale proponujemy uogólnienie modelu RP3Beta, w którym uzyskamy reprezentacje wektorowe użytkowników i przedmiotów. W przedstawionych poprzednio modelach definiowaliśmy proces przekazywania wiadomości w pojedynczej warstwie

**Tabela 7.1.** Metody przekazywania wiadomości oraz reprezentacja wierzchołków w poszczególnych warstwach modelu P3GNN. Metodę identycznościową oznaczyliśmy przez $i$, zaś metodę średniej w $l$-tej warstwie przez $s_l$. Ponadto oznaczyliśmy początkową macierz reprezentacji przedmiotów przez $p$

| Numer warstwy | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Metoda dla (użytkownik, interakcja, przedmiot) | - | $i$ | $s_2$ | $i$ |
| Metoda dla (przedmiot, interakcja, użytkownik) | - | $s_1$ | $i$ | $s_3$ |
| Reprezentacja użytkowników | - | $s_1(p)$ | $s_1(p)$ | $s_3(s_2(s_1(p)))$ |
| Reprezentacja przedmiotów | $p$ | $p$ | $s_2(s_1(p))$ | $s_2(s_1(p))$ |

sieci, przy czym dopuszczalne było złożenie dowolnej liczby takich warstw. W przypadku modelu P3GNN przyjmujemy istnienie dokładnie trzech warstw. Ponadto, model ten zakłada, iż operujemy na grafie dwudzielnym, gdzie istnieją tylko dwie różne trójki typów złożonych z typu wierzchołka początkowego, typu krawędzi oraz typu wierzchołka końcowego. W naszym przypadku są to trójki: (użytkownik, interakcja, przedmiot) oraz (przedmiot, interakcja, użytkownik). Musimy zatem zdefiniować 6 metod przekazywania wiadomości, dwie dla każdej z trzech warstw.

W naszej sieci wykorzystujemy dwie metody przekazywania wiadomości:

- metodę **identycznościową**, w której reprezentacje wierzchołków są identyczne do ich reprezentacji w warstwie poprzedniej,
- metodę **średniej**, w której przekazywanie informacji definiują równania:

$$\mathbf{m}_e^{(l+1)} = \mathbf{x}_u^{(l)}, (u, v, e) \in \mathcal{E},$$
$$\mathbf{x}_v^{(l+1)} = \frac{1}{|\mathcal{N}(v)|^{d^{(l)}}} \sum_{(u,v,e)\in\mathcal{E}} \mathbf{m}_e^{(l+1)},$$

gdzie $d^{(0)}, d^{(1)}, d^{(2)} \in \mathbb{R}$ są parametrami.

W tabeli 7.1 przedstawiliśmy wybór metod przekazywania wiadomości oraz reprezentacje użytkowników w poszczególnych warstwach w modelu P3GNN. Jako finalną reprezentację wierzchołków przyjmujemy reprezentację początkową dla przedmiotów oraz reprezentację z trzeciej warstwy dla użytkowników.

W modelu P3GNN informacje zawarte w cechach przedmiotów są propagowane w kolejnych warstwach. Finalnie otrzymujemy reprezentacje użytkowników będące sumą ważoną reprezentacji przedmiotów znajdujących się w odległości co najwyżej 3 na grafie dwudzielnym, gdzie waga ustalonej ścieżki $(i, u', i', u)$ wynosi:

$$\frac{1}{|\mathcal{N}(u')|^{d^{(0)}}} \frac{1}{|\mathcal{N}(i')|^{d^{(1)}}} \frac{1}{|\mathcal{N}(u)|^{d^{(2)}}}.$$

Zauważmy, że moglibyśmy zamienić rolę użytkowników i przedmiotów w modelu P3GNN. Takie sformułowanie byłoby bliższe przedstawionemu w podrozdziale 7.2.1 sposobowi przekazywania wiadomości w modelu P3LTR. Zdecydowaliśmy się jednak nie rozważać tego scenariusza, czego powody przedstawiamy poniżej.

- Bardziej powszechne jest występowanie cech przedmiotów niż użytkowników. Jak wspomnieliśmy wcześniej, w części naszych badań wykorzystywaliśmy uzyskane z innego modelu reprezentacje przedmiotów jako początkowe cechy wierzchołków. Powód ten nie ma jednak zastosowania dla modeli wspólnej filtracji, gdzie cechy wierzchołków traktowane są jako parametry optymalizowane podczas treningu.
- Jak pokażemy, model RP3Beta jest szczególnym przypadkiem modelu P3GNN, jeśli przyjmiemy odpowiednie reprezentacje przedmiotów, których wymiar równy jest liczbie przedmiotów. W przypadku odpowiednika modelu P3GNN, w którym rolę użytkowników i przedmiotów są zamienione, uzyskanie modelu RP3Beta jako szczególnego przypadku wymagałoby przyjęcia reprezentacji użytkowników, których wymiar równy jest liczbie użytkowników. W praktycznych zastosowaniach przyjęcie tak dużej reprezentacji może nie być możliwe ze względu na ograniczoną pamięć oraz moc obliczeniową. Liczba użytkowników jest zwykle dużo większa od liczby przedmiotów, a zatem utrata informacji związana z koniecznością ograniczenia wymiarowości reprezentacji jest prawdopodobnie mniejsza w przypadku modelu P3GNN niż w przypadku jego odpowiednika.

**Model RP3Beta jako szczególny przypadek modelu P3GNN**

Pokażemy, że model RP3Beta jest szczególnym przypadkiem modelu P3GNN (tzn. istnieje zestaw parametrów modelu P3GNN, przy których predykcje ocen są jednakowe dla obu modeli). Ponumerujmy przedmioty $i \in \mathcal{I}$ jako kolejne liczby całkowite dodatnie. Przyjmijmy początkową reprezentacją przedmiotu $i$ jako $|\mathcal{I}|$-wymiarowy wektor, którego współrzędna równa jego numerowi przyjmuje wartość $\frac{1}{|\mathcal{N}(i)|^{\frac{\beta}{2}}}$, zaś pozostałe współrzędne przyjmują wartość zero, gdzie $\beta \in \mathbb{R}$. Zauważmy, iż reprezentacja użytkownika $u$ w trzeciej warstwie modelu P3GNN jest $|\mathcal{I}|$-wymiarowym wektorem, którego $k$-ta współrzędna jest sumą wartości przypisanych do ścieżek długości 3 łączących przedmiot o numerze $k$ z użytkownikiem $u$. Wartość przypisana do zadanej ścieżki $(i, u', i', u)$ wynosi

$$\frac{1}{|\mathcal{N}(i)|^{\frac{\beta}{2}}} \frac{1}{|\mathcal{N}(u')|^{d^{(0)}}} \frac{1}{|\mathcal{N}(i')|^{d^{(1)}}} \frac{1}{|\mathcal{N}(u)|^{d^{(2)}}}.$$

Przyjmijmy ponadto $d^{(0)} = d^{(1)} = d^{(2)} = \alpha$ dla pewnej ustalonej liczby rzeczywistej $\alpha$. Predykcję oceny użytkownika $u$ względem przedmiotu $i$ definiujemy jako iloczyn skalarny reprezentacji użytkownika i przedmiotu. Zatem predykcja oceny, identycznie jak w modelu RP3Beta, jest sumą następujących wartości przypisanych do ścieżek długości 3:

$$\frac{1}{|\mathcal{N}(i)|^{\beta}} \frac{1}{|\mathcal{N}(u')|^{\alpha}} \frac{1}{|\mathcal{N}(i')|^{\alpha}} \frac{1}{|\mathcal{N}(u)|^{\alpha}}.$$

Zatem przy takim doborze początkowych reprezentacji wierzchołków, uzyskujemy model równoważny modelowi RP3Beta.

Poza możliwością uzyskania reprezentacji wektorowych wierzchołków, zaletą modelu P3GNN względem modelu P3LTR jest większa wydajność procedury trenowania. Umożliwia ona wykorzystanie większej liczby parametrów oraz cech wierzchołków. Z drugiej strony, model P3LTR redukuje się do modelu RP3Beta przy odpowiednim doborze parametrów. Model P3GNN również jest uogólnieniem modelu RP3Beta, jednak wymaga przyjęcia dużej wymiarowości początkowych cech przedmiotów, co może nie być wykonalne w praktycznych zastosowaniach. Przy mniejszej wymiarowości istnieje ryzyko uzyskania przez model P3GNN wyników gorszych niż w przypadku modelu RP3Beta. W przypadku modeli wspólnej filtracji zakładamy jednak, że początkowe cechy przedmiotów są parametrami optymalizowanymi w procesie uczenia. Podobnie jak w modelach faktoryzacji macierzy, mniejsza wymiarowość zmusza model do kompresji informacji, zatem sądzimy, iż nawet przy mniejszej liczbie wymiarów możliwe jest uzyskanie dobrych wyników.

## 7.4. Plan eksperymentu

Wszystkie opisane grafowe sieci neuronowe zostały zaimplementowane przez autora z wykorzystaniem bibliotek DGL [122] oraz PyTorch [96]. W celu generowania rekomendacji na podstawie uzyskanych reprezentacji wektorowych wierzchołków wykorzystana została metoda przybliżonych najbliższych sąsiadów oferowana przez bibliotekę Annoy[2] ze względu na jej popularność i łatwość użycia.

### 7.4.1. Zbiór danych

W przeciwieństwie do poprzednich badań, zdecydowaliśmy się wykorzystać zbiór danych pochodzący z jednego z mniejszych rynków, na których wdrażane są nasze rozwiązania. Uznaliśmy, iż znacznie przyspieszy to proces implementacji (ze względu na mniejszą potrzebę optymalizacji) oraz ewaluacji proponowanych modeli, a ponadto obniży koszty. Takie postępowanie, szczególnie w przypadku głębokich sieci neuronowych, niesie za sobą ryzyko. Możliwe jest bowiem, iż metody mniej skuteczne na mniejszym zbiorze danych uzyskają lepsze wyniki na większych zbiorach. Z tego powodu planujemy przeprowadzić podobne badania na większych zbiorach wykorzystując jednak wnioski z prezentowanych badań (np. poprzez zawężenie przestrzeni optymalizowanych hiperparametrów).

Podczas prowadzenia tych badań, wdrożone przez nas modele rekomendacji odpowiadały już za istotną część interakcji generowanych w serwisach Pracodawcy. W przypadku

[2] `https://github.com/spotify/annoy`, dostęp: 2023-11-18

analizowanego zbioru danych stanowiły one około 30% wszystkich interakcji. Z tego powodu porównanie metod na tym zbiorze danych mogłoby faworyzować model RP3Beta lub modele generujące podobne do niego rekomendacje. Zdecydowaliśmy się zatem usunąć te interakcje, zarówno ze zbioru treningowego, jak i ze zbioru testowego.

