Politechnika Wrocławska

Wydział Informatyki i Zarządzania

kierunek studiów: Informatyka

specjalność: Projektowanie Systemów Informatycznych

Praca dyplomowa - magisterska

Hybrydowy algorytm rekomendacji wykorzystujący metody filtrowania z analizą zawartości i filtrowania kolaboratywnego

Hybrid recommender algorithm based on content-based filtering and collaborative filtering

Katarzyna Biernat

słowa kluczowe:
KEYWORDS

krótkie streszczenie:
SHORT ABSTRACT

Promotor:	dr inż. Bernadetta Maleszka	.....	.....
	imię i nazwisko	*ocena*	*podpis*

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław 2016

Niniejszy dokument został złożony w systemie LaTeX.

Spis treści

Streszczenie

ABSTRACT PL

Abstract

ABSTRACT EN

Todo list

Rozdział 1

Wstęp

Wraz z rozwojem Internetu zmienił się sposób dostępu do informacji. Kiedyś to użytkownik musiał walczyć pozyskanie wiedzy; dzisiaj to informacje walczą u uwagę użytkowników. W świecie zalanym wiadomościami koniecznym wydaje się być zastosowanie filtra, który odsieje interesującą i wartościową zawartość od tej niechcianej. Pomocne okazują się zautomatyzowane mechanizmy rekomendacji.

Jednakże sama idea rekomendacji nie jest niczym nowym. Co więcej, zjawisko to możemy zaobserwować w naturze – na przykład wśród mrówek, które podążają wyznaczoną (rekomendowaną) ścieżką feromonową w poszukiwaniu pożywienia.

Ludzie od niepamiętnych czasów posiłkowali się opiniami innych, aby ułatwić sobie dokonanie wyboru, od najbliższego grona znajomych do ekspertów i autorytetów.

Wraz z rozwojem nauk informatycznych problem rekomendacji stał się problemem interesującym badaczy. Za pierwszy system rekomendacji uznaje się *Tapestry* stworzony w laboratoriach Xerox Palo Alto Research Center w 1992 roku. Motywacją było odfiltrowanie rosnącej liczby niechcianej poczty elektronicznej [21].

Wkrótce później idea ta została rozszerzona przez takich graczy jak Amazon, Google, Pandora, Netflix, Youtube, Yahoo etc. aż do formy, jaką znamy dzisiaj: systemu, który sugeruje użytkownikom produkty, filmy, muzykę, strony internetowe na podstawie ich aktywności w sieci [58].

Wielkie koncerny internetowe stale poprawiają jakość swoich algorytmów rekomendacji. Najlepszym przykładem jest tutaj Netflix, który w październiku 2006 zorganizował ogólnodostępny konkurs na najlepszy algorytm. Zadaniem uczestników było ulepszenie algorytmu Cinematch. Już po siedmiu dniach od ogłoszenia konkursu trzy zespoły zdołały poprawić wynik Cinematch o 1.06% [41, 43]. 18 września 2009 Netflix ogłosił, że zespół BellKor's Pragmatic Chaos poprawił Cinematch o 10,06% osiągając wynik $RMSE = 0.8567$. Tym samym wygrał nagrodę w wysokości $1,000,000 i zakończył konkurs [42, 44].

Systemy rekomendacji ulepszane są nieustannie, o czym świadczy chociażby organizowana rokrocznie konferencja *ACM International Conference on Recommender Systems*. Tematyka ta poruszana jest także na konferencjach *European Conference on Information Retrieval*, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* i wielu innych. Mimo dużego stopnia

zaawansowania wciąż istnieje pole manewru do ulepszania algorytmów rekomendacji i co za tym idzie zwiększanie zadowolenia użytkowników, które z kolei prowadzi do osiągania korzyści biznesowych.

Celem pracy jest opracowanie i zbudowanie hybrydowego algorytmu rekomendacji. Składowymi docelowego algorytmu są metody kolaboratywnego filtrowania oraz metody filtrowania z analizą zawartości.

(PÓŹNIEJ) CEL PRACY: trzeba rozszerzyć opis, w szczególności o to, co jest nowego i czym różni się od innych już istniejących algorytmów

W kolejnych rozdziałach przedstawione są istniejące oraz proponowane ulepszenia, które pozwalają na jeszcze lepsze dopasowanie wyników rekomendacji do oczekiwań użytkowników.

(PÓŹNIEJ) streszczenie spisu treści

Rozdział 2

Przegląd istniejących rozwiązań

W przeciągu ostatnich lat algorytmy rekomendacji stały się nieodłącznym elementem wielu popularnych serwisów internetowych z różnych domen. Są wykorzystywane m.in. na portalach muzycznych w celu podpowiadania kolejnych piosenek oraz na witrynach poświęconych tematyce filmowej, w celu sugerowania kolejnych interesujących produkcji kinowych. Odnalazły one szerokie zastosowanie w branży e-commerce, gdzie poprzez podpowiadanie klientom kolejnych produktów zwiększana jest sprzedaż i generowane są wyższe zyski. Dzięki tego typu rozwiązaniom możliwe jest indywidualne dopasowanie oferty sklepu do osobistych preferencji każdego użytkownika.

Poniżej zaprezentowany jest przegląd kilku wybranych popularnych serwisów, które korzystają z algorytmów rekomendacji w celu zwiększenia swojej atrakcyjności.

2.0.1. Rekomendacja muzyki

- **YouTube** – serwis powstały w 2005 roku, pozwalający na bezpłatne umieszczanie, odtwarzanie, ocenianie i komentowanie filmów. Od 2006 roku przejęty przez Google. YouTube buduje profil użytkownika w oparciu o jego aktywność w serwisie. Brane pod uwagę są polubienia, subskrypcje, udostępnianie a także informacje czy użytkownik obejrzał film do końca czy tylko pewien jego procent. Techniki rekomendacji stosowane przez serwis to przede wszystkim wydobywanie reguł asocjacyjnych i licznik wspólnych odwiedzin danego wideo w czasie trwania pojedynczej sesji [12].
- **LastFM** – internetowa radiostacja oferująca rozbudowany mechanizm rekomendacji piosenek "Audioscrobbler".
- **Pandora** – spersonalizowane radio internetowe wykorzystujące projekt Music Genome Project. Każda piosenka przeanalizowana jest pod kątem maksymalnie 450 cech; na tej podstawie budowane są rekomendacje [1].

2.0.2. Rekomendacja filmów

- **Netflix** – amerykańska platforma oferująca strumieniowanie filmów i seriali. Działający od 2007 roku gigant oferuje rozbudowany system rekomendacji Cinematch [48].

- **Filmweb** – polski serwis poświęcony filmom i jednocześnie druga największa baza filmowa na świecie. Oferuje system rekomendacji Gustomierz, który umożliwia poznawanie nowych filmów w guście użytkownika [14].
- **Internet Movie Database (IMDb)** – największa internetowa baza filmów. Baza zawiera $3,837,014$ pozycji, które są oceniane w skali od 1 do 10 przez użytkowników [24].

2.0.3. Platformy typu e-commerce

- **Allegro** – polski portal aukcyjny. Swoim użytkownikom oferuje panel rekomendacji. Prezentowane produkty wybierane są w oparciu o to, co dotychczas kupował i oglądał użytkownik [3].
- **Amazon** – największy na świecie sklep internetowy typu B2C. Amazon w swoich mechanizmach rekomendacji wykorzystuje algorytmy filtrowania kolaboratywnego typu element–element [35].

2.1. Techniki rekomendacji

Tradycyjnie wyróżniamy następujące techniki rekomendacji:

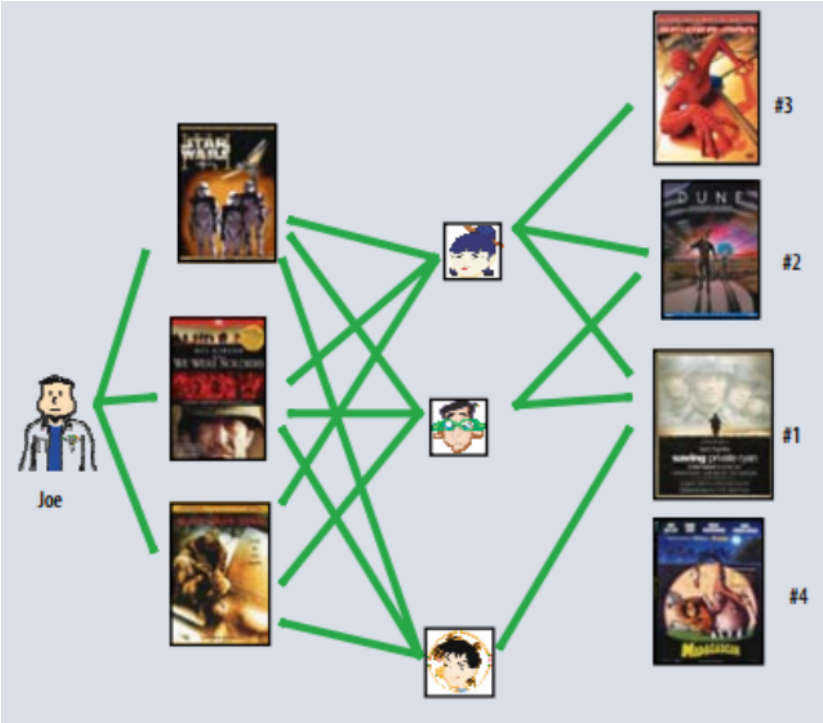
- **filtrowanie z analizą zawartości** (ang. *content–based filtering*), technika koncentrująca się na atrybutach elementów. Użytkownikowi rekomendowane są elementy, które podobne są do tych wybieranych przez niego w przeszłości;
- **filtrowanie kolaboratywne** (ang. *collaborative filtering*), technika polegająca na odnajdywaniu użytkowników o podobnych gustach i sugerowaniu lubianych przez nich elementów aktualnie aktywnemu użytkownikowi;
- **filtrowanie demograficzne** (ang. *demographic filtering*), technika koncentrująca się na sugerowaniu aktywnemu użytkownikowi elementów popularnych pośród użytkowników z tej samej okolicy bądź w podobnym przedziale wiekowym;
- **filtrowanie z analizą domeny wiedzy** (ang. *knowledge-based filtering*), technika dobierająca kolejne elementy na podstawie określonej domeny wiedzy na temat tego, jak dany element spełnia potrzeby i preferencje użytkownika;
- **filtrowanie z analizą społecznościową** (ang. *community-based filtering*), technika dobierająca rekomendacje dla użytkownika w zależności od preferencji innych użytkowników z jego sieci społecznościowej. W myśl zasady „powiedz mi kim są twoi przyjaciele a powiem ci kim jesteś";
- **hybrydowe systemy rekomendacji** to kombinacja dowolnych powyższych technik.

Każda z tych technik ma swoje pozytywne i negatywne strony w zależności od kontekstu, w którym ma być stosowana [50]. W systemie prezentowanym w tej pracy wykorzystane zostały algorytmy filtrowania kolaboratywnego, filtrowania z analizą zawartości i systemy hybrydowe. Omówienie koncepcji ich działania oraz ich zalet i wad znajduje się kolejno w sekcjach 2.2, 2.3 i 2.4. Dalsze rozdziały zawierają szczegóły dotyczące implementacji tych rozwiązań.

2.2. Filtrowanie kolaboratywne

Filtrowanie kolaboratywne opiera się na założeniu, że ludzie o zbliżonym guście dokonują podobnych wyborów. Użytkownicy o zbliżonym guście to osoby, które oceniły konkretne elementy w podobny sposób [50, 57, 22].

W przypadku filtrowania kolaboratywnego można wyróżnić dwa główne podejścia: oparte o regułę sąsiedztwa (ang. *neighborhood*) oraz oparte o model (ang. *model–based*), wykorzystujące modele ukrytych parametrów [30, 31].

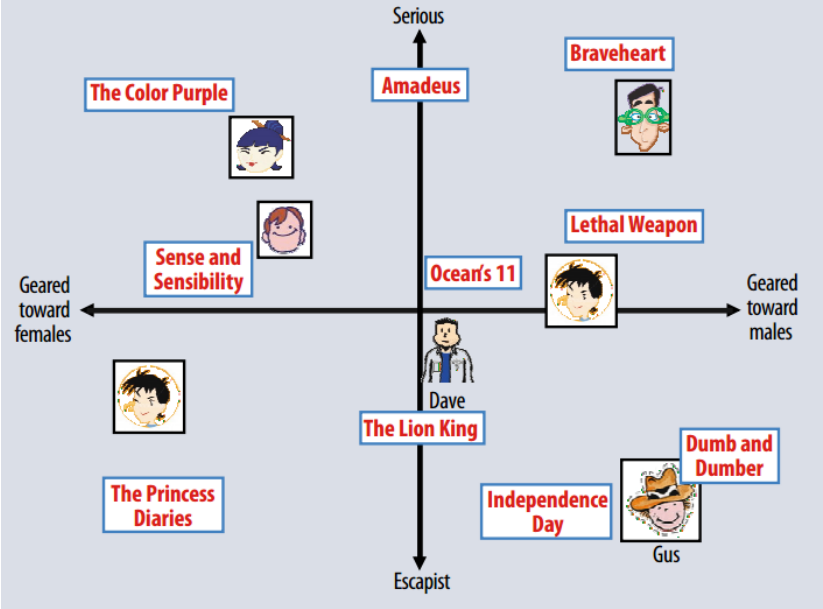


Rys. 2.1: Filtrowanie kolaboratywne metodą sąsiedztwa, zorientowane na użytkownika [31].

Filtrowanie w oparciu o regułę sąsiedztwa koncentruje się na związkach element–element bądź użytkownik–użytkownik [30]. Rysunek 2.1 pokazuje regułę sąsiedztwa skoncentrowaną na relacji użytkownik-użytkownik. Joe ocenił trzy filmy. System odnajduje innych użytkowników, którzy ocenili te trzy pozycje podobnie jak Joe. Każdy z nich pozytywnie ocenił film „Saving Private Ryan", zatem jest to pierwsza rekomendacja dla Joe.

Ideą podejścia w oparciu o model jest zbadanie i modelowanie zależności element–użytkownik wraz z czynnikami reprezentującymi ukryte własności elementów i użytkowników. Taki model jest następnie uczony przy użyciu dostępnych danych. W rezultacie można z niego odczytać przewidywaną ocenę elementu dla konkretnego użytkownika [13, 30].