Dokonaliśmy podziału zbioru danych na zbiór treningowy i testowy w taki sposób, że około 20% najnowszych interakcji każdego użytkownika przypisane było do zbioru testowego. Finalnie, zbiór treningowy zawierał 54 480 interakcji wykonanych przez 10 599 użytkowników względem 2 569 ogłoszeń.

### 7.4.2. Przeszukiwanie przestrzeni hiperparametrów

W celu przeszukiwania przestrzeni hiperparametrów, podzieliliśmy opisany zbiór treningowy na zbiór treningowy i walidacyjny, w ten sam sposób w jaki dokonaliśmy podziału zbioru danych na zbiór treningowy i testowy. Przeszukiwanie przestrzeni hiperparametrów jest kluczowe dla właściwego porównania algorytmów [31]. Jak opisaliśmy w rozdziale 7.2.2, grafowe sieci neuronowe złożone są z wielu elementów. Z każdym z nich możemy związać wiele hiperparametrów. Niestety przeszukiwanie dużej przestrzeni hiperparametrów jest bardzo kosztowne obliczeniowo, czasem wręcz niewykonalne. Konieczne jest zatem ograniczenie się do pewnej liczby hiperparametrów. Przykładowo Wang, Zhao i Shi [128] optymalizowali jedynie 3 hiperparametry w rozważanych przez siebie grafowych sieciach neuronowych, zaś dla pozostałych przyjęli często stosowane wartości.

Podczas optymalizacji chcieliśmy znaleźć zarówno liczbę warstw modelu, jak i wymiarowość poszczególnych warstw. Z tego powodu zdecydowaliśmy się wykorzystać bibliotekę Optuna [8], która umożliwia zdefiniowanie zależności pomiędzy hiperparametrami poprzez możliwość dynamicznego opisania przestrzeni przeszukiwanych hiperparametrów (ang. *define-by-run*). Kolejne zbiory wartości hiperparametrów wyznaczyliśmy za pomocą bayesowskiej metody TPE (ang. *Tree-structured Parzen Estimator* [17]). W metodzie tej w początkowych próbach hiperparametry generowane są losowo (w naszym przypadku wykonywaliśmy 10 takich prób). W każdej kolejnej $N+1$-szej iteracji dzielimy dotychczas testowane zbiory hiperparametrów na podzbiory $L$ i $G$. Do podzbioru $L$ zaliczamy $\lceil \gamma N \rceil$ obserwacji, dla których uzyskaliśmy najlepszy wynik optymalizowanej metryki, gdzie $\gamma$ jest ustalonym hiperparametrem (w naszych badaniach przyjęliśmy $\gamma = 0{,}1$), zaś $\lceil x \rceil$ oznacza najmniejszą liczbę całkowitą nie mniejszą od $x$. Pozostałe obserwacje umieszczamy w zbiorze $G$. Następnie dla zbiorów $L$ i $G$, wyznaczamy funkcje gęstości prawdpodobieństwa (odpowiednio $l$ i $g$) w taki sposób, aby masa prawdopodobieństwa skupiona była wokół obserwacji należących do tych zbiorów. W kolejnej próbie testujemy zestaw hiperparametrów $x$, dla którego wartość $\frac{g(x)}{l(x)}$ jest możliwie najmniejsza.

Dla każdego modelu wykonaliśmy 100 prób optymalizując metrykę Precision@10. Przyjęliśmy ograniczenie 30 minut na czas trwania pojedynczej próby. Po tym czasie

trening był przerywany zwracając wartość metryki równą -1. Ograniczenie to zostało osiągnięte 9 razy podczas przeprowadzenia łącznie 500 prób. Zarówno w procesie optymalizacji hiperparametrów jak i przy finalnym trenowaniu modeli przerywaliśmy trening, gdy suma wartości funkcji straty z ostatnich 50 partii danych była nie mniejsza od sumy wartości funkcji straty z poprzednich 50 partii danych.

W tabeli 7.2 prezentujemy opis oraz możliwe wartości hiperparametrów, które optymalizowaliśmy. Przypomnijmy, iż dla naszego zbioru danych konieczne jest zdefiniowanie metody przekazywania informacji dla trójek typów (użytkownik, interakcja, przedmiot) oraz (przedmiot, interakcja, użytkownik) w każdej warstwie. Moglibyśmy zatem zdefiniować różne metody dla poszczególnych warstw lub typów z odrębnymi zestawami hiperparametrów, co jednak zwiększyłoby ich liczbę i złożoność modelu. W tych badaniach postanowiliśmy rozważać jedynie sytuacje, gdzie metody przekazywania wiadomości dla wszystkich typów oraz warstw są jednakowe (a w przypadku modelu P3GNN zdefiniowane w rozdziale 7.3.5), opisane przez te same hiperparametry, lecz z odrębnymi parametrami.

W celu uczciwego porównania przeszukaliśmy także przestrzeń hiperparametrów modelu RP3Beta, gdzie za zbiór dopuszczalnych wartości hiperparametrów $\alpha$ oraz $\beta$ przyjęliśmy przedział $[0, 2]$.

Rysunek 7.1 przedstawia wartości hiperparametrów optymalizowanych modeli w kolejnych próbach. Zauważmy, iż wartość metryki Precision@10 co najwyżej nieznacznie rośnie po wykonaniu 50 iteracji. Można zatem przypuszczać, że wykonanie więcej niż 100 iteracji nie umożliwiłoby nam znalezienia istotnie lepszych hiperparametrów, a jedynie zwiększyłoby koszt poszukiwań.

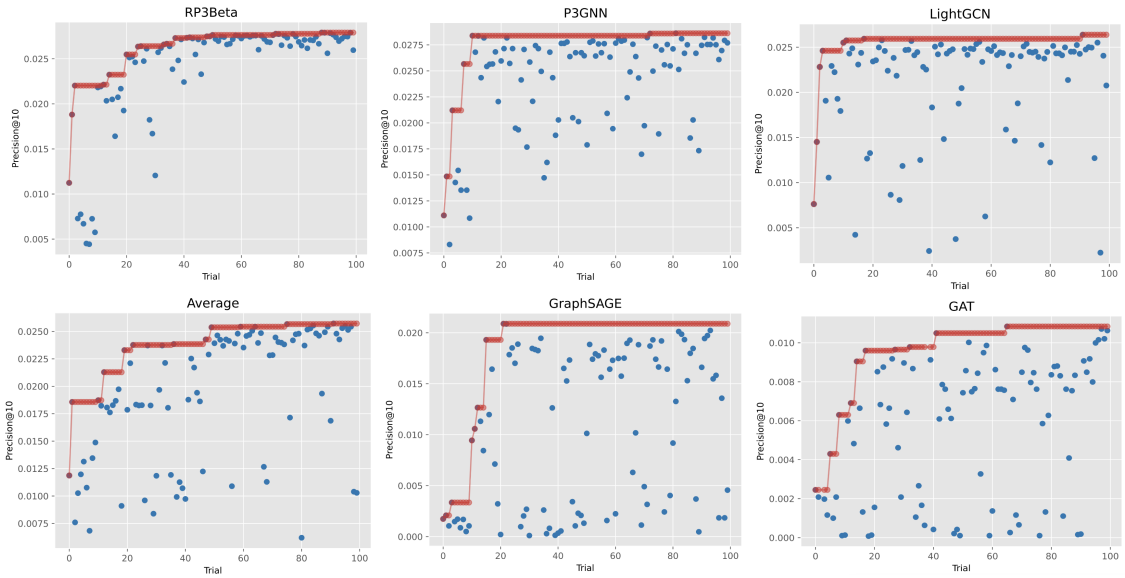
Tabela 7.3 zawiera znalezione optymalne wartości hiperparametrów dla każdego z modeli. W przypadku modeli LightGCN, Average oraz P3GNN, liczba wymiarów poszczególnych warstw jest jednakowa i równa wymiarowi początkowych reprezentacji, więc wyznaczanie wymiarowości poszczególnych warstw zostało pominięte. W trakcie badań zaobserwowaliśmy wzrost jakości modeli wraz ze wzrostem wymiarowości początkowej reprezentacji. Z drugiej strony wyższa wymiarowość reprezentacji oznacza dłuższy czas trenowania modelu. Zdecydowaliśmy się na ustalenie jej wymiaru na 256. W przypadku modelu P3GNN liczba warstw wynosi 3, więc wartość ta również nie była optymalizowana. Finalna reprezentacja wierzchołków może być uzyskana jako średnia reprezentacji wierzchołków z poszczególnych warstw tylko jeśli wymiarowość wszystkich warstw jest jednakowa, co nie zachodzi w ogólności dla modeli GAT i GraphSAGE. W tych modelach przyjęliśmy, że finalna reprezentacja to reprezentacja z ostatniej warstwy.

Na rysunku 7.2 prezentujemy istotność poszczególnych hiperparametrów dla każdego z rozważanych modeli obliczoną za pomocą algorytmu fANOVA [65]. Przedstawione zostały jedynie hiperparametry występujące w każdej próbie (przykładowo hiperpara-

**Tabela 7.2.** Opis optymalizowanych hiperparametrów

| Hiperparametr | Opis | Typ | Zbiór wartości |
|---|---|---|---|
| epochs | liczba epok | liczba całkowita | [5, 50] |
| batch_size | rozmiar partii | liczba całkowita | [128, 8192] |
| loss | funkcja straty | kategoryczny | {bpr, hinge} |
| hinge_constant | wartość stałej w funkcji straty Hinge | liczba rzeczywista | [0, 1] |
| learning rate | współczynnik uczenia | liczba rzeczywista | [0,001, 1] |
| regularization | współczynnik regularyzacji (L2) | liczba rzeczywista | [0, 0,1] |
| nb_negatives | liczba przykładów negatywnych dla każdego przykładu pozytywnego | liczba całkowita | [1,10] |
| norm | funkcja normalizacji (jednakowa dla każdej warstwy) | kategoryczny | {None, L1, L2} |
| n_layers | liczba warstw | liczba całkowita | [1,3] |
| layer_dim_1 | wymiarowość wyjścia z warstwy numer 1 | liczba całkowita | [64, 256] |
| layer_dim_2 | wymiarowość wyjścia z warstwy numer 2 | liczba całkowita | [64, layer_dim_1] |
| layer_dim_3 | wymiarowość wyjścia z warstwy numer 3 | liczba całkowita | [64, layer_dim_2] |
| final_repr | sposób uzyskania finalnej reprezentacji wierzchołków | kategoryczny | {last, average} |
| aggregator | funkcja aggregate w modelu GraphSAGE | kategoryczny | {mean, gcn, pool} |
| activation | funkcja aktywacji | kategoryczny | {ReLU, sigmoid, None} |
| negative_slope | współczynnik nachylenia funkcji LeakyRELU | liczba rzeczywista | [0,001, 0,5] |
| num_heads | liczba mechanizmów atencji | liczba całkowita | [1, 10] |

metr hinge_constant nie został uwzględniony, ponieważ nie występuje on dla funkcji straty BPR). W celu łatwiejszego porównania istotności pomiędzy algorytmami, wartości zostały przeskalowane w taki sposób, że suma istotności wszystkich hiperparametrów modelu wynosi 1. Zauważmy, iż sposób normalizacji znajduje się wśród dwóch najważniejszych hiperparametrów dla wszystkich rozważanych grafowych sieci neuronowych. W modelach Average oraz LightGCN, dla których optymalizowany był hiperparametr final_repr, uzyskał on najwyższą istotność.