Rysunek 2.2 pokazuje w sposób uproszczony podejście oparte o model. W układzie współrzędnym oznaczeni są użytkownicy wedle swoich preferencji oraz konkretnych cech (np. płeć) a także filmy, które stanowią odpowiedź na dany zestaw preferencji/cech [31].



Rys. 2.2: Filtrowanie kolaboratywne z wykorzystaniem modelu ukrytych parametrów [31].

2.2.1. Zalety filtrowania kolaboratywnego

Filtrowanie kolaboratywne dobrze radzi sobie nawet w takich wypadkach, gdzie elementy nie są dobrze opisane lub nie zawierają żadnych własności. Niektóre systemy zawierają zawartość trudną do przeanalizowania przez komputer, np. opinie, pomysły, komentarze. Mimo tego, dzięki filtrowaniu kolaboratywnemu i tak można zbudować efektywny system rekomendacji [38].

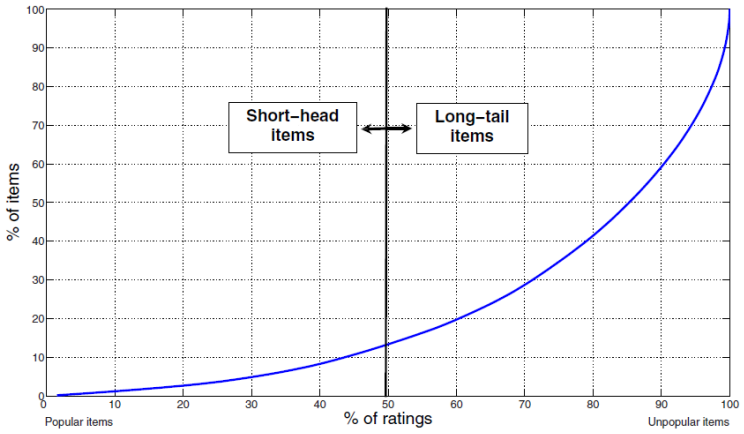
Dzięki zastosowaniu tej metody mniej prawdopodobne jest wystąpienia zjawiska zwanego "bańką informacyjną" (ang. *filter bubble*). Polega ono na tym, że użytkownik otrzymując informacje wyselekcjonowane przez określony algorytm zostaje zamknięty w pewnego rodzaju „bańce", tzn. sugerowane mu elementy są do siebie podobne, gdyż dobierane są w oparciu o jego przeszłe zainteresowania [46]. Dzięki filtrowaniu kolaboratywnemu użytkownik może otrzymać zaskakujące rekomendacje i (w zależności od domeny) odkryć nowy gatunek muzyczny, nowego reżysera lub nową kategorię produktów w sklepie.

2.2.2. Wady filtrowania kolaboratywnego

Jednym z problemów klasycznego podejścia do kolaboratywnego filtrowania jest brak uwzględnienia dynamiki zmian w gustach użytkowników. Ten sam użytkownik na przestrzeni kilku lat lub miesięcy może zupełnie inaczej ocenić ten sam film bądź piosenkę. Rozwiązaniem jest dodanie czynnika czasu podczas obliczania wag kolejnych ocen. [10, 25, 31].

Innym problemem jest tzw. zimny start (ang. *cold start*). Polega on na tym, że nowi użytkownicy w systemie ocenili zbyt mało elementów, aby można było zbudować dla nich dobre rekomendacje [54, 62].

Powszechnym zjawiskiem jest tzw. efekt długiego ogona. Rysunek 2.3 przedstawia jak rozkłada się procentowa ilość ocen danych elementów w zależności od ich popularności. Jeżeli algorytm rekomendacji nie wspiera mniej popularnych elementów, to istnieje ryzyko, że użytkownicy nie otrzymają możliwości eksplorowania nowych, niszowych materiałów [9, 54].



Rys. 2.3: Problem długiego ogona: 50% ocen dotyczy 10-12% najpopularniejszych elementów w systemie [54].

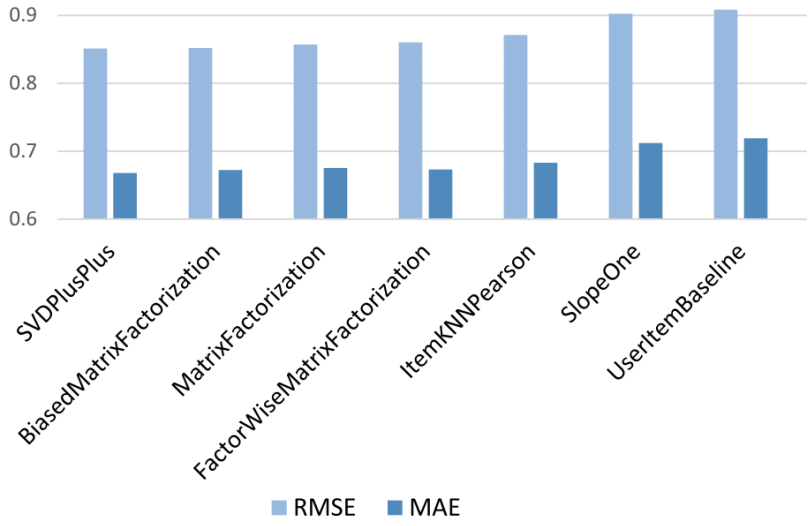
Systemy rekomendacji wykorzystujące filtrowanie kolaboratywne nie są skalowalne. Złożoność rośnie proporcjonalnie do ilości użytkowników i elementów. Wielkie koncerny internetowe takie jak Twitter wykorzystają klastry i maszyny z bardzo dużą pamięcią aby zachować płynność działania serwisu [17].

2.2.3. Algorytmy filtrowania kolaboratywnego

Istnieje wiele implementacji algorytmów filtrowania kolaboratywnego. Jedną z bibliotek, która takie algorytmy implementuje i która została wykorzystana w tym projekcie jako baza do algorytmów hybrydowych jest MyMediaLite [15, 16]. Zawiera ona zestaw metod opartych zarówno o model jak też o regułę sąsiedztwa:

- SVD++ [28]
- Biased Matrix Factorization [55, 49]
- Matrix Factorization
- Factor-Wise Matrix Factorization [7]
- Slope One [33]
- User-Item Baseline [29]

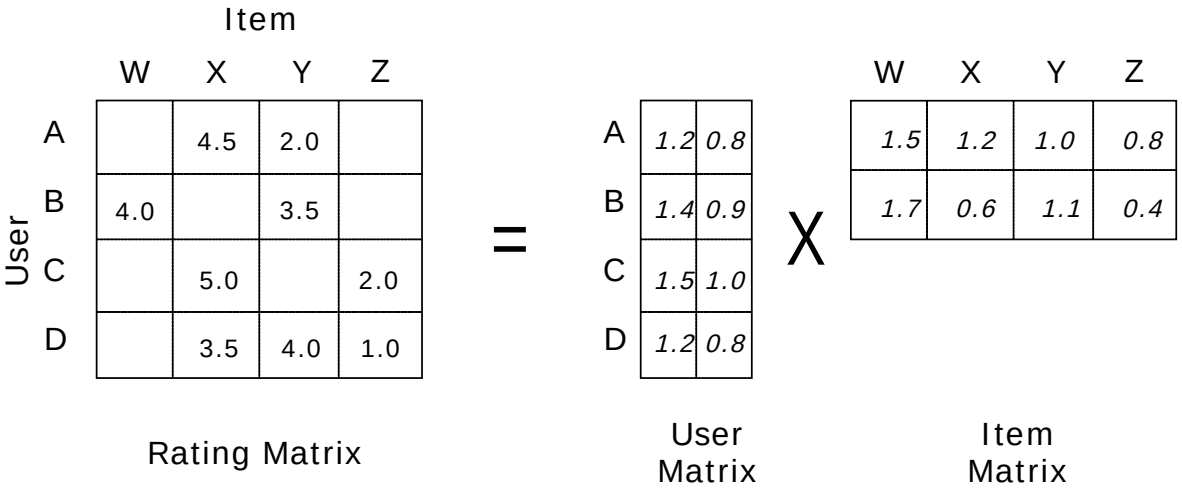
Rys. 2.4 przedstawia wyniki działania wymienionych algorytmów (im mniejsza wartość RMSE i MAE tym lepiej). Testy przeprowadzone zostały na bazie MovieLens M1 z pięciokrotną walidacją krzyżową [18]. Najefektywniejsze okazały się algorytmy SVD++, Biased Matrix Factorization i Matrix Factorization działające w oparciu o model.



Rys. 2.4: Test algorytmów filtrowania kolaboratywnego [40]

Matrix Factorization

Algorytm *Matrix Factorization* bazuje na matematycznej metodzie rozkładu macierzy na czynniki.



Rys. 2.5: Faktoryzacja macierzy [53]

Początkowo dana jest niekompletna macierz zawierająca oceny, jakie użytkownicy wystawili konkretnym elementom (*Rating Matrix*). Celem metody jest odnalezienie wartości, jakie można wstawić w puste miejsca, czyli przewidzenie jaką ocenę dany użytkownik wystawi nieocenionemu jeszcze elementowi.

W tym celu tworzone są odrębne macierze dla użytkowników i elementów zawierające ukryte własności. Każdy element powiązany jest z wektorem $q_i \in \mathbb{R}^f$ (wektory W, X, Y, Z na rys. 2.5) a każdy użytkownik z wektorem $p_u \in \mathbb{R}^f$ (wektory A, B, C, D na rys. 2.5). Wartości czynników ukrytych determinują stopień zainteresowania daną cechą (w przypadku macierzy użytkowników) bądź stopień, w jakim dany element posiada tę cechę (w przypadku macierzy elementów).

Iloczyn skalarny $q_i^T p_u$ przedstawia relację pomiędzy użytkownikiem a elementem. Na tej podstawie można wnioskować ocenę r_{ui}, jaką użytkownik może wystawić elementowi: $r_{ui} = q_i^T p_u$ i w konsekwencji estymować zainteresowanie użytkownika danym elementem [31].

Głównym wyzwaniem jest odnalezienie wartości macierzy użytkownika i elementu, które po przemnożeniu przez siebie dadzą kompletną macierz ocen. W przypadku omawianych algorytmów stosowana jest metoda stochastycznego gradientu prostego.

Stochastyczny gradient prosty

Stochastyczny gradient prosty (ang. *stochastic gradient descent*, *SGD*) jest iteracyjnym algorytmem optymalizacyjnym mającym za zadanie odnalezienie minimum bądź maksimum zadanej funkcji. Jest on uproszczeniem popularnej metody gradientu prostego [8].

Dana jest funkcja celu w postaci

$$Q(w) = \sum_{i=1}^{n} Q_i(w)\,. \tag{2.1}$$

Zadaniem algorytmu jest odnalezienie takiej wartości parametru w, dla którego $Q(w)$ będzie minimalne. Wykorzystując klasyczną metodę gradientu prostego poszukiwania można zapisać następującym wzorem:

$$w := w - \eta \sum_{i=1}^{n} \nabla Q_i(w)\,, \tag{2.2}$$

gdzie η symbolizuje współczynnik uczenia (ang. *learning rate*). W przypadku algorytmu stochastycznego następuje uproszczenie. Zamiast obliczać dokładny gradient $Q(w)$, w każdej iteracji jest on aproksymowany na podstawie pojedynczego, losowo wybranego przypadku:

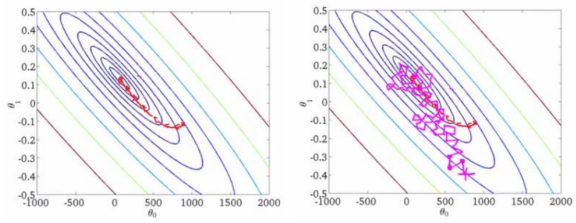
$$w := w - \eta \nabla Q_i(w)\,. \tag{2.3}$$

Algorytm przetwarza wszystkie elementy zbioru treningowego i dla każdego przypadku wykonuje aktualizację wartości w. Przebieg algorytmu przedstawiony jest poniżej (algorytm 1).

Algorytm 1 Stochastyczny gradient prosty

```
w ← początkowy wektor w
η ← współczynnik uczenia
while nie odnaleziono minimum do
  losowo mieszkaj przykłady ze zbioru testowego
  for i ∈ {1, 2, ..., n} do
    w := w − η∇Q_i(w)
  end for
end while
```

Stochastyczny gradient prosty wykonuje dużo więcej kroków niż jego klasyczny odpowiednik aby odnaleźć rozwiązanie optymalne. Mimo tego jest on szybszy, gdyż każdy pojedynczy krok jest mniej kosztowny niż w oryginale. Różnicę w działaniu przedstawia rys. 2.6.



Rys. 2.6: Różnica pomiędzy klasycznym gradientem prostym (po lewej) a jego stochastyczną wersją (po prawej) [5]

Przebieg algorytmu Matrix Factorization

Przebieg algorytmu składa się z dwóch faz. W pierwszej fazie inicjowany jest model (zob. algorytm 2). Daną wejściową jest macierz zawierająca dotychczasowe oceny elementów przez użytkowników w systemie. Na wyjściu otrzymywane są dwie nowe macierze reprezentujące ukryte własności użytkowników i elementów. Na tym etapie są one wypełnione wartościami losowymi w celu ułatwienia kolejnych obliczeń. W następnych etapach algorytmu wartości te zostaną zamienione tak, że reprezentować będą rzeczywistą relację pomiędzy elementem i użytkownikiem.

Algorytm 2 Matrix Factorization – Inicjacja modelu

$N \leftarrow$ liczba użytkowników
$M \leftarrow$ liczba elementów
$F \leftarrow$ liczba ukrytych własności
$ratings \leftarrow$ Macierz NxM zawierająca dotychczasowe oceny wszystkich elementów przez wszystkich użytkowników
$user\_factors \leftarrow$ Macierz NxF reprezentująca ukryte własności użytkowników
$item\_factors \leftarrow$ Macierz MxF reprezentująca ukryte własności elementów
for each $uf \in user\_factors$, $if \in item\_factors$ **do**
  wstaw losową wartość za pomocą transformacji Boxa–Mullera
end for
for each $user, item \in ratings$ **do**
  if $ratings_{user,item} = NULL$ **then**
    wstaw 0 do wiersza $user\_factors_{user}$ i $item\_factors_{item}$
  end if
end for
return $user\_factors, item\_factors$

W fazie drugiej następuje uczenie metodą stochastycznego gradientu prostego (zob. algorytm 3). Wynikiem tej fazy są macierze reprezentujące ukryte własności użytkowników i elementów. Mnożąc je ze sobą uzyskiwana jest przewidywana ocena każdego z elementów przez użytkowników.