**Rysunek 7.1.** Wartość metryki Precision@10 w kolejnych próbach optymalizacji hiperparametrów. Niebieskie kropki oznaczają wartość metryki Precision@10 w danej próbie, zaś czerwoną linią zaznaczone są najwyższe wyniki uzyskane w dotychczas przeprowadzonych próbach

## 7.5. Wyniki

W podrozdziale przedstawimy wyniki uzyskane na zbiorze testowym. Rozpoczniemy od przedstawienia wyników modeli wytrenowanych z wykorzystaniem hiperparametrów przedstawionych w tabeli 7.3. W kolejnych podrozdziałach przyjrzymy się wpływowi kilku wybranych elementów sieci.

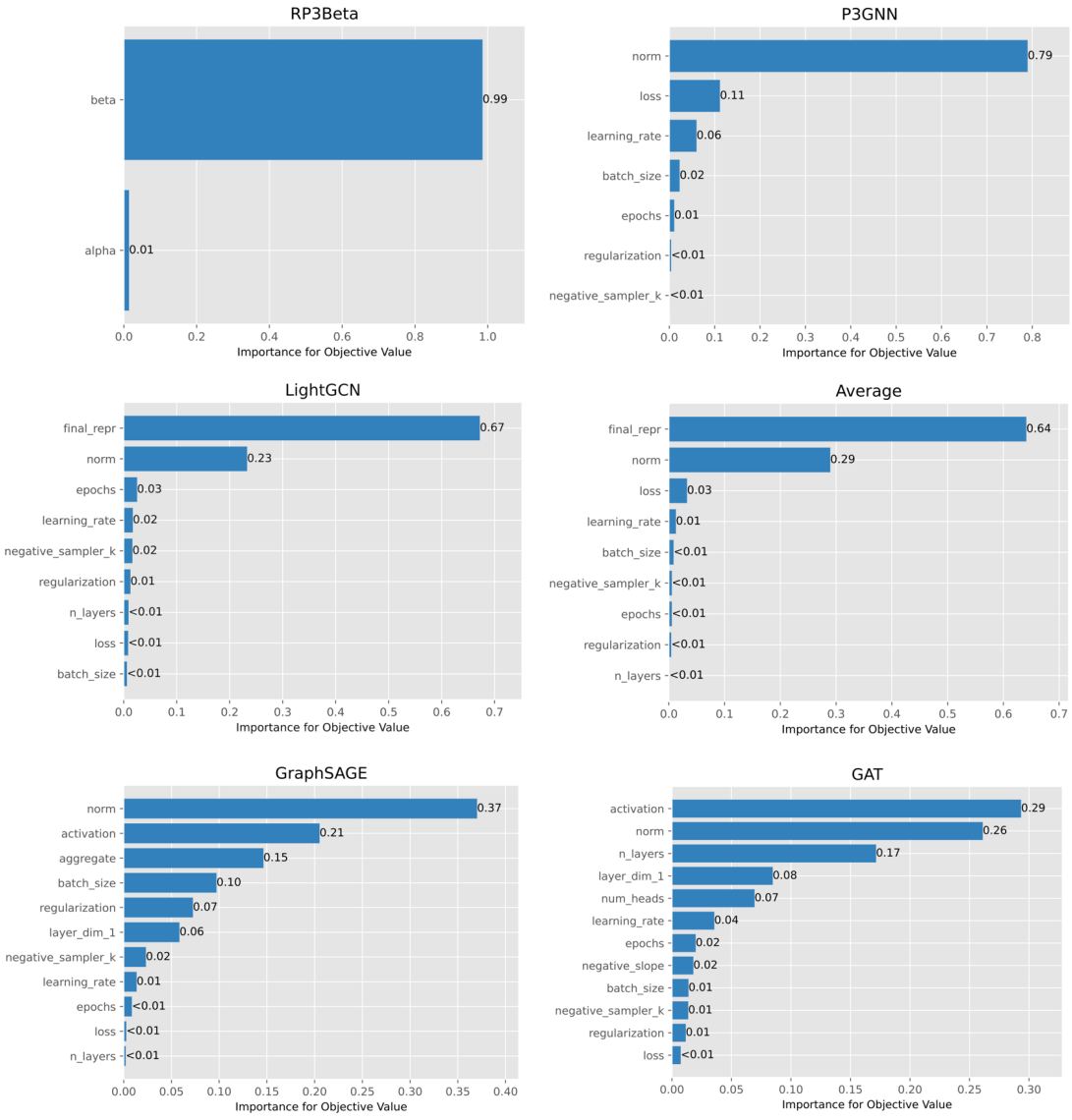
### 7.5.1. Ogólne wyniki

Wykorzystaliśmy znalezione na zbiorze walidacyjnym hiperperametry w celu wytrenowania modeli na pełnym zbiorze treningowym i przeprowadzenia ewaluacji na zbiorze testowym. Tabela 7.4 przedstawia uzyskane wyniki. Widzimy, że modele RP3Beta oraz P3GNN uzyskały podobne wyniki, z niewielką przewagą modelu RP3Beta w przypadku naszej głównej metryki, Precision@10. Nieznacznie gorsze wyniki prezentują modele LightGCN oraz Average. Podobne wyniki modeli P3GNN, LightGCN oraz Average wynikać mogą z dużego podobieństwa w budowie tych modeli. Modele GraphSAGE oraz GAT uzyskały znacznie gorsze wyniki. Być może liczba parametrów tych modeli była zbyt duża w stosunku do zbioru danych, przez co model nadmiernie dopasował

**Tabela 7.3.** Wartości hiperparametrów, przy których uzyskaliśmy najwyższe wyniki na zbiorze walidacyjnym. Symbol „*" po wartości hiperparametru oznacza, iż nie był on optymalizowany. Symbol „-" oznacza, iż dany hiperparametr nie występuje w przypadku zadanego modelu, zaś symbol „•" oznacza, że hiperparametr nie występuje z powodu wartości przyjętych przez pozostałe hiperparametry

| **Hiperparametr** | **P3GNN** | **LightGCN** | **Average** | **GraphSAGE** | **GAT** |
|---|---|---|---|---|---|
| epochs | 20 | 34 | 38 | 32 | 23 |
| batch_size | 3948 | 1430 | 7223 | 7005 | 6231 |
| loss | hinge | hinge | hinge | bpr | bpr |
| hinge_constant | 0,2311 | 0,4782 | 0,4495 | • | • |
| learning rate | 0,0252 | 0,0031 | 0,2369 | 0,6710 | 0,5075 |
| regularization | 0,0658 | 0,0980 | 0,0415 | 0,0083 | 0,0283 |
| nb_negatives | 10 | 9 | 5 | 9 | 7 |
| norm | l2 | l2 | l2 | None | None |
| n_layers | 3* | 1 | 2 | 1 | 1 |
| layer_dim_1 | 256* | 256* | 256* | 173 | 167 |
| layer_dim_2 | 256* | • | 256* | • | • |
| layer_dim_3 | 256* | • | • | • | • |
| final_repr | - | last | last | last* | last* |
| aggregator | - | - | - | gcn | - |
| activation | - | - | - | None | None |
| negative_slope | - | - | - | - | 0,4579 |
| num_heads | - | - | - | - | 8 |

się do danych treningowych. Jak wspominaliśmy wcześniej, sytuacje w których bardziej złożone modele neuronowe uzyskują słabsze wyniki od metod bazowych są powszechne w systemach rekomendacyjnych [31, 32].




**Rysunek 7.2.** Istotność poszczególnych hiperparametrów porównywanych metod

### 7.5.2. Funkcja straty

Zgodnie z rysunkiem 7.2 funkcja straty jest drugim najbardziej istotnym hiperparametrem modelu P3GNN. W podrozdziale sprawdzimy skuteczność różnych funkcji straty dla

**Tabela 7.4.** Wartości metryk dokładności oraz pokrycia dla porównywanych metod

| Metryka | RP3Beta | P3GNN | LightGCN | Average | GraphSAGE | GAT |
|---|---|---|---|---|---|---|
| Precision | **0,0197** | 0,0196 | 0,0188 | 0,0184 | 0,0129 | 0,0062 |
| Recall | 0,1300 | **0,1324** | 0,1258 | 0,1258 | 0,0818 | 0,0369 |
| NDCG | **0,0779** | 0,0770 | 0,0735 | 0,0707 | 0,0452 | 0,0202 |
| MAP | **0,0557** | 0,0544 | 0,0519 | 0,0487 | 0,0296 | 0,0125 |
| MRR | **0,0705** | 0,0675 | 0,0653 | 0,0612 | 0,0403 | 0,0197 |
| LAUC | 0,5627 | **0,5638** | 0,5605 | 0,5605 | 0,5385 | 0,5160 |
| HR | 0,1656 | **0,1666** | 0,1590 | 0,1590 | 0,1109 | 0,0556 |
| Pokrycie zbioru testowego | 0,8722 | **0,9113** | 0,8935 | 0,8683 | 0,8717 | 0,6365 |
| Entropia Shannona | 6,4828 | 6,8076 | 6,7644 | 6,8301 | **6,8612** | 6,2278 |
| Indeks Giniego | 0,7258 | 0,6538 | 0,6692 | 0,6511 | **0,6396** | 0,8064 |

tego modelu. W rozważanych badaniach uwzględniamy funkcje straty postaci:

$$\lambda\|\mathbf{\Theta}\|^2 + \frac{1}{|\mathcal{S}|}\sum_{(u,i,j)\in\mathcal{S}} l(\hat{r}_{ui} - \hat{r}_{uj}),$$

gdzie

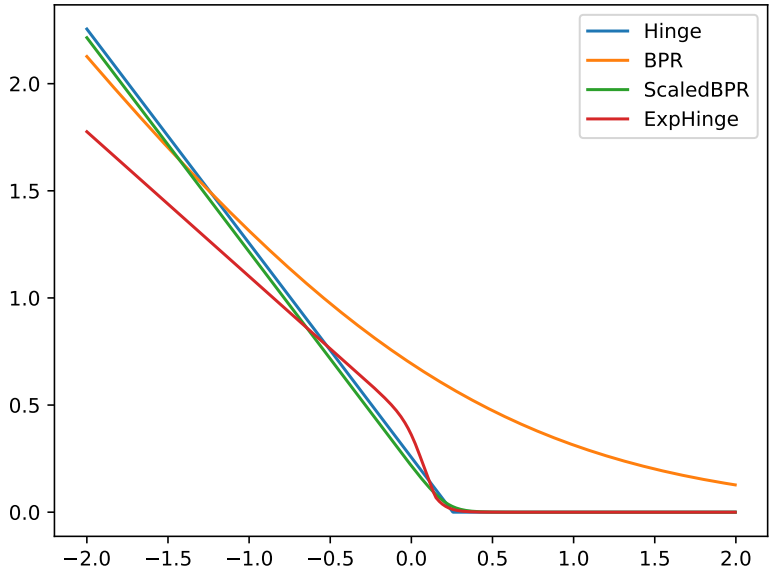
$$\mathcal{S} = \{(u,i,j)|u \in \mathcal{U}, i \in \mathcal{N}(u) \text{ oraz } j \in \text{Neg}(u,i)\},$$

$\lambda \geq 0$ jest hiperparametrem regularyzacji specyficznym dla modelu, $\mathbf{\Theta}$ jest wektorem parametrów, $\|\cdot\|$ jest normą euklidesową, zaś $\text{Neg}(u,i)$ oznacza zbiór wierzchołków, które przyjmujemy jako przykłady negatywne dla użytkownika $u$ względem przedmiotu pozytywnego $i$. W dalszej części rozdziału będziemy nazywać funkcję $l$ funkcją straty, jeśli nie będzie to prowadzić do nieporozumień. W procesie trenowania modelu rekomendacji chcemy doprowadzić do sytuacji, w której $\hat{r}_{ui} > \hat{r}_{uj}$, tzn. przykłady pozytywne posiadają wyższe predykcje ocen od przykładów negatywnych. Z tego powodu rozważamy nierosnące funkcje $l$. Rozważymy cztery postacie funkcji $l$, z których trzy posiadają dodatkowe hiperparametry. Wartości tych hiperparametrów znaleźliśmy na zbiorze walidacyjnym, przy czym wartość pozostałych hiperparametrów została ustalona zgodnie z tabelą 7.3. Dla każdej z funkcji wykonaliśmy 50 iteracji. Rysunek 7.3 przedstawia znalezione w ten sposób funkcje $l$. Funkcje te przedstawiamy jedynie na przedziale $[-2,2]$. Pozostałe wartości nie mają wpływu na proces treningu modeli, ponieważ przy normalizacji l2, wartości $\hat{r}_{ui}$ oraz $\hat{r}_{uj}$ modelu pochodzą z przedziału $[-1,1]$, a zatem wartości $\hat{r}_{ui} - \hat{r}_{uj}$ należą do przedziału $[-2,2]$. Definicję poszczególnych funkcji przedstawiamy poniżej.

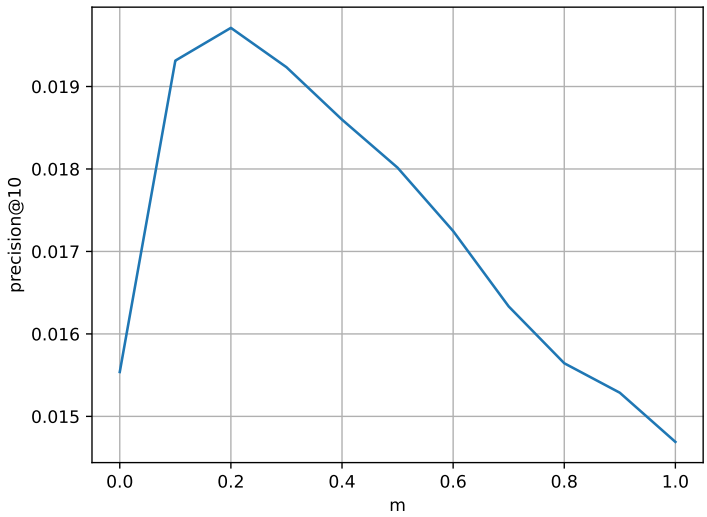
Funkcję straty **Hinge** [13] uzyskujemy przyjmując następującą postać funkcji $l$:

$$l(x) = \max(m - x, 0),$$

gdzie $m \geq 0$ jest ustalonym hiperparametrem. Hiperparametr ten służy do uzyskania przez przykłady pozytywne wyższych predykcji ocen od przykładów negatywnych o co




**Rysunek 7.3.** Postać funkcji $l$ dla analizowanych funkcji straty




**Rysunek 7.4.** Wartość metryki Precision@10 w zależności od wartości hiperparametru $m$ w funkcji straty Hinge

najmniej $m$. Na rysunku 7.4 przedstawiliśmy wartość metryki Precision@10 uzyskanej na zbiorze testowym w zależności od wartości hiperparametru $m$ dla modelu P3GNN, gdzie wartość pozostałych hiperparametrów została ustalona zgodnie z tabelą 7.3. Istotny

wpływ hiperparametru $m$ na jakość modelu pokazuje jak istotne jest dobranie właściwej funkcji straty. Z tego powodu ten podrozdział poświęcony jest optymalizacji tego elementu budowy sieci. W przypadku funkcji Hinge ograniczyliśmy poszukiwania hiperparametru $m$ do przedziału $[0, 1]$. Najwyższa wartość metryki Precision@10 na zbiorze walidacyjnym została uzyskana dla $m = 0{,}2551$.

Poniżej opiszemy pewne właściwości funkcji Hinge, które zainspirowały autora do konstrukcji nowej funkcji straty. Rozpatrzmy punkty $(u_1, i_1, j_1), (u_2, i_2, j_2) \in S$. Oznaczmy $x_1 = \hat{r}_{u_1 i_1} - \hat{r}_{u_1 j_1}$ oraz $x_2 = \hat{r}_{u_2 i_2} - \hat{r}_{u_2 j_2}$ oraz załóżmy, że $-2 \leq x_1 < x_2 \leq m$. Niech $\varepsilon$ będzie pewną liczbę dodatnią taką, że $x_2 + \varepsilon \leq m$ Zauważmy, iż wartość funkcji straty Hinge zmieni się tak samo niezależnie od tego czy zwiększymy o $\varepsilon$ wartość $x_1$ czy $x_2$, tzn. $l(x_1 + \varepsilon) + l(x_2) = l(x_1) + l(x_2 + \varepsilon)$. O kolejności przedmiotów na liście rekomendacji decyduje jednak jedynie dodatniość punktów $x_1$ oraz $x_2$. Z tego powodu korzystne może być przyjęcie funkcji $l$, której wartości zmniejszają się szybciej w okolicach zera. Ponadto zauważmy dużą zmianę funkcji $l$, która zachodzi w punkcie $m$. Dla wartości argumentów poniżej $m$ pochodna funkcji $l$ wynosi -1, zaś dla argumentów powyżej $m$ wynosi ona 0. Zerowa wartość pochodnej oznacza, iż dalszy wzrost wartości argumentów nie powoduje już zmniejszenia wartości funkcji. Być może skuteczniejsza byłaby funkcja straty, której zmiana wartości o $\varepsilon$ w pewnym otoczeniu punktu $M$ powodowałaby zmiany wartości funkcji straty o mniej niż $\varepsilon$, lecz więcej niż 0.

Problemy te adresujemy proponując funkcję **ExpHinge** zdefiniowaną wzorem:

$$l(x) = p \max(m - x, 0) + (1 - p)\sigma((-x + \mu)s),$$

gdzie $\sigma(x) = \frac{1}{1+e^{-x}}$, zaś $m$, $p$, $\mu$ oraz $s$ są hiperparametrami modelu. Zauważmy, iż dla $p = 1$ funkcja ExpHinge redukuje się do funkcji Hinge. W procesie przeszukiwania optymalnych hiperparametrów przyjęliśmy $p, m \in [0, 1]$, $\mu \in [-1, 1]$ oraz $s \in [0, 20]$. Najwyższa wartość metryki Precision@10 na zbiorze walidacyjnym została uzyskana przy następujących wartościach: $p = 0{,}6745, m = 0{,}15028, \mu = 0{,}0768, s = 17{,}8465$. Rysunek 7.3 ilustruje funkcje Hinge and ExpHinge przy uzyskanych wartościach hiperparametrów. Zauważmy, iż funkcje te zauważalnie różnią się od siebie, a więc funkcja ExpHinge nie zredukowała się do funkcji Hinge będącej jej szczególnym przypadkiem. Ponadto, funkcja ExpHinge adresuje opisane wcześniej ograniczenia funkcji Hinge.

Kolejną rozważaną funkcją straty jest funkcja **BPR**, dla której funkcja $l$ zdefiniowana jest następująco:

$$l(x) = -\ln \sigma(x).$$

Funkcja ta jest malejąca i wypukła. W tym przypadku, posługując się oznaczeniami wprowadzonymi przy funkcji ExpHinge, mamy $l(x_1 + \varepsilon) + l(x_2) < l(x_1) + l(x_2 + \varepsilon)$. Zatem korzystniejsza z punktu widzenia minimalizacji tej funkcji straty jest aktualizacja parametrów modelu w taki sposób, że zwiększymy o $\varepsilon$ argument $x_1$ zamiast argumentu $x_2$. Zauważmy ponadto, iż funkcja BPR jest ściśle malejąca, w przeciwieństwie do

**Tabela 7.5.** Wartości metryk dokładności oraz pokrycia dla porównywanych metod

| Metryka | ExpHinge | Hinge | ScaledBPR | BPR |
|---|---|---|---|---|
| Precision | **0,0196** | 0,0196 | 0,0193 | 0,0138 |
| Recall | **0,1326** | 0,1312 | 0,1307 | 0,0935 |
| NDCG | **0,0772** | 0,0764 | 0,0761 | 0,0525 |
| MAP | **0,0546** | 0,0539 | 0,0539 | 0,0358 |
| MRR | **0,0679** | 0,0675 | 0,0667 | 0,0460 |
| LAUC | **0,5639** | 0,5633 | 0,5630 | 0,5443 |
| HR | **0,1665** | 0,1653 | 0,1625 | 0,1204 |
| Pokrycie zbioru testowego | 0,8965 | **0,9064** | 0,8990 | 0,8757 |
| Entropia Shannona | 6,7268 | 6,7816 | 6,7737 | **6,8132** |
| Indeks Giniego | 0,6767 | 0,6628 | 0,6645 | **0,6588** |

funkcji Hinge, która jest stała dla argumentów wyższych od hiperparametru $m$. Zatem wartość funkcji BPR możemy zmniejszyć zwiększając różnicę pomiędzy predykcjami dla przykładów pozytywnych i negatywnych, niezależnie od tego jak duża jest ta różnica.