Przed rozpoczęciem uczenia ustalane są parametry: parametr regulujący (ang. *regularization*), współczynnik uczenia (ang. *learning rate*), parametr zanikania (ang. *decay*) i liczba iteracji. W trakcie trwania głównej pętli parametr regulujący pozostaje niezmienny. Służy on zapobieganiu zjawisku nadmiernego dopasowania (ang. *overfitting*). Tempo uczenia jest przy każdym przebiegu pętli mnożone przez parametr zanikania,

dzięki czemu można kontrolować w jakim stopniu kolejne przebiegi pętli wpływają na finalny wynik.

Ostatnim parametrem ustalanym przed główną pętlą jest skośność globalna (ang. *global bias*), która jest średnią wszystkich znanych ocen.

W pętli uczenia wykonywane są następujące operacje: dla każdej pary użytkownik – element budowana jest przewidywana ocena poprzez obliczenie iloczynu skalarnego odpowiednich wartości z macierzy wartości ukrytych. Ocena ta jest modyfikowana poprzez dodanie globalnej skośności a następnie porównywana z faktyczną oceną elementu przez użytkownika. Tak uzyskany błąd służy do wyliczenia delty. Macierze wartości ukrytych uaktualniane są o wyliczoną deltę.

Pod koniec każdej iteracji aktualizowany jest współczynnik uczenia.

Algorytm 3 Matrix Factorization – Faza uczenia

$global\_bias \leftarrow$ średnia wszystkich ocen
$X \leftarrow$ liczba iteracji
$regularization \leftarrow$ parametr regulujący
$current\_learnrate \leftarrow$ współczynnik uczenia
$decay \leftarrow$ parametr zanikania
for each $x \in X$ **do**
  for each $user, item \in ratings$ **do**
    $predicion = global\_bias + IloczynSkalarny(user\_factors_{user}, item\_factors_{item})$;
    $error = ratings_{user,item} - prediction$
    //dopasowanie własności ukrytych:
    for each $f \in F$ **do**
      $delta_u = error * item\_factors_{item,f} - regularization * user\_factors_{user,f}$
      $delta_i = error * user\_factors_{user,f} - regularization * item\_factors_{item,f}$
      $user\_factors_{user,f}$ += $current\_learnrate * delta_u$
      $item\_factors_{item,f}$ += $current\_learnrate * delta_i$
    end for
  end for
  $current\_learnrate$ *= $decay$
end for
return $user\_factors, item\_factors$

Biased Matrix Factorization

Algorytm *Biased Matrix Factorization* jest modyfikacją wyżej opisanego algorytmu *Matrix Factorization*. Podobnie jak jego pierwowzór składa się z dwóch faz. W pierwszej fazie dodatkowo inicjowane są dwa dodatkowe wektory: skośność użytkowników (ang. *user bias*) i skośność elementów (ang. *item bias*).

Inaczej jest też obliczana skośność globalna:

$$global\_bias = \frac{\frac{a - r_{min}}{r_{max} - r_{min}}}{1 - \frac{a - r_{min}}{r_{max} - r_{min}}}, \tag{2.4}$$

gdzie

a to średnia wszystkich ocen

r_{min} to minimalna ocena w systemie

r_{max} to maksymalna ocena w systemie

Faza druga wygląda podobnie jak w przypadku algorytmu *Matrix Factorization*, jednak są uwzględniane dodatkowe parametry i wykonywane dodatkowe kroki.

Algorytm 4 Biased Matrix Factorization – Faza uczenia

$global\_bias \leftarrow$ średnia wszystkich ocen
$X \leftarrow$ liczba iteracji
$regU, regI, BiasReg \leftarrow$ parametry regulujące dla użytkownika, elementu i ogólny
$current\_learnrate \leftarrow$ współczynnik uczenia
$BiasLearnRate \leftarrow$ współczynnik uczenia skośności
$decay \leftarrow$ parametr zanikania
for each $x \in X$ **do**
  for each $user, item \in ratings$ **do**
    $score = global\_bias + user\_bias_{user} + item\_bias_{item} + IloczynSkalarny(user\_factors_{user}, item\_fact$
    $sig\_score = \frac{1}{1+\exp(-score)}$
    $predicion = rating_{min} + sig\_score + (rating_{max} - rating_{min})$
    $error = ratings_{user,item} - prediction$
    $gradient\_common = err * sig\_score * (1 - sig\_score) * (rating_{max} - rating_{min})$
    //dopasowanie skośności:
    $user\_bias_{user} \mathrel{+}= BiasLearnRate * current\_learnrate * (gradient\_common - BiasReg * RegU * user\_bias_{user})$
    $item\_bias_{item} \mathrel{+}= BiasLearnRate * current\_learnrate * (gradient\_common - BiasReg * RegI * item\_bias_{item})$
    //dopasowanie własności ukrytych:
    for each $f \in F$ **do**
      $delta_u = gradient\_common * item\_factors_{item,f} - RegU * user\_factors_{user,f}$
      $delta_i = gradient\_common * user\_factors_{user,f} - RegI * item\_factors_{item,f}$
      $user\_factors_{user,f} \mathrel{+}= current\_learnrate * delta_u$
      $item\_factors_{item,f} \mathrel{+}= current\_learnrate * delta_i$
    end for
  end for
  $current\_learnrate \mathrel{*}= decay$
end for
return $user\_factors, item\_factors$

SVD++

SVD++ jest rozszerzeniem metody SVD (dekompozycja głównych składowych, ang. *singular value decomposition*). Od poprzednich omawianych algorytmów różni go przede wszystkim to, że korzysta nie tylko z bezpośredniej informacji zwrotnej do tworzenia profilu użytkownika ale także z pośredniej (zob. 2.3.3).

Model SVD++ opisywany jest równaniem:

$$r_{ui} = \mu + b_u + b_i + q_i^T(p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j) \tag{2.5}$$

Użytkownik powiązany jest z wektorem $p_u \in \mathbb{R}^f$ reprezentującym zainteresowanie konkretnymi cechami. Taki model uzupełniany jest sumą $\frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j$, która reprezentuje pośrednie informacje zwrotne. Przykładem informacji niejawnej jest fakt, że użytkownik w ogóle zareagował na dany element (np. ocenił go), bez względu na wynik tej interakcji.

Zmienne b_u i b_i reprezentują obserwowane odchylenie od średniej dla użytkowników i elementów, natomiast μ to średnia wszystkich ocen [28].

2.3. Filtrowanie z analizą zawartości

Filtrowanie z analizą zawartości opiera się na cechach elementów w systemie. Rekomendowane są obiekty, które podobne są do tych pozytywnie ocenionych wcześniej przez użytkownika [22]. W zależności od domeny pod uwagę mogą być brane słowa kluczowe, cechy takie jak rok wydania, reżyser, autor, kompozytor, gatunek itp.

2.3.1. Zalety filtrowania z analizą zawartości

Do zalet filtrowania z analizą zawartości należy niezależność użytkownika. Podczas budowania rekomendacji brany pod uwagę jest tylko jego profil, aktywność innych aktorów w systemie nie wpływa na wynik końcowy. Nawet, gdy w bazie będzie istniało niewielu użytkowników lub ocenią oni zupełnie inne elementy niż użytkownik aktywny, to rekomendacja w dalszym ciągu może być miarodajna.

Inną przewagą jest przejrzystość – każda propozycja jest w pełni uzasadniona, gdyż opiera się na działaniach użytkownika w przeszłości. Ponadto, tego typu algorytm ma możliwość zaproponowania elementu, który nie był nigdy wcześniej oceniany przez nikogo. Zapobiega to zjawisku długiego ogona [36].

2.3.2. Wady filtrowania z analizą zawartości

Aby rekomendacja była skuteczna użytkownik powinien ocenić jak najwięcej elementów. Problematyczni są zatem użytkownicy, którzy dopiero co dołączyli do serwisu oraz tacy, którzy nie są aktywni i rzadko zostawiają po sobie ślad [37].

Filtrowanie z analizą zawartości jest podatne na pułapkę tzw. bańki informacyjnej. Jeżeli w systemie rekomendującym produkcje kinowe użytkownik do tej pory oceniał jedynie filmy akcji, to mało prawdopodobne jest, że algorytm zaproponuje mu ciekawy dramat obyczajowy. Nowe propozycje nie są zaskakujące [36].

2.3.3. Metody tworzenia profilu użytkownika

Profil użytkownika może być tworzony na dwa sposoby. Jeżeli użytkownik jawnie pozostawia informacje można mówić bezpośredniej informacji zwrotnej (ang. *explicit feedback*). Do takich informacji należą: ocena konkretnych elementów, przycisk lubienia lub nielubienia (tzw. „łapka" w górę lub w dół), komentarz itp.

Gdy użytkownik nie zostawia po sobie tego typu śladów, to informacje na temat jego zachowań i preferencji można uzyskać korzystając z pośredniej informacji zwrotnej (ang. *implicit feedback*). System bierze wówczas pod uwagę aktywność użytkownika taką jak: historia zakupów, historia przeglądarki a nawet ruchy myszką. W przypadku

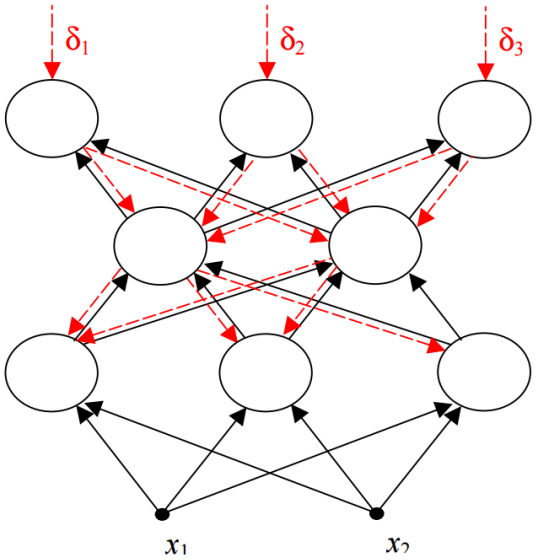
serwisu z muzyką czy filmem pośrednią informacją zwrotną będzie fakt, czy użytkownik wysłuchał lub obejrzał dany materiał do końca czy też wyłączył go po paru sekundach [30, 36].

2.3.4. Algorytmy wykorzystywane w implementacji filtrowania z analizą zawartości

Najprostszym sposobem implementacji filtrowania z analizą zawartości jest wyciąganie średnich wartości cech z wektora ocenianego elementu. Bardziej efektywne jest jednak wykorzystanie technik uczenia maszynowego takich jak naiwny klasyfikator bayesowski, klasteryzacja, drzewa decyzyjne lub sztuczne sieci neuronowe. W niniejszej pracy wykorzystana została sztuczna sieć neuronowa uczona algorytmami opisanymi poniżej.

Propagacja wsteczna

Algorytm propagacji wstecznej jest jedną z popularniejszych metod uczenia nadzorowanego jednokierunkowych sieci neuronowych. Zbiór uczący składa się z danych wejściowych (wektory cech elementu) i z informacji o oczekiwanym wyniku (ocena elementu). Różnica między wynikiem zwróconym przez sieć a wartością oczekiwaną stanowi miarę błędu sieci neuronowej.



Rys. 2.7: Algorytm propagacji wstecznej w sieci trójwarstwowej – idea działania [32]

Wiadomo, że funkcja celu jest funkcją ciągłą. Bazując na gradientowych metodach optymalizacji, wagi w sieci neuronowej aktualizowane są w następujący sposób:

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n)\,, \tag{2.6}$$

$$\Delta W_{ij}(n) = \eta p(W)\,, \tag{2.7}$$

gdzie

η to współczynnik uczenia (ang. *learning rate*)
$p(W)$ to kierunek w przestrzeni wielowymiarowej W

Aby wyznaczyć kierunek $p(W)$) dla wszystkich warstw sieci należy przejść przez kolejne etapy uczenia [19, 20, 32, 45, 59].

1. W kroku pierwszym sieć neuronowa poddawana jest analizie o zwykłym kierunku przepływu sygnałów. Wynikiem są wartości sygnałów wychodzących z neuronów warstw wyjściowej i ukrytych oraz pochodne funkcji aktywacji w kolejnych warstwach.
2. W kroku drugim kierunek przepływu sygnałów zostaje odwrócony (stąd nazwa propagacja wsteczna). Funkcje aktywacji zostają zastąpione przez swoje pochodne. Na oryginalne wyjście sieci (aktualnie wejście) podana zostaje wartość równa różnicy pomiędzy wynikiem oczekiwanym a wynikiem zwróconym przez sieć.
3. Obliczana zostaje wartość różnic wstecznych.
4. W kolejnym kroku następuje wreszcie proces adaptacji wag. Odbywa się on zarówno dla sieci zwykłej jak i dla sieci o propagacji wstecznej. Reguła modyfikacji ma postać:

$$\Delta W_{ij} = \eta \sum_{\text{wzroce}} \delta_{\text{wyjście}} \cdot x_{\text{wejście}} \,, \tag{2.8}$$

5. Powyższe kroki potarzane są dla każdej pary dane wejściowe – oczekiwany wynik tak długo, aż poziom błędu spadnie poniżej akceptowalnej wartości bądź osiągnięta zostanie maksymalna liczba iteracji.

Algorytm propagacji wstecznej obrazuje schemat blokowy 2.8.

Współczynnik bezwładności

W celu zwiększenia efektywności działania algorytmu został on zmieniony poprzez dołączenie członu zwanego współczynnikiem bezwładności (momentum). Standardowy sposób aktualizacji wag sieci (równanie 2.6) został zmodyfikowany w następujący sposób:

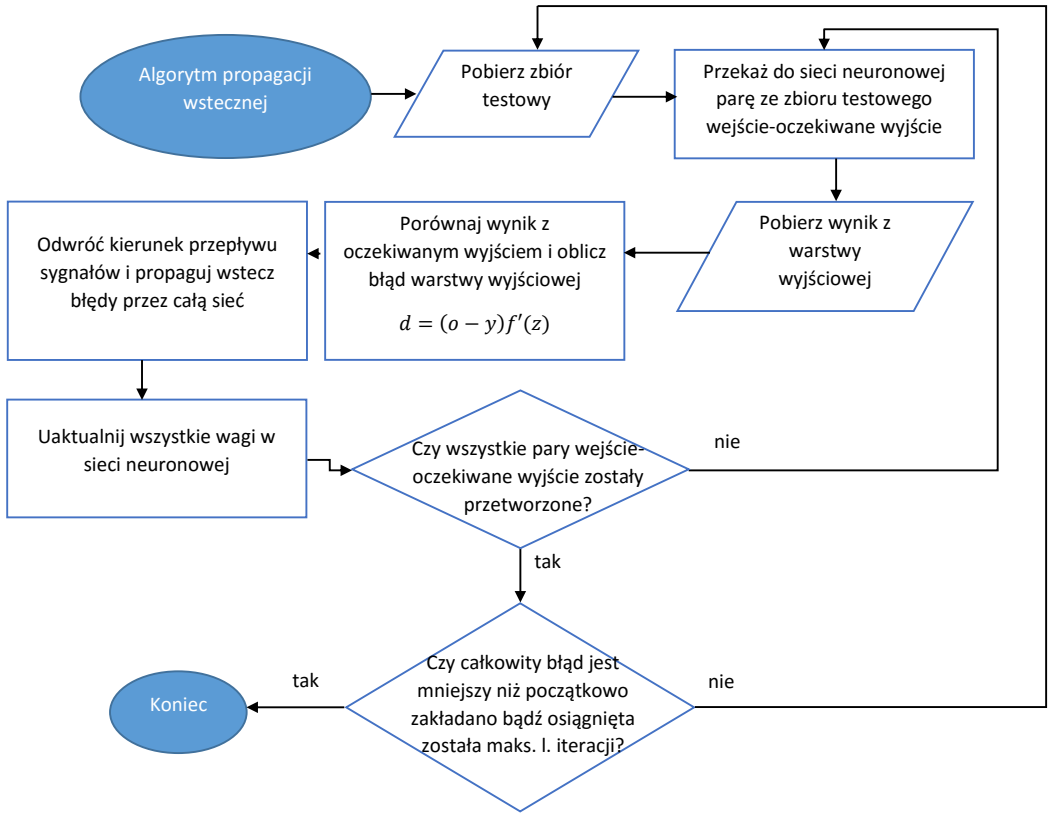
$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n) \,, \tag{2.9}$$

$$\Delta W_{ij}(n) = \eta(n) \cdot p(n) + \alpha(W_{ij}(n) - W_{ij}(n-1)) \,, \tag{2.10}$$

gdzie α jest współczynnikiem bezwładności z zakresu $[0, 1]$. Im wyższa wartość współczynnika tym większy jego wpływ na ostateczny kształt kolejnych wag. W dalszej części pracy przeprowadzono eksperymenty mające na celu dobranie optymalnej wartości współczynnika bezwładności.

Algorytm RPROP

Pomimo dużej popularności algorytmu propagacji wstecznej nie jest on pozbawiony wad. Sporym problemem jest jego relatywnie niska szybkość działania, szczególnie w przypadku bardzo dużych sieci neuronowych – a więc w przypadku odpowiadającemu



Rys. 2.8: Algorytm propagacji wstecznej

potrzebom systemów rekomendacji. W odpowiedzi na te problemy Martin Riedmiller i Heinrich Braun zaproponowali w 1992 roku alternatywny algorytm – Resilient Backpropagation (RPROP).

Główną różnicą jest wykorzystywanie jedynie informacji o dodatniości lub ujemności każdej składowej gradientu zamiast o ich wartości (jak to się odbywa w oryginalnym algorytmie). Ponadto, współczynnik uczenia modyfikowany jest w każdym kolejnym kroku.

Modyfikacja współczynników odbywa się zgodnie ze wzorem:

$$\eta(t) = \begin{cases} min\{a\eta(t-1), \eta_{max}\}, & \text{gdy } \frac{\partial E^2(t)}{\partial v(t)} \frac{\partial E^2(t-1)}{\partial v(t-1)} > 0 \\ max\{b\eta(t-1), \eta_{min}\}, & \text{gdy } \frac{\partial E^2(t)}{\partial v(t)} \frac{\partial E^2(t-1)}{\partial v(t-1)} < 0 \\ \eta(n-1), & \text{w każdym innym przypadku} \end{cases}, \tag{2.11}$$

gdzie $\frac{\partial E^2(t)}{\partial v(t)}$ jest dokładną wartością składowej v gradientu. Ponadto stałe a, b, η_{min} i η_{max} są zdefiniowane następująco:

$a = 1.2$

$b = 0.5$

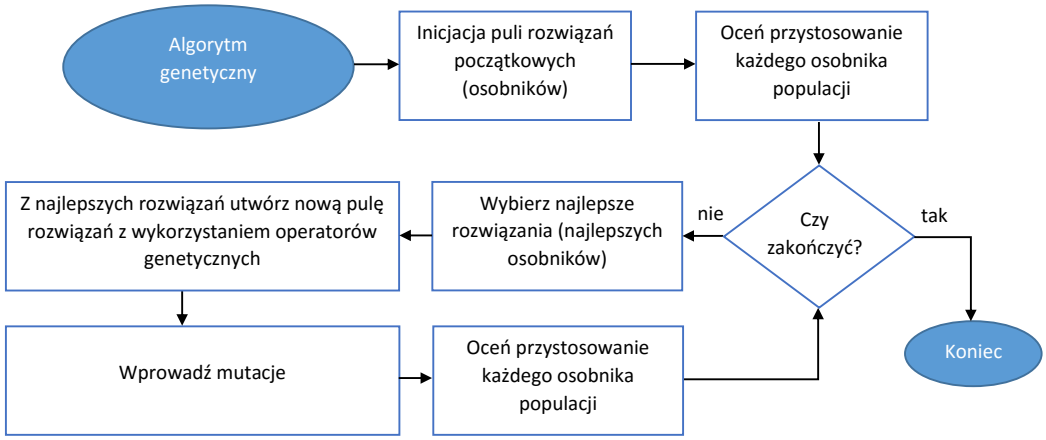
$\eta_{min} = 10^{-6}$

$\eta_{max} = 50$

Algorytm RPROP jest efektywniejszy pod kątem prędkości działania względem tradycyjnej propagacji wstecznej, więc dobrze sprawdza się tam, gdzie szybkość jest ważniejsza od wysokiej poprawności wyników [51, 52].

Algorytm genetyczny

Koncepcja algorytmów genetycznych została zaproponowana już 1960 roku przez Johna Hollanda. Należą one do klasy algorytmów ewolucyjnych, inspirowanych zasadą doboru naturalnego Darwina. Analogią do środowiska naturalnego jest pewien problem, dla którego poszukiwane jest optymalne rozwiązanie. Populacja składa się z potencjalnych rozwiązań, które są oceniane funkcją przystosowania. Zgodnie z zasadą Darwina zwycięża najsilniejszy, czyli najlepsze (chociaż nie zawsze optymalne) rozwiązanie [47]. Schemat działania algorytmów genetycznych przedstawia rys. 2.9.



Rys. 2.9: Ogólny schemat algorytmu genetycznego

Jako, że algorytmy genetyczne należą do grupy algorytmów optymalizacyjnych mogą zostać wykorzystane do uczenia sieci neuronowych. W przypadku tej implementacji populację stanowią propozycje wag dla każdego perceptronu. Preferowane są takie zestawy wag, dla których sieć zwraca rezultaty obarczone najmniejszym błędem. Najlepsze zestawy są ze sobą krzyżowane oraz podlegają mutacji. W wyniku takich mechanizmów ewolucyjnych powstaje rozwiązanie, dla których sieć neuronowa zwraca oczekiwane rezultaty [27, 39].

2.4. Algorytmy hybrydowe

W wielu przypadkach podejście hybrydowe okazuje się bardziej skuteczne niż samo filtrowanie z analizą zawartości lub kolaboratywne. Algorytmy hybrydowe można budować na wiele różnych sposobów. Jednym z nich jest odrębna implementacja filtrowania z analizą zawartości i filtrowania kolaboratywnego a następnie łączenie ich wyników. Łączyć wyniki można poprzez liniową kombinację ocen bądź też schemat głosowania [11]. Alternatywą jest wybór algorytmu decydującego na podstawie zadanych metryk

jakości, na przykład ilości cech danego elementu bądź kompletności macierzy ocen użytkownik-element.

Innym sposobem jest wykorzystanie charakterystyki filtrowania kolaboratywnego w podejściu z analizą zawartości. Pozwala ono na redukcję wymiarowości. Alternatywą jest podejście odwrotne, tzn wykorzystanie charakterystyki filtrowania z analizą zawartości w podejściu kolaboratywnym. Uwzględnienie profilu użytkownika i charakterystyki elementów pozwala na pokonanie problemów związanych ze zbyt rzadkim wypełnieniem macierzy ocen użytkownik-element [2].

Kolejnym spotykanym podejściem jest konstruowanie ogólnego, jednolitego modelu, który zawiera charakterystyki zarówno filtrowania z analizą zawartości jak i kolaboratywnego. Przykładowo [6] proponuje wykorzystanie charakterystyki opartej o zawartość i kolaboratywnej w pojedynczym klasyfikatorze regułowym.

Rozdział 3

Algorytmy

W celu ulepszenia wyników dostarczanych przez algorytmy filtrowania kolaboratywnego: Matrix Factorization, Biased Matrix Factorization i SVD++ stworzony został nowy algorytm filtrowania z analizą zawartości. W wyniku połączenia obu tych metod powstał algorytm hybrydowy. Ten rozdział zawiera model systemu, opis działania i implementacji opracowanego algorytmu filtrowania z analizą zawartości oraz opis sposobu działania zaproponowanego algorytmu hybrydowego.

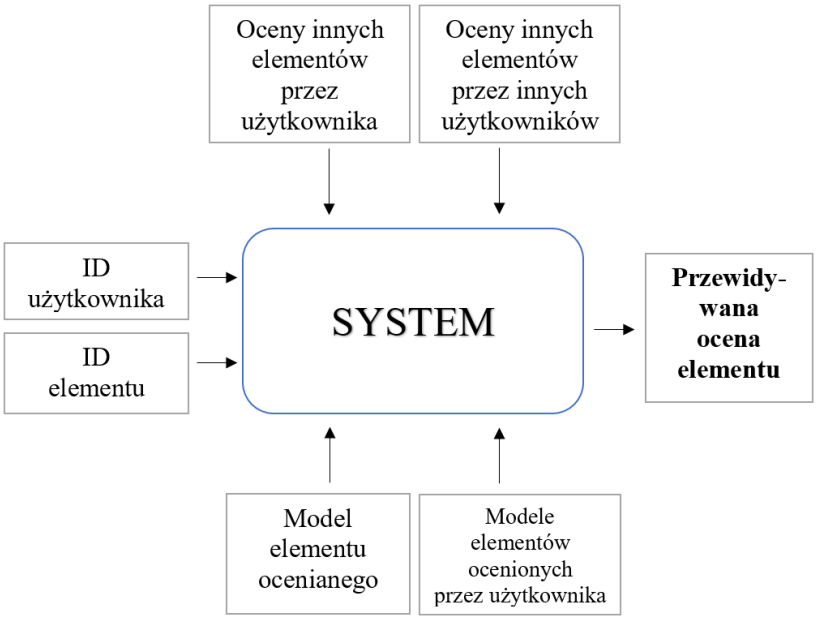
3.1. Model systemu

Głównym założeniem systemu zaproponowanego przez autorkę jest połączenie zalet kolaboratywnego filtrowania i filtrowania w oparciu zawartość minimalizując jednocześnie wady obu podejść.

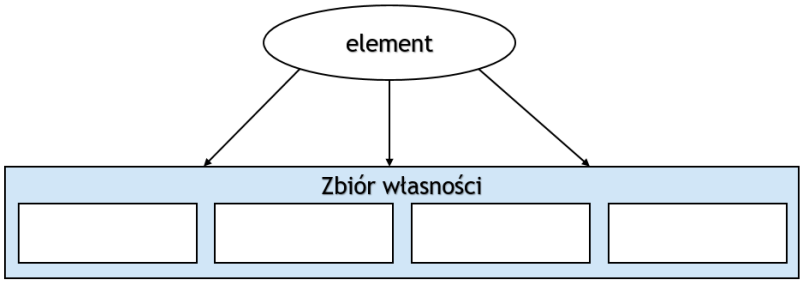
Rys. 3.1 przedstawia czarnoskrzynkowy model systemu. Danymi wejściowymi są numery identyfikacyjne użytkownika, dla którego ma być zbudowana rekomendacja oraz elementu, dla którego ma być przewidziana ocena. System pobiera model elementu a także modele wszystkich innych elementów, które użytkownik ocenił w przyszłości. Jednocześnie pobierane są informacje o tym, jak użytkownicy systemu ocenili inne elementy. Wynikiem wyjściowym jest predykcja – jak aktywny użytkownik oceni element.

Założeniem systemu jest uniwersalność, zatem model elementu jest uogólniony i dostosowuje się w zależności do domeny, w której system jest wykorzystywany. Rys. 3.2 przedstawia reprezentację elementu w systemie. W zbiorze wartości mogą znaleźć się takie pozycje jak lista aktorów, reżyser (w przypadku filmów), gatunek, wykonawca (w przypadku muzyki), typ produktu lub cena (w przypadku systemów typu e-commerce).

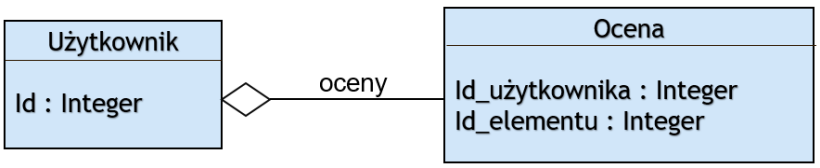
Każdy użytkownik systemu jest anonimowy. Nie jest znana jego płeć, wiek, pochodzenie itp. System nie przechowuje także informacji właściwych mediom społecznościowym, takich jak relacje między użytkownikami (przyjaźnie, śledzenie). Wiadomym jest jedynie, jakie elementy zostały ocenione i jak zostały ocenione. Rys. 3.3 przedstawia reprezentację użytkownika w systemie.