Postanowiliśmy uogólnić funkcję BPR w taki sposób, aby tempo spadku jej wartości zależało od jej hiperparametrów. Funkcję ScaledBPR definujemy wzorem:

$$l(x) = -\frac{1}{s} \ln \sigma((x - m)s),$$

gdzie $m$ oraz $s$ to hiperparametry. Zauważmy, że czynnik $\frac{1}{s}$ nie zależy od $x$, więc jego pominięcie (przy odpowiedniej modyfikacji parametru regularyzacji i współczynnika uczenia) nie ma wpływu na działanie modelu. Umieściliśmy go jednak w celu łatwiejszej wizualizacji funkcji ScaledBPR w porównaniu z innymi funkcjami straty. Autor przypuszcza również, iż utrzymywanie wartości funkcji strat w podobnym przedziale ułatwia znalezienie optymalnych wartości regularyzacji i współczynnika uczenia w procesie optymalizacji hiperparametrów. Najwyższa wartość metryki Precision@10 na zbiorze walidacyjnym została uzyskana przy następujących wartościach: $m = 0{,}2147$, $s = 16{,}1498$. Na rysunku 7.3 możemy zauważyć, iż uzyskana funkcja ScaledBPR niewiele różni się od funkcji Hinge, lecz znacznie odbiega od funkcji BPR.

Znalezione w procesie optymalizacji hiperparametrów na zbiorze walidacyjnym funkcje straty wykorzystaliśmy do trenowania modelu na pełnym zbiorze treningowym i ewaluacji na zbiorze testowym. Wyniki przedstawia tabela 7.5. Funkcje ExpHinge, Hinge oraz ScaledBPR uzyskały zbliżone wyniki, czego można było spodziewać się biorąc pod uwagę ich podobieństwo przedstawione na rysunku 7.3. Funkcja ExpHinge uzyskała najlepsze wyniki pod względem wszystkich metryk dokładności, przy czym przewaga nad funkcją Hinge jest niewielka (w przypadku metryki Precision@10 przewaga ta nie jest widoczna po zaokrągleniu). Przy zastosowaniu zaproponowanej funkcji ScaledBPR

uzyskaliśmy lepsze wyniki niż w przypadku powszechnie stosowanej funkcji BPR. Niestety w przypadku naszego zbioru danych funkcja BPR daje dużo słabszy wynik od funkcji Hinge. W rezultacie za pomocą funkcji ScaledBPR uzyskaliśmy wynik zbliżony do funkcji Hinge.

Podsumowując, za pomocą zaproponowanych funkcji ExpHinge oraz ScaledBPR uzyskaliśmy porównywalne lub lepsze wyniki niż w przypadku zastosowania ich znanych odpowiedników.

### 7.5.3. Trudniejsze przykłady uczące

W dotychczasowych badaniach przykłady negatywne wybieraliśmy w sposób losowy w następujący sposób. Dla zadanej krawędzi łączącej wierzchołek $u$ z wierzchołkiem $v$, losowaliśmy z powtórzeniami $k$ wierzchołków ze zbioru wszystkich wierzchołków tego samego typu co wierzchołek $v$ (rozważaliśmy dwa typy wierzchołków: użytkowników i przedmioty). Krawędzie łączące wierzchołek $u$ z tymi krawędziami traktowaliśmy jako negatywne.

W praktycznych zastosowaniach liczba i różnorodność przedmiotów jest zwykle na tyle duża, że zdecydowana większość przykładów negatywnych istotnie różni się od przykładów pozytywnych. Przykładem negatywnym dla użytkownika poszukującego pracy jako analityk danych w Warszawie może być oferta pracy jako hydraulik w Rzeszowie. W omawianym przykładzie ważne jest, aby model uzyskał zdolność rozróżniania przedmiotów z branży informatycznej w Warszawie, co może być trudne bez odpowiedniej liczby przykładów negatywnych w zbiorze uczącym. Problem ten adresujemy poprzez dodanie do losowych przykładów negatywnych, tzw. przykładów *trudniejszych*. Wykorzystanie podobnego rozwiązania okazało się skuteczne w przypadku modelu PinSage [135]. Poniżej prezentujemy szczegóły proponowanego przez nas rozwiązania.

Rozpoczniemy od opisania procesu wyliczania zadanej liczby $k_h$ trudniejszych przykładów negatywnych dla zadanego wierzchołka $u$. Generujemy $N = 1000$ spacerów losowych o długości 3 rozpoczynających się w wierzchołku $u$. Niech $l$ oznacza listę przedmiotów, w których zakończył się przynajmniej jeden spacer losowy, a które nie są bezpośrednimi sąsiadami wierzchołka $u$. Przyjmijmy ponadto, że lista $l$ jest posortowana rosnąco względem liczby ścieżek, które zakończyły się w danym wierzchołku. Niech $h \in [0, 1]$ będzie ustalonym hiperparametrem określającym trudność wybranych przykładów. Ponadto, ustalmy pewną liczbą całkowitą $k_{max}$ (w naszych badaniach przyjęliśmy $k_{max} = 50$). Z listy $l$ wybieramy $k_{max}$ kolejnych elementów rozpoczynając od elementu o numerze $\lfloor h|l| \rfloor + 1$, gdzie $|l|$ oznacza liczbę elementów listy $l$, zaś $\lfloor x \rfloor$ oznacza największą liczbę całkowitą nie większą od $x$. Dodatkowo:

- jeśli na liście $l$ nie istnieje $k_{max}$ elementów rozpoczynając się od $\lfloor h|l| \rfloor + 1$, to brane jest ostatnie $k_{max}$ elementów,

- jeśli lista $l$ zawiera mniej niż $k_{max}$ elementów, to brana jest pod uwagę cała lista oraz $k_{max} - |l|$ losowo wygenerowanych wierzchołków.

Z tak wybranych elementów losowane jest $k_h$ trudniejszych przykładów.

Sprawdziliśmy, czy dodanie trudniejszych przykładów pozwoli na uzyskanie lepszych wyników modelu P3GNN. Podobnie jak w poprzednim podrozdziale przyjęliśmy wartości hiperparametrów tego modelu zgodnie z tabelą 7.3. Poszukiwaliśmy najlepszych wartości hiperparametrów $k_h$ oraz $h$ poprzez ewaluację na zbiorze walidacyjnym wykonując 50 prób wcześniej opisanego algorytmu TPE. Poszukiwaliśmy wartości hiperparametru $k_h$ w zbiorze $\{0, 1, \ldots, 10\}$ oraz $h$ w zbiorze $[0, 1]$. Procedurę tę wykonaliśmy także dla funkcji strat BPR, ScaledBPR oraz ExpHinge przyjmując hiperparametry tych funkcji straty znalezione w poprzednim podrozdziale. Powodem, dla którego spodziewamy się różnego wpływu dodania przykładów negatywnych w zależności od funkcji straty $l$ jest obserwacja, że trudność przykładów wpływa na rozkład wartości jej argumentów.

W przypadku funkcji straty ExpHinge oraz Hinge najwyższą wartość metryki Precision@10 na zbiorze walidacyjnym uzyskaliśmy przy $k_h = 0$, a zatem bez dodania trudniejszych przykładów. Zaobserwowaliśmy, iż dodanie wielu przykładów negatywnych wpływa negatywnie na jakość modelu przy wykorzystaniu tych funkcji strat. W przypadku modelu ScaledBPR uzyskaliśmy wartości $k_h = 1$ oraz $h = 0{,}5720$. Liczba przykładów trudniejszych dająca najlepsze wyniki na zbiorze walidacyjnym jest jednak znacznie wyższa dla funkcji BPR, dla której $k_h = 9$ oraz $h = 0{,}5927$.

Przy znalezionych wartościach hiperparametrów wytrenowaliśmy modele na pełnym zbiorze treningowym dla funkcji straty BPR oraz ScaledBPR. Funkcje Hinge oraz ExpHinge pominęliśmy, ponieważ dla $k_h = 0$ nasza metoda dodawania trudniejszych przykładów redukuje się do generowania przykładów losowych. W tabeli 7.6 przedstawiliśmy wartości metryk dokładności oraz pokrycia dla zbioru testowego w zależności od wykorzystania lub pominięcie trudniejszych przykładów negatywnych. Zauważmy znaczną poprawę jakości modelu w przypadku funkcji BPR przy wykorzystaniu trudniejszych przykładów negatywnych. Różnica ta może wynikać z faktu, iż w przypadku trudniejszych przykładów argumenty funkcji straty $(\hat{r}_{ui} - \hat{r}_{uj})$ przyjmują mniejsze wartości, a zatem mniejsze znaczenia ma kształt funkcji straty dla dużych argumentów. Przypomnijmy, iż to właśnie dla dużych argumentów funkcja BPR najbardziej różni się od pozostałych rozważanych funkcji (rysunek 7.3). W przypadku funkcji straty ScaledBPR, dodanie trudniejszych przykładów uczących pozwoliło na uzyskanie nieznacznie lepszych wyników względem wszystkich rozważanych metryk. Zauważmy, iż w przypadku obu funkcji straty, dodanie przykładów negatywnych pozytywnie wpłynęło na wartości metryk pokrycia.

Podsumowując, zaproponowana metoda generowania trudniejszych przykładów nie pozwoliła na uzyskanie istotnej poprawy jakości modelu. Uzyskaliśmy jednak dużą poprawę w przypadku powszechnie stosowanej funkcji BPR. Interesujące byłoby zbadanie

**Tabela 7.6.** Wartości metryk dokładności przy wykorzystaniu trudniejszych przykładów negatywnych (ScaledBPR+T, BPR+T) oraz przy wykorzystaniu wyłącznie losowych przykładów negatywnych (ScaledBPR, BPR)

| Metryka | ScaledBPR+T | ScaledBPR | BPR+T | BPR |
|---|---|---|---|---|
| Precision | **0,0197** | 0,0193 | 0,0167 | 0,0138 |
| Recall | **0,1331** | 0,1307 | 0,1130 | 0,0935 |
| NDCG | **0,0775** | 0,0761 | 0,0662 | 0,0525 |
| MAP | **0,0547** | 0,0539 | 0,0467 | 0,0358 |
| MRR | **0,0684** | 0,0667 | 0,0597 | 0,0460 |
| LAUC | **0,5642** | 0,5630 | 0,5541 | 0,5443 |
| HR | **0,1669** | 0,1625 | 0,1431 | 0,1204 |
| Pokrycie zbioru testowego | **0,9237** | 0,8990 | 0,9118 | 0,8757 |
| Entropia Shannona | 6,7933 | 6,7737 | **6,8618** | 6,8132 |
| Indeks Giniego | 0,6511 | 0,6645 | **0,6299** | 0,6588 |

jakości zaproponowanej metody dla zbioru danych w przypadku, którego funkcja straty BPR daje lepsze wyniki od innych funkcji straty.

## 7.6. Podsumowanie

W rozdziale omówiliśmy budowę oraz wykorzystanie grafowych modeli neuronowych w systemach rekomendacyjnych. Wskazaliśmy różnice między proponowanym wcześniej modelem P3LTR a standardowym podejściem stosowanym w sieciach grafowych. Zaproponowaliśmy nowe podejście oparte na modelu RP3Beta, które jest bardziej skalowalne od modelu P3LTR oraz umożliwia generowanie reprezentacji wektorowej wierzchołków. Porównaliśmy proponowane podejście z powszechnie stosowanymi modelami LightGCN, GraphSAGE oraz GAT. Do porównania dodaliśmy także model RP3Beta oraz model Average, prostą grafową sieć neuronową, w której reprezentacje wierzchołków w kolejnych warstwach obliczane są jako średnia reprezentacji sąsiednich wierzchołków z warstwy poprzedniej.

Dla każdego z modeli zdefiniowaliśmy liczne hiperparametry. Szczegółowo opisaliśmy proces ich optymalizacji za pomocą algorytmu TPE, a ponadto podaliśmy istotność poszczególnych hiperparametrów korzystując z algorytmu fANOVA.

Pokazaliśmy, iż proponowany model P3GNN uzyskał najwyższą wartość wszystkich metryk dokładności wśród rozważanych grafowych modeli neuronowych. Jego wynik zbliżony jest do wyniku uzyskanego za pomocą modelu RP3Beta (model P3GNN uzyskał lepsze wyniki od modelu RP3Beta dla 3 z 7 rozważanych metryk).

Następnie zaproponowaliśmy nowe funkcje straty oparte na parach: ScaledBPR

i ExpHinge. Pokazaliśmy, że funkcja ScaledBPR pozwala na uzyskanie lepszych wyników niż powszechnie stosowana funkcja straty BPR, która jest jej przypadkiem szczególnym. Przy zastosowaniu funkcji ExpHinge uzyskaliśmy podobne wyniki co w przypadku funkcji Hinge.

W ostatnim podrozdziale zaproponowaliśmy nową metodą generowania trudniejszych przykładów negatywnych. Pokazaliśmy, iż pozwala ona uzyskać lepsze wyniki w przypadku funkcji straty BPR oraz nieznacznie lepsze w przypadku funkcji straty ScaledBPR.

Podsumowując, zaproponowaliśmy nowy model rekomendacji P3GNN przewyższający pozostałe rozważane grafowe modele neuronowe na rozważanym przez nas zbiorze danych. Model ten pozwala na uzyskanie wyników zbliżonych do modelu RP3Beta, a ponadto pozwala na generowanie reprezentacji wektorowych użytkowników i przedmiotów. Zatem model P3GNN stanowi jedną z metod wspomnianych w celu głównym rozprawy: *przedstawienie nowych metod rekomendacji opracowanych dla serwisów ogłoszeniowych oraz wykazanie ich przewagi względem metod opisanych w literaturze*.

Model P3GNN jako grafowy model neuronowy złożony jest z wielu elementów, których poprawa może pozwolić na uzyskanie lepszych wyników. Przykładowo możliwe jest uwzględnienie w nim cech przedmiotów oraz krawędzi. Uzyskany model jest zatem podstawą do dalszych badań nad wykorzystaniem grafowych sieci neuronowych w celu rekomendacji ofert pracy.

Rozdział 8

# Podsumowanie

Celem głównym rozprawy doktorskiej było *przedstawienie nowych metod rekomendacji opracowanych dla serwisów ogłoszeniowych oraz wykazanie ich przewagi względem metod opisanych w literaturze*. Poniżej podsumujemy jak cel ten został zrealizowany, odnosząc się również do realizacji celów pomocniczych.

W pierwszym rozdziale przedstawiliśmy szczególne cechy serwisów ogłoszeniowych mające wpływ na dobór metod rekomendacji oraz ich ewaluację. Zaprezentowaliśmy specyfikę przedsiębiorstwa, w którym rozwiązania te zostały przez autora wdrożone. Opisaliśmy także proces wdrażania tych rozwiązań. W momencie rozpoczęcia prac w przedsiębiorstwie nie istniały modele rekomendacji dedykowane kategorii praca. Jednak wskutek wdrożonych rozwiązań w pierwszym kwartale 2023 roku, prawie **5 milionów odpowiedzi na oferty pracy** było następstwem udzielonych rekomendacji. W ten sposób autor zrealizował oczekiwania Pracodawcy względem podjętej współpracy w ramach programu „Doktorat wdrożeniowy". W kolejnych rozdziałach koncentrujemy się na wykazaniu wartości naukowej przeprowadzonych badań.

W drugim rozdziale zdefiniowaliśmy pojęcie systemów rekomendacji oraz przedstawiliśmy najważniejsze ich typy, wśród których znajdują się modele wspólnej filtracji będące obiektem badań tej rozprawy. Omówiliśmy najbardziej powszechne wyzwania związane z modelami rekomendacji. Przedstawiliśmy także metody ewaluacji stosowane w rozprawie. Do wielu zagadnień omawianych w tym rozdziale odnosimy się w rozdziałach kolejnych.

W trzecim rozdziale zrealizowaliśmy pierwszy cel pomocniczy rozprawy: *przedstawienie opublikowanego przez autora zbioru danych*. Publikacja zbioru danych pozwala innym badaczom na weryfikację raportowanych przez autora wyników badań. Wskazaliśmy liczne zalety opublikowanego zbioru danych w porównaniu z innymi zbiorami danych stosowanymi do ewaluacji modeli rekomendacyjnych ofert pracy. Do dnia 10 listopada 2023 roku zbiór ten został wyświetlony 3516 razy oraz pobrany 107 razy, co świadczy o zainteresowaniu innych badaczy.

W czwartym rozdziale zrealizowany został drugi cel pomocniczy rozprawy: *za-*

*prezentowanie sposobu doboru i ewaluacji modeli rekomendacyjnych w przypadku serwisów ogłoszeniowych oraz przedstawienie wyników takiej ewaluacji dla metod opisanych w literaturze*. Przedstawiliśmy istotne z punktu widzenia serwisów ogłoszeniowych cechy modeli rekomendacji wspólnej filtracji: dokładność, pokrycie i skalowalność. Analizowaliśmy także podobieństwo rekomendacji generowanych z różnych modeli. Dokonaliśmy ewaluacji offline pięciu metod rekomendacji. Pokazaliśmy istotną statystycznie przewagę modelu RP3Beta nad innymi modelami względem głównej metryki ewaluacji, Precision@10. Następnie przeprowadziliśmy ewaluację online z użytkownikami serwisów Pracodawcy za pomocą testów A/B. W pierwszym teście pokazaliśmy, że wysyłanie rekomendacji ofert pracy z wykorzystaniem modelu ALS zwiększa liczbę osób odpowiadających na ogłoszenia o pracę o ponad 5%. W drugim teście wykazaliśmy, że zastąpienie modelu ALS modelem RP3Beta pozwala nam uzyskać o około 20% większy wpływ na tę liczbę. Na podstawie tych wyników, model RP3Beta został wdrożony w kategorii praca w serwisach Pracodawcy. W kolejnych rozdziałach autor przedstawia własne metody inspirowane tym modelem.

W rozdziale piątym przedstawiliśmy szczegółowy opis infrastruktury pozwalającej na generowanie rekomendacji w czasie rzeczywistym. Rozwiązanie to zostało wdrożone w serwisach Pracodawcy działających na różnych rynkach. W ten sposób, w połączeniu z opisem wdrożeń w rozdziale pierwszym, zrealizowaliśmy trzeci cel pomocniczy rozprawy: *opisanie procesu skutecznego wdrożenia modeli rekomendacyjnych w serwisach ogłoszeniowych Pracodawcy*. Pokazaliśmy również sposób wykorzystania wdrożonej infrastruktury dla uzyskania rekomendacji w czasie rzeczywistym w przypadku modelu RP3Beta proponując model **RP3Beta real-time**. Pokazaliśmy, że zastąpienie modelu RP3Beta generującego rekomendacje w trybie wsadowym przez model RP3Beta generujący rekomendacje w czasie rzeczywistym, zwiększyło liczbę osób odpowiadających na rekomendowane oferty pracy o ponad 10%. Model RP3Beta real-time jest zatem jednym z modeli wspomnianych w celu głównym rozprawy. Ponadto, zaproponowana infrastruktura może być wykorzystana także dla innych modeli rekomendacji.

W rozdziale szóstym wprowadziliśmy nową, grafową metodę rekomendacji, P3 Learning to Rank (**P3LTR**) będącą uogólnieniem modelu RP3Beta. Zaproponowana metoda, w przeciwieństwie do modelu RP3Beta, posiada parametry optymalizowane podczas trenowania modelu, a także umożliwia uwzględnienie cech interakcji, użytkowników i przedmiotów. Przedstawiliśmy metodę trenowania modelu P3LTR oraz zdefiniowaliśmy odpowiednie do tego celu funkcje straty. Wymieniliśmy również liczne zalety proponowanego podejścia, między innymi wyjaśnialność rekomendacji, wydajność ich generowania, a także możliwość uzyskania rekomendacji w czasie rzeczywistym poprzez zastosowanie proponowanej wcześniej infrastruktury. Pokazaliśmy przewagę modelu P3LTR nad modelem RP3Beta pod względem metryk dokładności oraz pokrycia na

opublikowanym wcześniej zbiorze danych. Zatem model P3LTR jest drugim modelem realizującym cel główny rozprawy.

W rozdziale siódmym omówiliśmy budowę oraz wykorzystanie grafowych modeli neuronowych w systemach rekomendacyjnych. Zaproponowaliśmy podejście **P3GNN** będące, bardziej skalowalnym od modelu P3LTR, uogólnieniem modelu RP3Beta. Pokazaliśmy przewagę zaproponowanej metody nad istniejącymi grafowymi sieciami neuronowymi. Model P3GNN uzyskał porównywalne wyniki do modelu RP3Beta. W przeciwieństwie do modelu RP3Beta, proponowany model umożliwia generowanie reprezentacji wektorowej wierzchołków oraz złożony jest z wielu elementów, których poprawa może pozwolić na uzyskanie lepszych wyników rekomendacji. Zatem model P3GNN jest trzecim modelem realizującym cel główny rozprawy.

Ponadto, w rozdziale siódmym autor zaproponował nowe funkcje straty, a także metodę generowania trudniejszych przykładów negatywnych. Niestety metody te nie pozwoliły na istotną poprawę wyników modelu P3GNN. Uzyskaliśmy jednak znacznie lepsze wyniki modelu przy wykorzystaniu funkcji straty ScaledBPR niż w przypadku powszechnie stosowanej funkcji BPR będącej szczególnym przypadkiem funkcji ScaledBPR. Przy wykorzystaniu metody generowania trudniejszych przykładów negatywnych zaobserwowaliśmy dużą poprawę przy zastosowaniu funkcji straty BPR. Zatem zaproponowane rozwiązania mogą okazać się skuteczne dla zbiorów danych, na których zastosowanie funkcji straty BPR daje najlepsze wyniki.