Rys. 3.1: Model czarnoskrzynkowy



Rys. 3.2: Uogólniony model elementu



Rys. 3.3: Uogólniony model elementu

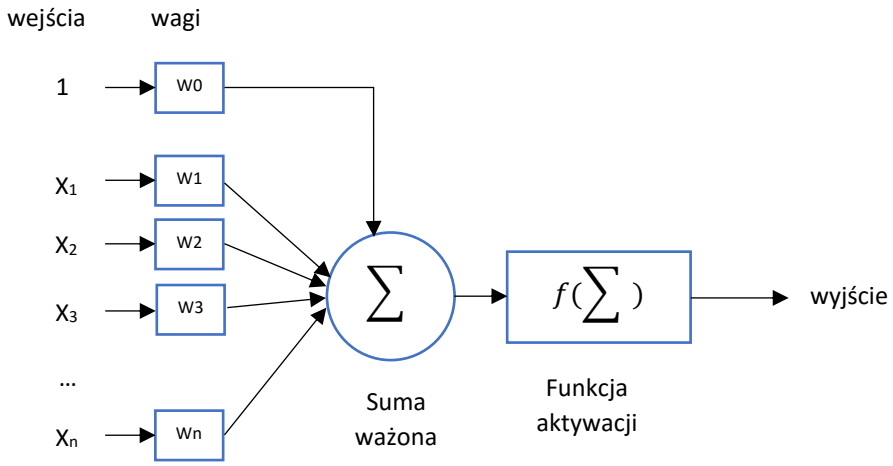
3.2. Propozycja algorytmu filtrowania z analizą zawartości

Algorytmy filtrowania z analizą zawartości budują rekomendację na podstawie ocen, jakie zostały dotychczas wystawione przez użytkownika. Analizowane są cechy elementów i ich wartości oraz określana jest ich siła wpływu na finalną ocenę.

W tym celu dla każdego użytkownika tworzona jest sieć neuronowa, która uczy się jego preferencji. W projekcie wykorzystana została implementacja sieci neuronowych z biblioteki AForge.NET Framework [26].

3.2.1. Struktura perceptronów i funkcja aktywacji

Wykorzystywana sieć neuronowa składa się z trzech warstw neuronów (perceptronów). W każdej warstwie wszystkie neurony mają konstrukcję na jak rys. 3.4



Rys. 3.4: Schemat perceptronu

Do neuronu przekazywany jest zestaw wartości w postaci wektora x. Następnie obliczana jest suma ważona tych wartości w zależności od nadanych wag w. Proces dobierania odpowiednich wag jest nazywany uczeniem (zob. 3.2.3). W następnym kroku suma ważona przekazywana jest do funkcji aktywacji neuronu. Jeżeli funkcja przyjmie wartość wyższą lub równą niż określony próg aktywacji, to perceptron zostanie pobudzony (zwróci wartość 1). Proces ten obrazuje równanie 3.1.

$$N(x_1, x_2, x_3, \ldots, x_n) = \begin{cases} 1 & \text{jeśli } f(w_0 + \sum_{i=1}^{n} w_i x_i) \geqslant \eta \\ 0 & \text{jeśli } f(w_0 + \sum_{i=1}^{n} w_i x_i) < \eta \end{cases}, \tag{3.1}$$

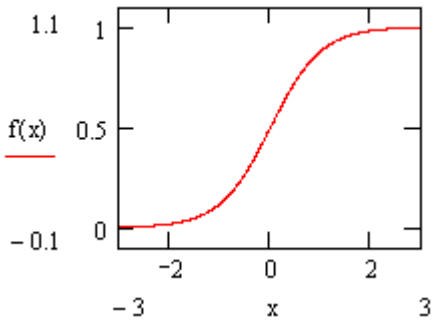
gdzie

- w to wagi kolejnych wejść
- x to wartości przekazywane do wejść
- $f(u)$ to funkcja aktywacji neuronu
- η to próg aktywacji neuronu

Na potrzeby algorytmu rekomendacji zdecydowano się przyjąć sigmoidalną unipolarną funkcję aktywacji neuronu (równanie 3.2). Funkcja przyjmuje wartości z zakresu $[0, 1]$.

$$f(x) = \frac{1}{1 + \exp(-\alpha x)}. \tag{3.2}$$

Wykres funkcji wygląda jak na rys. 3.5.



Rys. 3.5: Wykres sigmoidalnej funkcji aktywacji perceptronu [26]

3.2.2. Struktura sieci i przebieg algorytmu

Pierwszym etapem algorytmu jest analiza cech elementów ocenionych przez użytkownika. Tworzona jest lista wszystkich występujących cech które powtarzają się minimum tyle razy, ile wynosi wartość parametru *minimumRepeatingFeatures*.

Następnie inicjowana jest sieć neuronowa. Ilość neuronów warstwy wejściowej jest równa ilości wyodrębnionych cech. Warstwa ukryta zawiera tyle neuronów ile jest to określone parametrem *hiddenLayerNeurons*. Warstwa wyjściowa składa się z tylko jednego neuronu (rys. 3.6).

W kolejnym etapie dla każdego elementu tworzona jest mapa cech. Jeżeli element zawiera daną cechę o danej wartości przypisywana jest wartość 1. W przeciwnym razie wstawiane jest 0. Rys. 3.7 przedstawia przykładową mapę cech. Tak przygotowana lista przekazywana jest do sieci neuronowej.

Na wyjściu sieć zwraca przewidywaną ocenę elementu.

3.2.3. Uczenie sieci neuronowej

Aby sieć zwracała jak najlepsze wyniki musi wcześniej zostać nauczona preferencji użytkownika. Uczenie sieci polega na odnalezieniu odpowiednich wag dla każdego wejścia każdego perceptronu sieci (budowa perceptronu zob. 3.2.1). Proces ten można zapisać w postaci

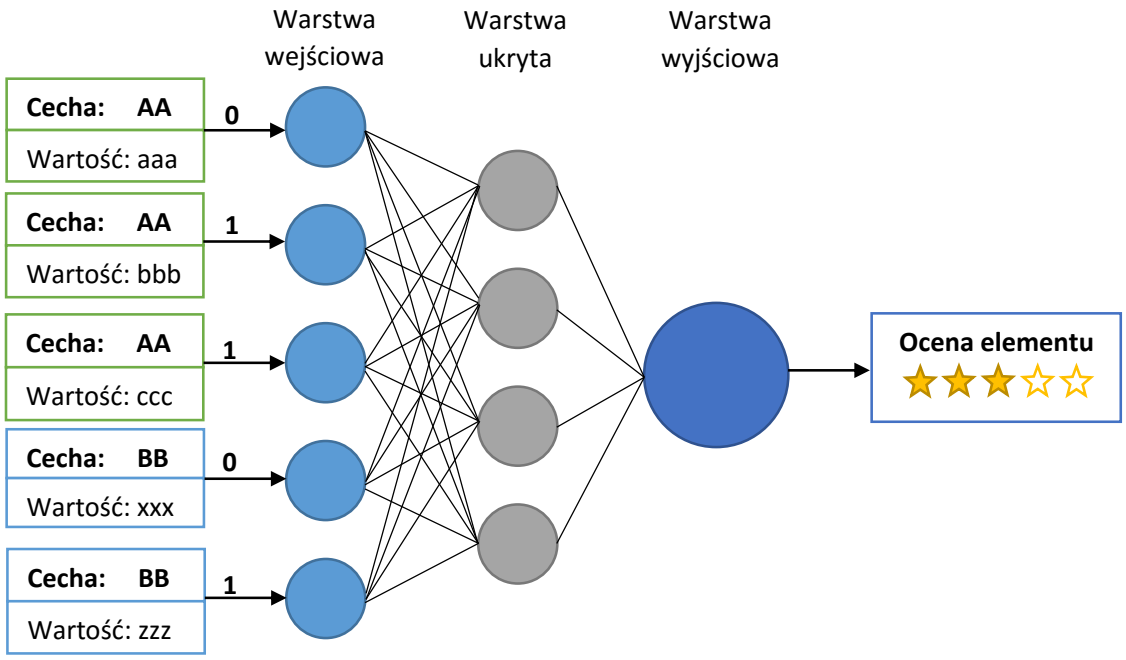
$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n)\,, \tag{3.3}$$

gdzie

$W_{ij}(n)$	to poprzednie wagi wejść
$W_{ij}(n+1)$	to nowe wagi wejść
n	numer cyklu uczącego

Dostrajanie wartości wag można wykonać na wiele sposobów. Można wyróżnić uczenie nadzorowane (z nauczycielem), uczenie z krytykiem (ang. *reinforcement learning*) i uczenie samoorganizujące się (bez nadzoru) [45].

Na potrzeby systemu rekomendacji z analizą zawartości opisywanego w tej pracy wykorzystane zostały trzy metody uczenia: algorytm propagacji wstecznej, algorytm



Rys. 3.6: Schemat sieci neuronowej

Cecha	Wartość	Czy zawiera?
Aktor	Julia Roberts	1
Aktor	Al Pacino	1
Aktor	Brad Pitt	0
...		
Reżyser	Francis Ford Coppola	1
Reżyser	Darren Aronofsky	0
...		

Rys. 3.7: Mapa cech elementu. Wiadomo, że element X zawiera cechę „Aktor" o wartościach „Julia Roberts, Al Pacino" oraz cechę „Reżyser" o wartości „Francis Ford Coppola". Element nie zawiera cechy „Aktor" o wartości „Brad Pitt" ani cechy „Reżyser" o wartości „Darren Aronofsky" więc w te miejsca wstawiane jest 0.

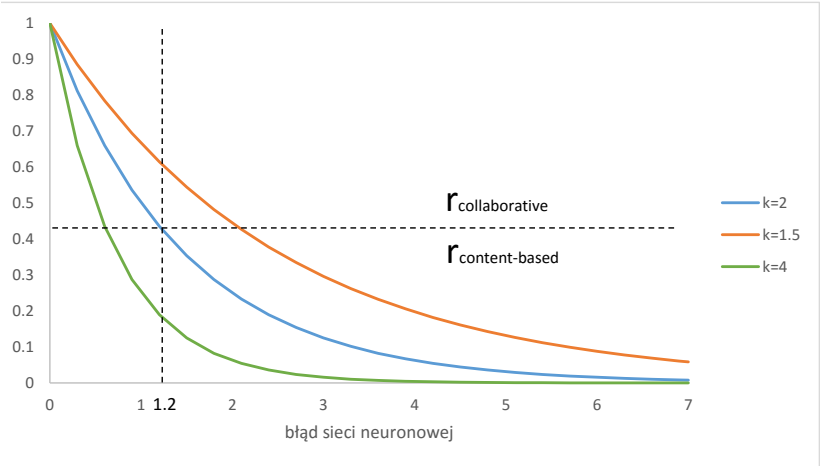
RPROP i algorytm genetyczny (zob. 2.3.4). Wymienione metody należą do kategorii uczenia nadzorowanego.

3.3. Propozycja nowych algorytmów hybrydowych

Zaproponowane nowe algorytmy są hybrydą filtrowania kolaboratywnego i filtrowania z analizą zawartości. Obie te metody uruchamiane są osobno, w efekcie dla każdej pary użytkownik–element uzyskiwane są dwie predykowane oceny z obu metod: $r_{content-based_{ui}}$ i $r_{collaborative_{ui}}$. Ponadto, znany jest błąd $d \geqslant 0$, z jakim zakończyło się uczenie sieci neuronowej dla użytkownika u. Przewidywana ocena obliczana jest wzorem:

$$r_{ui} = (\frac{1}{k})^d \cdot r_{content-based_{ui}} + (1 - (\frac{1}{k})^d) \cdot r_{collaborative_{ui}}\,, \tag{3.4}$$

gdzie k jest parametrem konfigurowalnym (domyślnie $k = 2$), który determinuje stopień nachylenia funkcji wykładniczej. Parametr musi spełniać warunek $k > 1$. Im wyższa wartość k, tym większy wpływ na rezultat ma wynik algorytmu kolaboratywnego. Zależność tę obrazuje rys. 3.8.



Rys. 3.8: Wpływ parametru k na wynik algorytmu. Jeżeli parametr $k = 2$ a uczenie sieci neuronowej zakończyło się z błędem równym $d = 1,2$, to końcowy stosunek wag wyniku filtrowania z analizą zawartości do wyniku filtrowania kolaboratywnego wyniesie ok. 43,5% do 56,5%.

Zaimplementowanych zostało 9 algorytmów łączących różne metody filtrowania kolaboratywnego i filtrowania z analizą zawartości:

1. **Matrix Factorization i sieć neuronowa uczona metodą propagacji wstecznej** — ten wariant łączy ze sobą szybki, ale najmniej efektywny ze wszystkich wykorzystywanych metod kolaboratywnych algorytm z dość dokładnym, lecz nie najszybszym sposobem uczenia sieci neuronowej. Spodziewanym efektem jest dość stosunkowo wysoki wpływ algorytmu z analizą zawartości na końcowy wynik.

2. **Matrix Factorization i sieć neuronowa uczona metodą RPROP** — jako, że zarówno metoda RPROP jak i Matrix Factorization są najszybszymi algorytmami wśród innych zaprezentowanych propozycji, to takie połączenie dostarcza najszybsze, lecz najmniej efektywne rozwiązanie.
3. **Matrix Factorization i sieć neuronowa uczona algorytmem genetycznym** — sieć neuronowa uczona algorytmem genetycznym cechuje się dość wysoką efektywnością, jednakże czas uczenia jest dużo większy w porównaniu do innych metod. Spodziewanym efektem jest stosunkowo wysoki wpływ algorytmu z analizą zawartości na końcowy wynik przy dość wolnym czasie wykonania.
4. **Biased Matrix Factorization i sieć neuronowa uczona metodą propagacji wstecznej** — algorytm Biased Matrix Factorization cechuje się wyższą skutecznością niż Matrix Factorization ale też dłuższym czasem wykonania. W związku z tym taki wariant będzie wolniejszy ale bardziej efektywny niż wariant nr 1.
5. **Biased Matrix Factorization i sieć neuronowa uczona metodą RPROP** — analogicznie jak powyżej, taki wariant powinien cechować się wyższą skuteczności i niższą prędkością niż wariant 2. Z drugiej strony będzie on szybszy i mniej skuteczny niż wariant 4.
6. **Biased Matrix Factorization i sieć neuronowa uczona algorytmem genetycznym** — z powodu wykorzystania algorytmu genetycznego będzie to jeden z wolniejszych wariantów, ale cechować się będzie dobrą efektywnością.
7. **SVD++ i sieć neuronowa uczona metodą propagacji wstecznej** — algorytm SVD++ jest najwolniejszą metodą filtrowania kolaboratywnego ale jednocześnie najefektywniejszą. Połączenie z siecią neuronową uczoną metodą propagacji wstecznej może dać najlepsze wyniki ze wszystkich wariantów.
8. **SVD++ i sieć neuronowa uczona metodą RPROP** — w tym wypadku długi czas wykonania algorytmu SVD++ może zbilansować szybki czas algorytmu RPROP. Uzyskany wynik możenie być tak dobry jak w wariancie 7 z powodu niższej efektywności RPROP niż klasycznej propagacji wstecznej.
9. **SVD++ i sieć neuronowa uczona algorytmem genetycznym** — jest to najwolniejszy wariant ze wszystkich lecz może konkurować efektywnością z wariantem 7 (zależnie od konfiguracji algorytmu genetycznego).

3.3.1. Zalety zaproponowanych metod

Największą przewagą zaproponowanych algorytmów jest to, że łączą on zalety filtrowania z analizą zawartości i filtrowania kolaboratywnego minimalizując jednocześnie ich wady poprzez dynamiczne dobieranie proporcji ich wpływu na końcowy wynik. Nowa metoda cechuje się uniwersalnością, gdyż model elementu zawierający jego własności budowany jest automatycznie. Jednocześnie nie są wymagane informacje o użytkowniku inne, niż to jakie elementy ocenił i jak je ocenił.

Im więcej zostanie dostarczonych informacji na temat elementów, tym lepsze będą wyniki. Jednakże w przypadku braku takowych informacji metoda ciągle będzie działać, wówczas wynik będzie opierał się głównie na rezultatach filtrowania kolaboratywnego.

Analogicznie, w przypadku małej liczby powiązań z innymi użytkownikami, zaproponowany algorytm hybrydowy ulepszy wynik filtrowania kolaboratywnego dzięki czynnikowi filtrowania z analizą zawartości.

W zależności od tego, czy bardziej istotna jest dokładność czy czas wykonania, można dobrać odpowiedni wariant nowej metody.

W zależności od konstrukcji danych można tak dopasować parametr k, aby wynik był dokładniejszy. W przypadku takiej konfiguracji danych, gdzie macierz użytkownik-element jest wypełniona w małym stopniu, istnieje możliwość takiego dostrojenia parametru, żeby większy wpływ na wynik końcowy miało filtrowanie z analizą zawartości. W przypadku odwrotnym preferowanym jest aby większy wpływ na wynik końcowy miał algorytm filtrowania kolaboratywnego.

3.3.2. Wady zaproponowanych metod

Największą wadą nowej metody jest wyższa złożoność. Czas budowania modelu jest sumą czasów budowania modelu kolaboratywnego i modelu filtrowania z analizą zawartości.

Istnieje też ryzyko, że w momencie gdy wynik filtrowania kolaboratywnego będzie bardzo zły, to rezultat osiągnięty przez nową metodę będzie gorszy niż z samego filtrowania z analizą zawartości.

Niebezpieczeństwem może też być złe dopasowanie parametru k do przetwarzanych danych.

Rozdział 4

Ocena eksperymentalna

4.1. Opis metody badawczej

W celu zbadania jakości algorytmów zaprojektowany został eksperyment. W pierwszej kolejności definiowane są wartości parametrów, zgodnie z którymi ma zostać wykonany kod algorytmów. Dla parametru podlegającemu testom definiowany jest zakres wartości.

Następnie do pamięci ładowany jest zestaw danych zawierający oceny elementów przez użytkowników oraz dodatkowe własności elementów (jeżeli testowanym algorytmem jest filtrowanie z analizą zawartości lub filtrowanie hybrydowe). W celu przeprowadzenia walidacji krzyżowej zbiór dzielony jest k części. Z nich konstruowanych jest następnie k par: zbiór testowy-zbiór treningowy. Rozmiar każdego zbioru testowego jest równy rozmiarowi jednej części ($\frac{1}{k}$ liczby wszystkich ocen) a rozmiar zbioru treningowego to $k - 1*$rozmiar części .

Dla każdej pary zbiorów treningowego i testowego przeprowadzana jest nauka i sprawdzanie algorytmu. Z każdego przebiegu błędy są sumowane a na końcu wyciągana jest średnia.

Przebieg algorytmu wygląda następująco:

Algorytm 5 Przebieg eksperymentu

$recommenderType \leftarrow$ silnik rekomendacji: filtrowanie kolaboratywne, filtrowanie z analizą zawartości bądź filtrowanie hybrydowe
$parameters[] \leftarrow$ ustaw parametry wybranego algorytmu rekomendacji
$pt[] \leftarrow$ zdefiniuj wartości parametru testowanego
$cbUsersCount \leftarrow$ liczba użytkowników do pobrania do filtrowania z analizą zawartości
$cfUsersCount \leftarrow$ liczba użytkowników do pobrania do filtrowania kolaboratywnego

$minimumRepeatingFeatures \leftarrow$ minimalna ilość powtórzeń danej cechy
$pt[] \leftarrow$ zdefiniuj wartości parametru testowanego
$k \leftarrow$ parametr podziału dla do sprawdzianu krzyżowego
for $i \in pt$ **do**
  $parameters[parametrZmienny] := i$ **//Faza pobierania danych**

if *recommenderType* == filtrowanie kolaboratywne **or** filtrowanie hybrydowe **then**
  $cfUsers \leftarrow$ załaduj oceny *cfUsersCount* użytkowników;
else if *recommenderType* == filtrowanie z analizą zawartości **or** filtrowanie hybrydowe **then**
  $cbUsers \leftarrow$ załaduj oceny *cbUsersCount* użytkowników wraz z własnościami elementów;
end if
podziel zbiory na k części
//Faza nauki
for $i \in k$ **do**
  wybierz testowy i a z pozostałych danych utwórz zbiór treningowy
  if *recommenderType* == filtrowanie kolaboratywne **then**
    $cfRecommender \leftarrow$ ucz filtrowaniem kolaboratywnym wykorzystując zbiór treningowy $cfUsers$;
  else if *recommenderType* == filtrowanie z analizą zawartości **then**
    cbRecommender $\leftarrow$ ucz filtrowaniem z analizą zawartości wykorzystując zbiór treningowy $cbUsers$;
  else if *recommenderType* == filtrowanie hybrydowe **then**
    hybridRecommender $\leftarrow$ ucz filtrowaniem hybrydowym wykorzystując zbiory treningowe $cbUsers$ i $cfUsers$;
  end if
  Wykonaj test silnika rekomendacji wykorzystując zbiór testowy
  Oblicz błąd RMSE i MAE
end for
Oblicz średnią błędów RMSE i MAE ze wszystkich iteracji
Wyświetl wyniki
end for

4.1.1. Miara oceny

W celu zbadania jakości algorytmów zostały zastosowane miary oceny: średnia kwadratowa błędów (RMSE) i średni błąd bezwzględny (MAE).

Średnia kwadratowa błędów

Średnia kwadratowa błędów (ang. RMSE – *root mean square error*) jest często wykorzystywaną miarą służącą zmierzeniu różnicy pomiędzy wartościami przewidywanymi a rzeczywistymi (obserwowanymi).

RMSE jest stosunkowo dobrą miarą dokładności ale tylko w celu porównania różnych modeli dla tego samego zestawu danych. RMSE jest zależne od skali, zatem nie sprawdza się najlepiej w przypadku porównywania ze sobą różnych zmiennych [23].

Średnią kwadratową błędów wylicza się ze wzoru:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}\,, \tag{4.1}$$

gdzie

$\hat{y}_i$ to wartość przewidywana

y_i to wartość rzeczywista

Im niższa wartość RMSE tym bardziej zbliżone są wartości przewidywane do rzeczywistych, zatem tym lepszy jakościowo jest model.

Średni błąd bezwzględny

Inną miarą mierzenia jakości modeli predykcyjnych jest MAE (ang. *mean absolute error*). Podobnie jak RMSE miara ta jest zależna od skali, zatem najlepiej sprawdza się w działaniu na tym samym zestawie danych [23].

Średni błąd bezwzględny wylicza się ze wzoru:

$$MAE = \frac{1}{n} \sum_{t=1}^{n} |\hat{y}_t - y_t| , \tag{4.2}$$

gdzie

$\hat{y}_i$ to wartość przewidywana

y_i to wartość rzeczywista

4.1.2. Zbiory danych

W celu uzyskania obiektywnych wyników algorytmy zostały przetestowane na dwóch różnych bazach danych z dwóch różnych domen. Eksperyment nie ma na celu porównania wyników między różnymi zestawami danych, lecz porównanie wyników testowanych algorytmów w różnych warunkach.

MovieLens

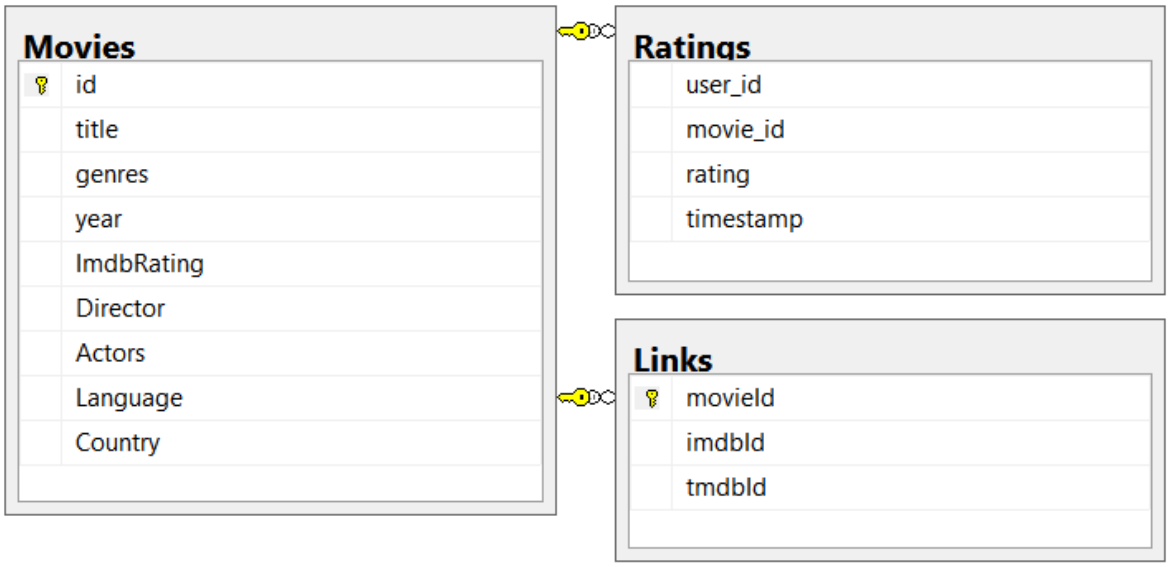
MovieLens [18] to baza zawierająca oceny filmów przez użytkowników portalu movielens.org. Baza zawiera 3,706 filmów i 1,000,209 ocen wystawionych przez 6,040 unikalnych użytkowników pomiędzy 25 kwietnia 2000 a 28 lutym 2003, średnio 166 ocen na użytkownika i 270 ocen na film. Filmy oceniane są w skali od 1 do 5, gdzie 1 jest oceną najgorszą a 5 najlepszą.

Baza zawiera tabelę łączącą numery identyfikacyjne filmów z bazą IMDB.com. Wykorzystując te informacje baza filmów została rozszerzona o informacje pobrane z IMDB.com. Ostateczny kształt bazy widoczny jest na rys. 4.1.

Amazon Meta

AmazonMeta to baza danych zawierająca informacje na temat zakupu produktów ze sklepu internetowego amazon.com. Dane zostały zgromadzone w połowie roku 2006. Zawierają 548,552 produktów i 7,781,990 ocen produktów [4, 34]. Podobnie jak w przypadku bazy MovieLens, produkty ocenione są w skali od 1 do 5, gdzie 1 jest oceną najgorszą a 5 najlepszą.

Produkty opisane są takimi atrybutami jak nazwa, kategoria, ranking sprzedażowy, liczba produktów kupowanych jednocześnie, liczba recenzji i średnia ocena.



Rys. 4.1: Schemat bazy MovieLens

4.2. Środowisko symulacyjne

4.2.1. Oprogramowanie

Rys. 4.2 przedstawia zrzut ekranu środowiska symulacyjnego opracowanego przez autorkę pracy na potrzeby przeprowadzenia badań algorytmów. Główny interfejs programu składa się z trzech części: ustawienia podstawowe, okno z wynikiem oraz panel sterujący ustawieniami zaawansowanymi.