Podsumowując, pokazaliśmy przewagę przedstawionych metod RP3Beta real-time, P3LTR oraz P3GNN względem metod opisanych w literaturze w przypadku serwisów ogłoszeniowych Pracodawcy. **Cel główny rozprawy został zatem zrealizowany**.

Obiecującym kierunkiem dalszych prac jest wzbogacenie modelu P3GNN o cechy krawędzi w celu uzyskania modelu o podobnej jakości do modelu P3LTR. Warto byłoby także zbadać możliwość poprawy jakości modelu P3GNN poprzez uwzględnienie cech użytkowników i przedmiotów, a następnie porównać z modelami hybrydowymi opisanymi w literaturze. Ponadto, z punktu widzenia wdrożeniowego, reprezentacje wektorowe użytkowników i przedmiotów generowane przez model P3GNN mogą zostać wykorzystane w celu personalizacji wyników wyszukiwarek ogłoszeń.

# Bibliografia


[1] *RecSys Challenge '16: Proceedings of the Recommender Systems Challenge*, New York, NY, USA, 2016. Association for Computing Machinery.

[2] *RecSys Challenge '17: Proceedings of the Recommender Systems Challenge 2017*, New York, NY, USA, 2017. Association for Computing Machinery.

[3] Fabian Abel, András Benczúr, Daniel Kohlsdorf, Martha Larson, Róbert Pálovics. Recsys challenge 2016: Job recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, strona 425–426, New York, NY, USA, 2016. Association for Computing Machinery.

[4] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, Daniel Kohlsdorf. Recsys challenge 2017: Offline and online evaluation. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, strona 372–373, New York, NY, USA, 2017. Association for Computing Machinery.

[5] Gediminas Adomavicius, Alexander Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[6] Gediminas Adomavicius, Alexander Tuzhilin. Context-aware recommender systems. *Recommender Systems Handbook*, strony 191–226. Springer US, Boston, MA, 2015.

[7] Charu Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, wydanie 1, 2016.

[8] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, strona 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery.

[9] Marie Al-Ghossein, Talel Abdessalem, Anthony Barré. A survey on stream-based recommender systems. *ACM Computing Surveys (CSUR)*, 54(5), 2021.

[10] Mohammad Yahya H. Al-Shamri. User profiling approaches for demographic recommender systems. *Knowledge-Based Systems*, 100:175–187, 2016.

[11] Shaha Alotaibi. A survey of job recommender systems. *International Journal of the Physical Sciences*, 7:5127–5142, 2012.

[12] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Claudio Pomo. Reenvisioning the comparison between neural collaborative filtering and matrix factorization. *Fifteenth*

*ACM Conference on Recommender Systems*, RecSys '21, strona 521–529, New York, NY, USA, 2021. Association for Computing Machinery.

[13] Bahare Askari, Jaroslaw Szlichta, Amirali Salehi-Abari. Variational autoencoders for top-k recommendation with implicit feedback. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, strona 2061–2065, New York, NY, USA, 2021. Association for Computing Machinery.

[14] Paul Baltescu, Haoyu Chen, Nikil Pancha, Andrew Zhai, Jure Leskovec, Charles Rosenberg. Itemsage: Learning product embeddings for shopping recommendations at pinterest. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, strona 2703–2711, New York, NY, USA, 2022. Association for Computing Machinery.

[15] Oren Barkan, Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, strony 1–6, 2016.

[16] James Bennett, Stan Lanning, i in. The Netflix Prize. *Proceedings of KDD Cup and Workshop*, wolumen 2007, strona 35. New York, NY, USA., 2007.

[17] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, wolumen 24. Curran Associates, Inc., 2011.

[18] Peter Boatwright, Joseph Nunes. Reducing assortment: An attribute-based approach. *Journal of Marketing – J MARKETING*, 65:50–63, 2001.

[19] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[20] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li. Learning to rank: From pairwise approach to listwise approach. *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, strona 129–136, New York, NY, USA, 2007. Association for Computing Machinery.

[21] Badrish Chandramouli, Justin J. Levandoski, Ahmed Eldawy, Mohamed F. Mokbel. Streamrec: A real-time recommender system. *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, strona 1243–1246, New York, NY, USA, 2011. Association for Computing Machinery.

[22] Hao Chen, Zefan Wang, Feiran Huang, Xiao Huang, Yue Xu, Yishi Lin, Peng He, Zhoujun Li. Generative adversarial framework for cold-start item recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, strona 2565–2571, New York, NY, USA, 2022. Association for Computing Machinery.

[23] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Trans. Inf. Syst.*, 41(3), 2023.

[24] Rui Chen, Qingyi Hua, Yan-Shuo Chang, Bo Wang, Lei Zhang, Xiangjie Kong. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE Access*, 6:64301–64320, 2018.

[25] Ting Chen, Yizhou Sun, Yue Shi, Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. *Proceedings of the 23rd ACM SIGKDD International*

*Conference on Knowledge Discovery and Data Mining*, KDD '17, strona 767–776, New York, NY, USA, 2017. Association for Computing Machinery.

[26] Alexander Chernev. When more is less and less is more: The role of ideal point availability and assortment in consumer choice. *Journal of Consumer Research*, 30:170–183, 2003.

[27] Fabian Christoffel, Bibek Paudel, Chris Newell, Abraham Bernstein. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, strona 163–170, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, Yiannis Siantos. Random walks in recommender systems: Exact computation and simulations. *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, strona 811–816, New York, NY, USA, 2014. Association for Computing Machinery.

[29] Maurizio Ferrari Dacrema. RP3beta – Github Repository. `https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation/blob/master/GraphBased/RP3betaRecommender.py`, 2021. dostęp: 2023-11-18.

[30] Maurizio Ferrari Dacrema. SLIM – Github Repository. `https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation/blob/master/SLIM_ElasticNet/SLIMElasticNetRecommender.py`, 2021. dostęp: 2023-11-18.

[31] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems*, 39:1–49, 2021.

[32] Maurizio Ferrari Dacrema, Paolo Cremonesi, Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, strona 101–109, New York, NY, USA, 2019. Association for Computing Machinery.

[33] Corné De Ruijt, Sandjai Bhulai. Job recommender systems: A review. 2021. arXiv:2111.13576.

[34] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogín, Tommaso Di Noia. Recommender systems fairness evaluation via generalized cross entropy. 2019.

[35] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[36] Yue Deng. Recommender systems based on graph embedding techniques: A review. *IEEE Access*, 10:51587–51633, 2021.

[37] Mukund Deshpande, George Karypis. Item-based top- n recommendation algorithms. *ACM Transactions on Information Systems - TOIS*, 22:143–177, 2004.

[38] Juhi Dhameliya, Nikita Desai. Job recommendation system using content and collaborative filtering based techniques. *Int J Soft Comput Eng*, 9(3):8–15, 2019.

[39] Kim Falk. *Praktyczne systemy rekomendacji*. Wydawnictwo Naukowe PWN, 2020.

[40] Chencheng Fang, Jiantong Zhang, Wei Qiu. Online classified advertising: A review and bibliometric analysis. *Scientometrics*, 113(3):1481–1511, 2017.

[41] Oleksandr Ferludin, Arno Eigenwillig, Martin Blais, Dustin Zelle, Jan Pfeifer, Alvaro Sanchez-Gonzalez, Sibon Li, Sami Abu-El-Haija, Peter Battaglia, Neslihan Bulut, Jonathan Halcrow, Filipe Miguel Gonçalves de Almeida, Silvio Lattanzi, André Linhares, Brandon Mayer, Vahab Mirrokni, John Palowitch, Mihir Paradkar, Jennifer She, Anton Tsitsulin,

Kevin Villela, Lisa Wang, David Wong, Bryan Perozzi. TF-GNN: graph neural networks in tensorflow. *CoRR*, abs/2207.03522, 2022.

[42] Matthias Fey, Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[43] Ben Frederickson. ALS – Github Repository. `https://github.com/benfred/implicit/blob/master/implicit/als.py`, 2023. dostęp: 2023-11-18.

[44] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[45] Milton Friedman. A comparison of alternative tests of significance for the problem of $m$ rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[46] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, Yong Li. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst.*, 1(1), 2023.

[47] Salvador García, Alberto Fernández, Julián Luengo, Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.

[48] Salvador García, Francisco Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(12):2677–2694, 2008.

[49] Mouzhi Ge, Francesco Ricci, David Massimo. Health-aware food recommender system. *Proceedings of the 9th ACM Conference on Recommender Systems*, strony 333–334, 2015.

[50] Xue Geng, Hanwang Zhang, Jingwen Bian, Tat-Seng Chua. Learning image and user features for recommendation in social networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, strony 4274–4282, 2015.

[51] Carlos A. Gomez-Uribe, Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), 2016.

[52] John Gourville, Dilip Soman. Overchoice and assortment type: When and why variety backfires. *Marketing Science*, 24:382–395, 2005.

[53] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, Doug Sharp. E-commerce in your inbox: Product recommendations at scale. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, strona 1809–1818, New York, NY, USA, 2015. Association for Computing Machinery.

[54] Olivier Grisel. SLIM – Github Repository. `https://github.com/Mendeley/mrec/blob/master/mrec/item_similarity/slim.py`, 2014. dostęp: 2023-11-18.

[55] Aditya Grover, Jure Leskovec. node2vec: Scalable feature learning for networks. wolumen 2016, strony 855–864, 2016.

[56] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2022.

[57] Carlos Gómez-Uribe, Neil Hunt. The Netflix recommender system. *ACM Transactions on Management Information Systems*, 6:1–19, 2015.

[58] William L. Hamilton, Rex Ying, Jure Leskovec. Inductive representation learning on large graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, strona 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[59] John T. Hancock, Taghi M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):28, 2020.

[60] F. Maxwell Harper, Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5:19:1–19:19, 2015.

[61] Jianming He, Wesley W. Chu. A social network-based recommender system (snrs). *Data Mining for Social Network Data*, strony 47–74. Springer US, Boston, MA, 2010.

[62] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, strona 639–648, New York, NY, USA, 2020. Association for Computing Machinery.

[63] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua. Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[64] Yifan Hu, Yehuda Koren, Chris Volinsky. Collaborative filtering for implicit feedback datasets. *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, strona 263–272, USA, 2008. IEEE Computer Society.

[65] Frank Hutter, Holger Hoos, Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. *Proceedings of the 31st International Conference on Machine Learning*, wolumen 32 serii *Proceedings of Machine Learning Research*, strony 754–762, Bejing, China, 2014. PMLR.

[66] Hyunwoo Hwangbo, Yang Sok Kim, Kyung Cha. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications*, 28:94–101, 2018.

[67] Ronald L. Iman, James M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics – Theory and Methods*, 9(6):571–595, 1980.