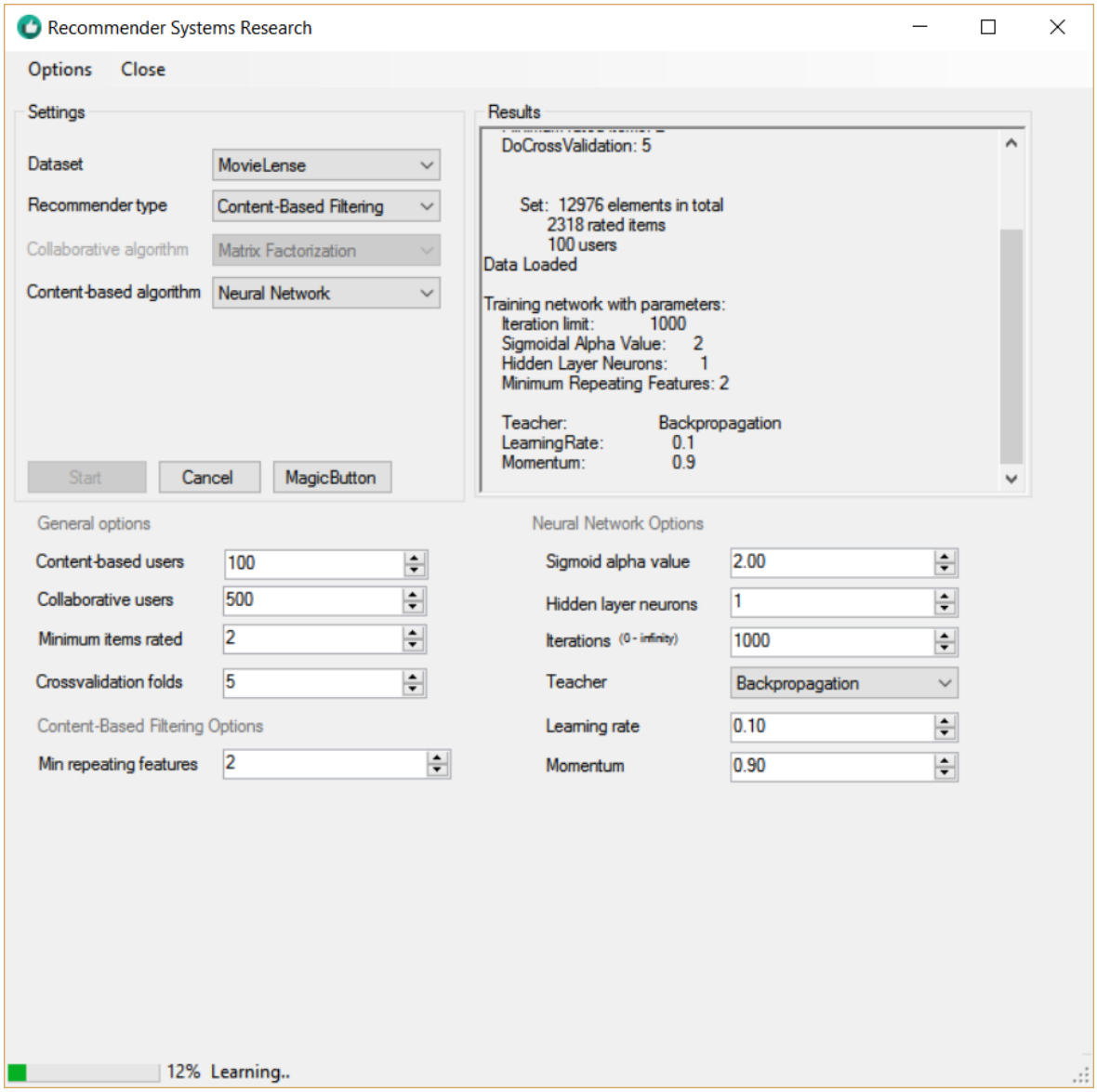
W sekcji z ustawieniami podstawowymi możliwy jest wybór:

- bazy danych, która zostanie wykorzystana do pomiarów;
- typu algorytmu rekomendującego (filtrowanie z analizą zawartości, filtrowanie kolaboratywne bądź algorytm hybrydowy)
- w przypadku wyboru kolaboratywnego filtrowania lub filtrowania hybrydowego możliwy jest wybór typu algorytmu: Matrix Factorization, Biased Matrix Factorization lub SVD++.
- w przypadku wyboru filtrowania z analizą zawartości lub filtrowania hybrydowego automatycznie wybierany jest algorytm oparty na sieci neuronowej.

Widok ustawień zaawansowanych zmienia się w zależności od wybranego filtrowania. W przypadku wyboru filtrowania z analizą zawartości istnieje możliwość regulowania parametrów sieci neuronowej. W każdym wariancie można regulować kryteria doboru zestawu danych.

Kryteria doboru zestawu danych są następujące:

- **Liczba użytkowników CBF** (*cbUsersCount*) – liczba użytkowników uwzględnianych w metodzie filtrowania z analizą zawartości;
- **Liczba użytkowników CF** (*cfUsersCount*) – liczba użytkowników uwzględnianych w metodzie filtrowania kolaboratywnego;



Rys. 4.2: Zrzut ekranu środowiska symulacyjnego

- **Minimum ocenionych elementów** (*minimumItemsRated*) – minimalna liczba elementów, które użytkownik ocenił aby zostać uwzględnionym w czasie przebiegu algorytmu;
- **Minimum powtarzających się cech** (*minimumRepeatingFeatures*) – minimalna ilość powtórzeń danej cechy elementów, aby była brana pod uwagę w trakcie budowania rekomendacji;
- **Parametr walidacji krzyżowej** (*crossvalidationFolds*) – ilość podzbiorów, na które dzielony jest główny zbiór w trakcie walidacji krzyżowej.

Parametry sieci neuronowej podlegające strojeniu to:

- α – parametr funkcji aktywacji sieci neuronowej;
- **Ilość neuronów w warstwie ukrytej** (*hiddenLayerNeurons*) – liczba neuronów w warstwie ukrytej;

- **Maksymalna ilość iteracji nauki** (*iterationLimit*) – maksymalna liczba iteracji uczenia sieci neuronowej;
- **Algorytm uczący** (*teacherFunction*) – algorytm uczący sieć neuronową: propagacja wsteczna, rprop (resilient backpropagation) lub algorytm genetyczny;
- **Rozmiar populacji** (*populationSize*) – w przypadku wyboru algorytmu genetycznego – rozmiar każdej kolejnej populacji.

4.2.2. Sprzęt

Wszystkie badania zostały przeprowadzone w tych samych warunkach na komputerze klasy PC działającym pod kontrolą 64–bitowego systemu Windows 10 Pro. Konfiguracja sprzętowa wygląda następująco:

Procesor:	Intel Core i5–6200U 2.30GHz
RAM:	8 GB
Typ dysku twardego:	SSD

4.3. Eksperymentalne dopasowanie parametrów sieci neuronowej

Pierwszy przeprowadzony eksperyment miał na celu dopasowanie optymalnych parametrów sieci neuronowej dla filtrowania z analizą zawartości odrębnie dla baz MovieLens i AmazonMeta.

4.3.1. Strojenie sieci dla bazy MovieLens

Dopasowanie maksymalnej liczby iteracji

Uczenie sieci neuronowej odbywa się w pętli tak długo, jak wartość błędu na wyjściu spadnie poniżej określonej wartości bądź przekroczona zostanie maksymalna ilość iteracji. Eksperyment miał na celu zbadanie wartości błędów w zależności od liczby iteracji.

Konfiguracja parametrów podstawowych	
Baza danych:	MovieLens
Testowana metoda:	Filtrowanie z analizą zawartości
Konfiguracja filtrowania z analizą zawartości	
Algorytm:	Sieć neuronowa uczona propagacją wsteczną, RPROP i algorytmem genetycznym
Maksymalna ilość iteracji nauki:	?
Ilość neuronów w warstwie ukrytej:	1
Współczynnik bezwładności:	0.9
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0

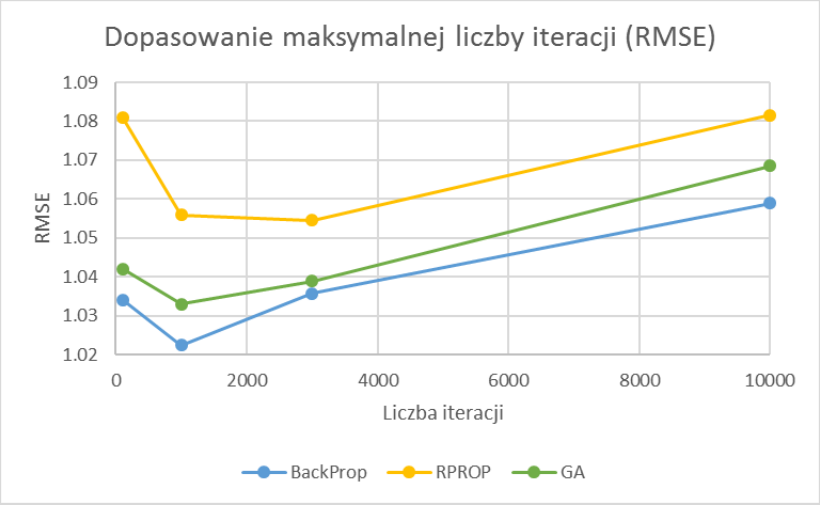
Rozmiar populacji:	100
Minimum powtarzających się cech:	10
Minimum ocenionych elementów przez użytkownika:	10
Ilość testowanych użytkowników:	10

Tabela 4.1: Konfiguracja dla eksperymentu maksymalnej liczby iteracji

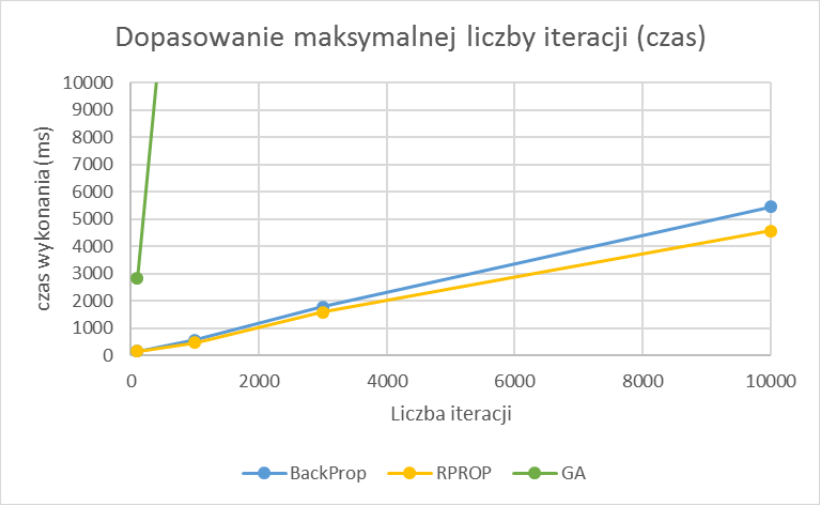
Wyniki eksperymentu przedstawiają wykresy 4.3 i 4.4 oraz tabela 4.2.

L. iter.	**Propagacja wsteczna**		**RPROP**		**Algorytm genetyczny**	
	RMSE	**czas**	**RMSE**	**czas**	**RMSE**	**czas**
100	1.03402	155	1.081065	139	1.042103	2824
1000	1.022469	564	1.055915	463	1.033005	24831
3000	1.035766	1770	1.054472	1581	1.038922	70579
10000	1.058892	5444	1.081511	4567	1.068489	239775

Tabela 4.2: Zależność błędu algorytmu wyrażonego za pomocą RMSE i czasu wykonania od maksymalnej liczy iteracji (baza MovieLens).



Rys. 4.3: Wyniki eksperymentu testującego parametr maksymalnej liczby iteracji na bazie MovieLens. Wykres przedstawia zależność pomiędzy liczbą iteracji a błędem algorytmu wyrażonym za pomocą RMSE. Wraz ze wzrostem liczby iteracji wartość błędu maleje, jednakże po przekroczeniu pewnej wartości następuje ponowny wzrost. Wynika to z występowania zjawiska nadmiernego dopasowania (ang. *overfitting*).



Rys. 4.4: Wyniki eksperymentu testującego parametr maksymalnej liczby iteracji na bazie MovieLens. Wykres przedstawia zależność pomiędzy liczbą iteracji a czasem wykonania. Zgodnie z oczekiwaniami, czas wykonania jest wprost proporcjonalny do liczby iteracji. W trakcie tego eksperymentu potwierdzone zostały również przypuszczenia dotyczące prędkości działania algorytmów. Najszybszy okazał się RPROP, jednakże kosztem nieco większych błędów. Najgorzej wypadł algorytm genetyczny, który charakteryzował się bardzo szybko rosnącym czasem wykonania przy jednoczesnym nie najlepszym wynikiem RMSE.

Dopasowanie współczynnika bezwładności

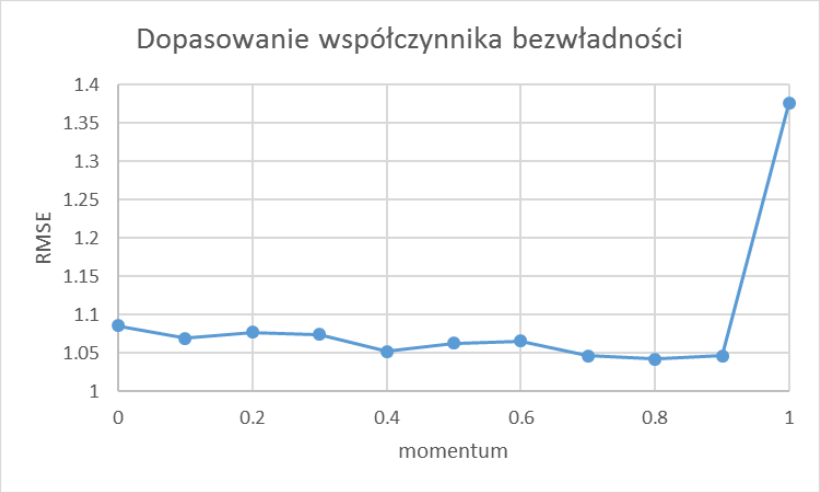
Współczynnik bezwładności jest dodatkowym współczynnikiem mającym na celu ulepszenie działania algorytmu propagacji wstecznej (zob. 2.3.4). Eksperyment miał na celu zbadanie dla jakich wartości osiągany jest najniższy błąd.

Wyniki eksperymentu przedstawia wykres 4.5 i tabela 4.4.

Konfiguracja parametrów podstawowych	
Baza danych:	MovieLens
Testowana metoda:	Filtrowanie z analizą zawartości
Konfiguracja filtrowania z analizą zawartości	
Algorytm:	Sieć neuronowa uczona propagacją wsteczną
Maksymalna ilość iteracji nauki:	1000
Ilość neuronów w warstwie ukrytej:	2
Współczynnik bezwładności:	?
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0
Minimum powtarzających się cech:	10
Minimum ocenionych elementów przez użytkownika:	100

Ilość testowanych użytkowników: 5

Tabela 4.3: Konfiguracja dla eksperymentu dopasowania wartości współczynnika bezwładności



Rys. 4.5: Wyniki eksperymentu testującego optymalną wartość współczynnika bezwładności na bazie MovieLens. Wykres przedstawia zależność pomiędzy wartością parametru a błędem RMSE. W zakresie wartości $[0, 0.9]$ błąd oscyluje w granicach podobnych wartości z nieznaczną tendencją malejącą. Wyraźny skok na niekorzyść odnotowany jest dopiero przy momentum$= 1$.

Momentum	**RMSE**	**MAE**
0	1.086007	0.855964
0.1	1.069125	0.851345
0.2	1.077596	0.860146
0.3	1.074588	0.851467
0.4	1.052529	0.8424
0.5	1.063205	0.846198
0.6	1.065994	0.856964
0.7	1.046752	0.83462
0.8	1.042235	0.842353
0.9	1.046243	0.840319
1	1.375336	1.071652

Tabela 4.4: Zależność błędu algorytmu RMSE od wartości współczynnika bezwładności (baza MovieLens).