[68] Sheena Iyengar, Mark Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology*, 79:995–1006, 2001.

[69] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, Li Chen. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[70] Umair Javed, Kamran Shaukat Dar, Ibrahim Hameed, Farhat Iqbal, Talha Mahboob Alam, Suhuai Luo. A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning (iJET)*, 16:274–306, 2021.

[71] Liaoliang Jiang, Yuting Cheng, Li Yang, Jing Li, Hongyang Yan, Xiaoqin Wang. A trust-based collaborative filtering algorithm for e-commerce recommendation system. *Journal of Ambient Intelligence and Humanized Computing*, 10:3023–3034, 2019.

[72] Michael Jugovac, Dietmar Jannach, Mozhgan Karimi. Streamingrec: A framework for benchmarking stream-based news recommenders. *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, strona 269–273, New York, NY, USA, 2018. Association for Computing Machinery.

[73] Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park,

Sang-Wook Kim. Linear, or non-linear, that is the question! *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2021.

[74] Yehuda Koren, Robert Bell, Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[75] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. 2015. arXiv:1507.08439.

[76] Maciej Kula. LightFM – Github Repository. `https://github.com/lyst/lightfm`, 2023. dostęp: 2023-11-18.

[77] Saurabh Kulkarni, Sunil F. Rodd. Context aware recommendation systems: A review of the state of the art techniques. *Computer Science Review*, 37:100255, 2020.

[78] Robert Kwieciński, Agata Filipowska, Tomasz Górecki, Viacheslav Dubrov. Job recommendations: benchmarking of collaborative filtering methods for classifieds, 2023. arXiv:2301.07946.

[79] Robert Kwieciński, Tomasz Górecki, Agata Filipowska. Learning edge importance in bipartite graph-based recommendations. *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, strony 227–233, 2022.

[80] Robert Kwieciński, Grzegorz Melniczak, Tomasz Górecki. Comparison of real-time and batch job recommendations. *IEEE Access*, 11:20553–20559, 2023.

[81] Emanuel Lacic, Markus Reiter-Haas, Tomislav Duricic, Valentin Slawicek, Elisabeth Lex. Should we embed? a study on the online performance of utilizing embeddings for real-time job recommendations. RecSys '19, strona 496–500, New York, NY, USA, 2019. Association for Computing Machinery.

[82] Emanuel Lacic, Markus Reiter-Haas, Dominik Kowald, Manoj Reddy Dareddy, Junghoo Cho, Elisabeth Lex. Using autoencoders for session-based job recommendations. *User Modeling and User-Adapted Interaction*, 30(4):617–658, 2020.

[83] Asher Levi, Osnat Mokryn, Christophe Diot, Nina Taft. Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system. *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, strona 115–122, New York, NY, USA, 2012. Association for Computing Machinery.

[84] Shan Li, Baoxu Shi, Jaewon Yang, Ji Yan, Shuai Wang, Fei Chen, Qi He. Deep job understanding at linkedin. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, strona 2145–2148, New York, NY, USA, 2020. Association for Computing Machinery.

[85] Siyuan Li, Elena Karahanna. Online recommendation systems in a b2c e-commerce context: A review and future directions. *Journal of the Association for Information Systems*, 16:72–107, 2015.

[86] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022.

[87] Blerina Lika, Kostas Kolomvatsos, Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065–2073, 2014.

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

[89] Zhuang Liu, Yunpu Ma, Yuanxin Ouyang, Zhang Xiong. Contrastive learning for recommender system. 2021. arXiv:2101.01317.

[90] Fernando B Pérez Maurera, Maurizio Ferrari Dacrema, Pablo Castells, Paolo Cremonesi. Impression-aware recommender systems. 2023. arXiv:2308.07857.

[91] Mary L McHugh. The chi-square test of independence. *Biochemia Medica*, 23(2):143–149, 2013.

[92] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Proceedings of the 26th International Conference on Neural Information Processing Systems – Volume 2*, NIPS'13, strona 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[93] Ravita Mishra, Sheetal Rathi. Enhanced dssm (deep semantic structure modelling) technique for job recommendation. *Journal of King Saud University - Computer and Information Sciences*, 34(9):7790–7802, 2022.

[94] Adrien Mogenet, Tuan Anh Nguyen Pham, Masahiro Kazama, Jialin Kong. Predicting online performance of job recommender systems with offline evaluation. *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, strona 477–480, New York, NY, USA, 2019. Association for Computing Machinery.

[95] Xia Ning, George Karypis. Slim: Sparse linear methods for top-n recommender systems. *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, strona 497–506, USA, 2011. IEEE Computer Society.

[96] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[97] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of the KDD Cup and Workshop*, strony 39–42, 2007.

[98] Bibek Paudel, Fabian Christoffel, Chris Newell, Abraham Bernstein. Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Transactions on Interactive Intelligent Systems*, 7:1–34, 2016.

[99] Michael J. Pazzani, Daniel Billsus. Content-based recommendation systems. *The Adaptive Web*, strona 325–341. Springer, 2007.

[100] Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, Paolo Cremonesi. Contentwise impressions: An industrial dataset with impressions included. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, strona 3093–3100, New York, NY, USA, 2020. Association for Computing Machinery.

[101] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. Deepwalk: Online learning of social representations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[102] Luiz Pizzato, Tomek Rej, Thomas Chung, Kalina Yacef, Irena Koprinska, Judy Kay. Reciprocal recommenders. *CEUR Workshop Proceedings*, 606, 2011.

[103] Steffen Rendle. Factorization machines. *2010 IEEE International Conference on Data Mining*, strony 995–1000, 2010.

[104] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009*, strona 452–461, 2012.

[105] Stephen Robertson. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. *Journal of Documentation - J DOC*, 60:503–520, 2004.

[106] Gunnar Schröder, Maik Thiele, Wolfgang Lehner. Setting goals and choosing metrics for recommender system evaluations. *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, wolumen 23, strona 53, 2011.

[107] Guy Shani, Asela Gunawardana. Evaluating recommendation systems. *Recommender Systems Handbook*, strony 257–297. Springer US, Boston, MA, 2011.

[108] Aneesh Sharma, Jerry Jiang, Praveen Bommannavar, Brian Larson, Jimmy Lin. Graphjet: Real-time content recommendations at Twitter. *Proc. VLDB Endow.*, 9(13):1281–1292, 2016.

[109] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, Shaoping Ma. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10(5):813–831, 2019.

[110] Brent Smith, Greg Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21:12–18, 2017.

[111] Shahab Saquib Sohail, Jamshed Siddiqui, Rashid Ali. Classifications of recommender systems: A review. *Journal of Engineering Science & Technology Review*, 10(4), 2017.

[112] Yading Song, Simon Dixon, Marcus Pearce. A survey of music recommendation systems and future perspectives. *9th International Symposium on Computer Music Modeling and Retrieval*, wolumen 4, strony 395–410. Citeseer, 2012.

[113] Gábor Takács, István Pilászy, Domonkos Tikk. Applications of the conjugate gradient method for implicit feedback collaborative filtering. *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, strona 297–300, New York, NY, USA, 2011. Association for Computing Machinery.

[114] Yan-Martin Tamm, Rinchin Damdinov, Alexey Vasilev. Quality metrics in recommender systems: Do we calculate metrics consistently? *Fifteenth ACM Conference on Recommender Systems*, RecSys '21, strona 708–713, New York, NY, USA, 2021. Association for Computing Machinery.

[115] Niek Tax, Sander Bockting, Djoerd Hiemstra. A cross-benchmark comparison of 87 learning to rank methods. *Information Processing & Management*, 51(6):757–772, 2015.

[116] Bartłomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, strona 273–276, New York, NY, USA, 2016. Association for Computing Machinery.

[117] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', Yoshua Bengio. Graph attention networks. 2017. arXiv:1710.10903.

[118] Petar Veličković. Message passing all the way up, 2022. arXiv:2202.11097.

[119] M.K. Vijaymeena, K. Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3:19–28, 2016.

[120] Antônio David Viniski, Jean Paul Barddal, Alceu de Souza Britto Jr., Fabrício Enembreck,

Humberto Vinicius Aparecido de Campos. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Systems with Applications*, 176:114890, 2021.

[121] Chong Wang, David Blei. Collaborative topic modeling for recommending scientific articles. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, strony 448–456, 2011.

[122] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. 2019. arXiv:1909.01315.

[123] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, Philip S. Yu. Graph learning based recommender systems: A review. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, strony 4644–4652. International Joint Conferences on Artificial Intelligence Organization, 2021. Survey Track.

[124] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, strona 525–534, New York, NY, USA, 2018. Association for Computing Machinery.

[125] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua. Neural graph collaborative filtering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, strona 165–174, New York, NY, USA, 2019. Association for Computing Machinery.

[126] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua. Neural graph collaborative filtering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, strona 165–174, New York, NY, USA, 2019. Association for Computing Machinery.

[127] Yusen Wang, Kaize Shi, Zhendong Niu. A session-based job recommendation system combining area knowledge and interest graph neural networks. *SEKE*, strony 489–492, 2020.

[128] Zhenyi Wang, Huan Zhao, Chuan Shi. Profiling the design space for graph neural networks based collaborative filtering. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, strona 1109–1119, New York, NY, USA, 2022. Association for Computing Machinery.

[129] Jason Weston, Samy Bengio, Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. *Twenty-Second International Joint Conference on Artificial Intelligence*, strona 2764–2770, 2011.

[130] Jason Weston, Hector Yee, Ron J. Weiss. Learning to rank recommendations with the k-order statistic loss. *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, strona 245–248, New York, NY, USA, 2013. Association for Computing Machinery.

[131] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[132] Jibang Wu, Renqin Cai, Hongning Wang. Déjà vu: A contextualized temporal attention

mechanism for sequential recommendation. *Proceedings of The Web Conference 2020*, WWW '20, strona 2199–2209, New York, NY, USA, 2020. Association for Computing Machinery.

[133] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), 2022.

[134] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, Jian Lu. Ice-breaking: Mitigating cold-start recommendation problem by rating comparison. *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, strona 3981–3987. AAAI Press, 2015.

[135] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, strona 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[136] Guoqing Zhu, Naga Kopalle, Yongzhen Wang, Xiaozhong Liu, Kemi Jona, Katy Börner. Community-based data integration of course and job data in support of personalized career-education recommendations. *Proceedings of the Association for Information Science and Technology*, 57, 2020.

[137] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, Jingren Zhou. Aligraph: A comprehensive graph neural network platform. *Proc. VLDB Endow.*, 12(12):2094–2105, 2019.

[138] Dávid Zibriczky. A combination of simple models by forward predictor selection for job recommendation. *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery.

[139] Radim Řehůřek. Word2Vec – Github Repository. `https://github.com/RaRe-Technologies/gensim`, 2023. dostęp: 2023-11-18.