Dopasowanie rozmiaru populacji

Parametr rozmiaru populacji determinuje ilość osobników, jaka zostanie wygenerowana w każdej epoce przebiegu algorytmu genetycznego (zob. 2.3.4). Eksperyment miał na celu zbadanie korelacji pomiędzy rozmiarem populacji a błędem algorytmu rekomendacji. Mierzony jest również czas uczenia sieci neuronowej.

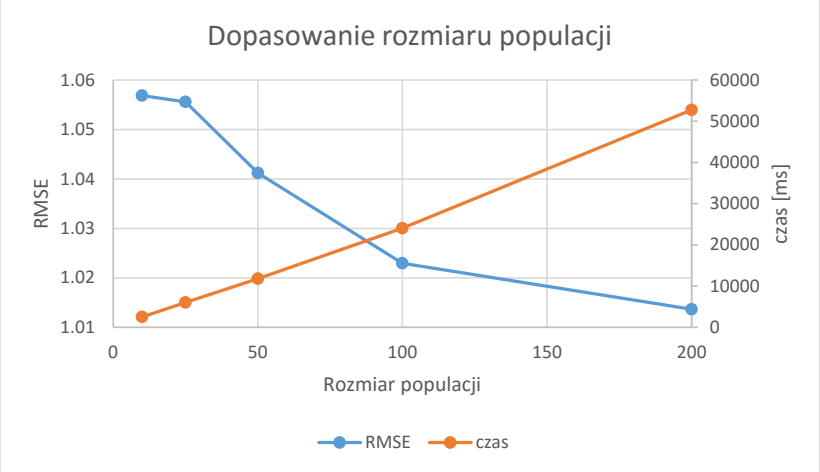
Konfiguracja parametrów podstawowych	
Baza danych:	MovieLens
Testowana metoda:	Filtrowanie z analizą zawartości
Konfiguracja filtrowania z analizą zawartości	
Algorytm:	Sieć neuronowa uczona algorytmem genetycznym
Maksymalna ilość iteracji nauki:	1000
Ilość neuronów w warstwie ukrytej:	1
Współczynnik bezwładności:	0.9
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0
Rozmiar populacji:	?
Minimum powtarzających się cech:	10
Minimum ocenionych elementów przez użytkownika:	10
Ilość testowanych użytkowników:	10

Tabela 4.5: Konfiguracja dla eksperymentu dopasowania wielkości populacji

Wyniki eksperymentu przedstawia wykres 4.6 i tabela 4.6.

Populacja	**RMSE**	**MAE**	**czas**
10	1.056848	0.861325	2519
25	1.055562	0.859391	6015
50	1.041198	0.850866	11804
100	1.022943	0.825477	24062
200	1.013656	0.816789	52751

Tabela 4.6: Zależność błędu algorytmu RMSE od rozmiaru populacji.



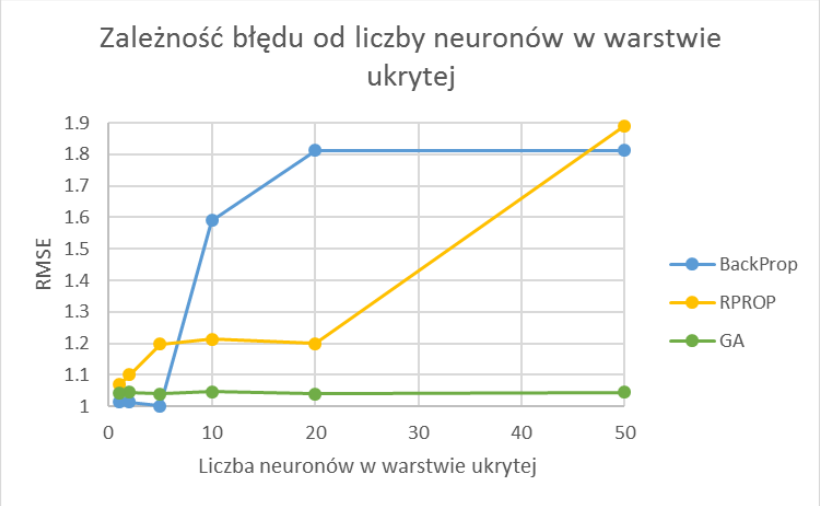
Rys. 4.6: Wyniki eksperymentu testującego optymalny rozmiar populacji na bazie MovieLens. Wykres przedstawia zależność pomiędzy rozmiarem a błędem RMSE. Zgodnie z oczekiwaniami występuje dodatnia korelacja pomiędzy rozmiarem populacji a czasem wykonania i ujemna pomiędzy rozmiarem populacji a błędem algorytmu rekomendacji. Zależność pomiędzy rozmiarem populacji a czasem wykonania jest liniowa, natomiast błąd maleje wykładniczo gdy populacja osiąga większe rozmiary. Optymalnym zatem jest przyjęcie rozmiaru populacji równej 100.

Dopasowanie ilości neuronów w warstwie ukrytej

Eksperyment miał na celu zbadanie dla jakiej liczby neuronów w warstwie pośredniej algorytm osiąga optymalne rezultaty (zob. 3.2.2). Wyniki eksperymentu przedstawia wykres 4.7 i tabela 4.8.

Konfiguracja parametrów podstawowych	
Baza danych:	MovieLens
Testowana metoda:	Filtrowanie z analizą zawartości
Konfiguracja filtrowania z analizą zawartości	
Algorytm:	Sieć neuronowa uczona propagacją wsteczną, RPROP i algorytmem genetycznym
Maksymalna ilość iteracji nauki:	1000
Ilość neuronów w warstwie ukrytej:	?
Współczynnik bezwładności:	0.9
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0
Minimum powtarzających się cech:	10
Minimum ocenionych elementów przez użytkownika:	10
Ilość testowanych użytkowników:	20

Tabela 4.7: Konfiguracja dla eksperymentu dopasowania rozmiaru ukrytej warstwy neuronów



Rys. 4.7: Wyniki eksperymentu testującego optymalny rozmiar ukrytej warstwy neuronów na bazie MovieLens. Wykres przedstawia zależność pomiędzy ilością neuronów a błędem RMSE. Wyniki prezentują się odmiennie w zależności od typu algorytmu wykorzystanego do uczenia sieci. W przypadku algorytmów propagacji wstecznej i RPROP występuje dodatnia korelacja pomiędzy rozmiarem warstwy ukrytej a błędem algorytmu rekomendacji. Dla algorytmu BackProp optymalny rezultat osiągnięty został dla zaledwie jednego neuronu w warstwie ukrytej, dla algorytmu RPROP dla pięciu neuronów. W tym ostatnim przypadku poprawa znajduje się jednak na marginesie błędu statystycznego. W przypadku algorytmu genetycznego, niezależnie od ilości neuronów wynik oscyluje na podobnym poziomie w zakresie testowanych wartości.

L. neuronów	**Propagacja wsteczna**		**RPROP**		**Alg. genetyczny**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
1	1.013503	0.811766	1.068271	0.846588	1.042597	0.836481
2	1.013808	0.816021	1.100579	0.862516	1.045096	0.836361
5	1.002053	0.807083	1.197693	0.93349	1.038662	0.831887
10	1.589821	1.232048	1.213635	0.9478	1.046554	0.835417
20	1.812912	1.458733	1.199205	0.931307	1.039479	0.833191
50	1.812912	1.458733	1.891729	1.562207	1.044693	0.837189

Tabela 4.8: Zależność błędu algorytmu RMSE od rozmiaru ukrytej warstwy neuronowej (baza MovieLens).

Dopasowanie współczynnika uczenia

(NA KOŃCU, BADANIA) Czy trzeba?

Dopasowanie wartości alfa

(NA KOŃCU, BADANIA) Czy trzeba

Podsumowanie

Bazując na przeprowadzonych eksperymentach zostały zdefiniowane optymalne parametry algorytmu filtrowania z analizą zawartości. Zostały one przedstawione w tabeli 4.9.

Optymalna konfiguracja filtrowania z analizą zawartości dla bazy MovieLens	
Maksymalna ilość iteracji nauki:	1000
Współczynnik bezwładności:	0.9
Rozmiar populacji:	100
Ilość neuronów w warstwie ukrytej:	1
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0

Tabela 4.9: Konfiguracja dla eksperymentu dopasowania rozmiaru ukrytej warstwy neuronów

4.3.2. Strojenie sieci dla bazy AmazonMeta

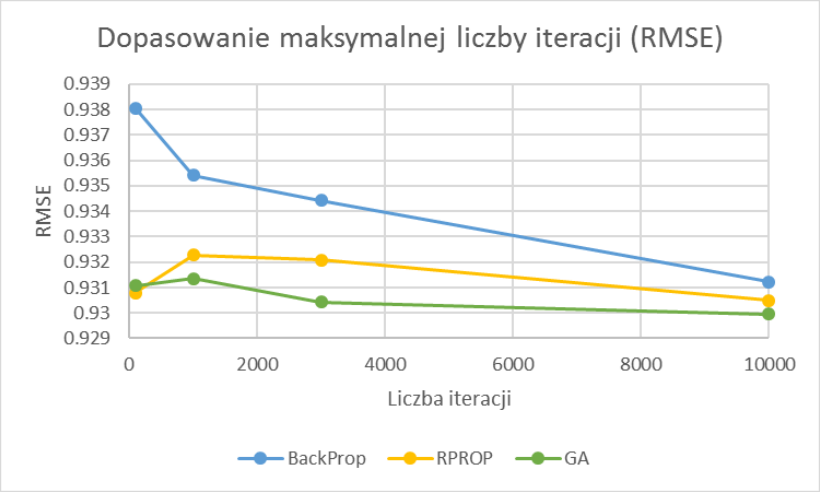
W analogiczny sposób zostały przeprowadzone badania dla bazy AmazonMeta. Konfiguracja wstępna wszystkich eksperymentów była taka sama jak dla bazy MovieLens (zob. 4.3.1).

Dopasowanie maksymalnej liczby iteracji

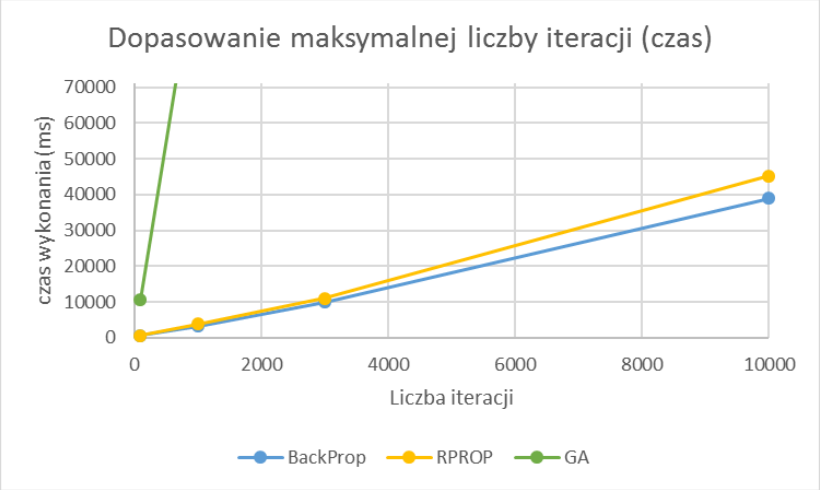
Eksperyment miał na celu zbadanie wartości błędów w zależności od liczby iteracji. Wyniki eksperymentu przedstawiają wykresy 4.8 i 4.9 oraz tabela 4.10.

L. iter.	**Propagacja wsteczna**		**RPROP**		**Algorytm genetyczny**	
	RMSE	**czas**	**RMSE**	**czas**	**RMSE**	**czas**
100	0.938058	564	0.930788	585	0.931064	10641
1000	0.935423	3289	0.93228	3784	0.931359	108559
3000	0.934419	9915	0.932073	11052	0.930428	719324
10000	0.931222	38917	0.93049	45258	0.929965	3229148

Tabela 4.10: Zależność błędu algorytmu wyrażonego za pomocą RMSE i czasu wykonania od maksymalnej liczy iteracji (baza AmazonMeta).



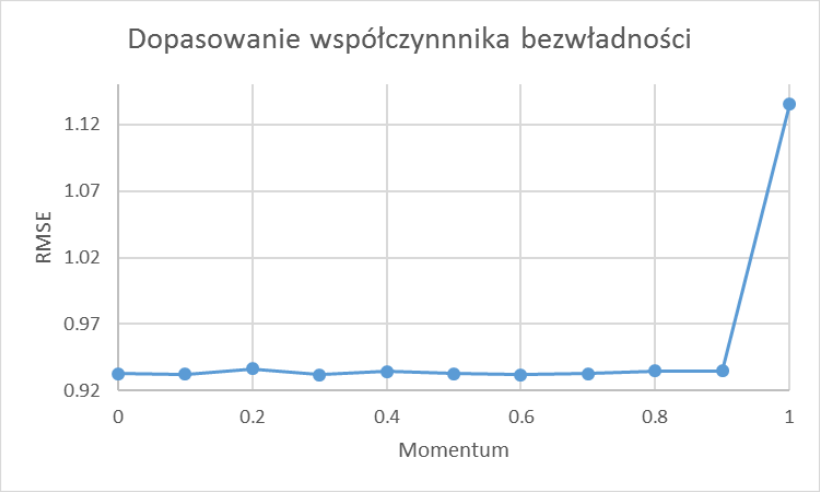
Rys. 4.8: Wyniki eksperymentu testującego parametr maksymalnej liczby iteracji na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy liczbą iteracji a błędem algorytmu wyrażonym za pomocą RMSE. Początkowo wartość błędu szybko maleje, jednakże wraz ze wzrostem liczby iteracji kąty nachylenia kolejnych odcinków krzywej ulegają zwiększeniu. Biorąc pod uwagę rosnący liniowo czas wykonania (zob. 4.9) w pewnym momencie koszt w postaci czasu wykonania przewyższa korzyść wynikającą z niższego błędu RMSE. W zakresach liczby iteracji od 100 do 2000 występują nieznaczne zachwiania, które znajdują się w zakresie błędu statystycznego.



Rys. 4.9: Wyniki eksperymentu testującego parametr maksymalnej liczby iteracji na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy liczbą iteracji a czasem wykonania. Czas wykonania jest wprost proporcjonalny do liczby iteracji.

Dopasowanie współczynnika bezwładności

Eksperyment miał na celu zbadanie dla jakich wartości współczynnika bezwładności osiągany jest najniższy błąd. Wyniki eksperymentu przedstawia wykres 4.10 i tabela 4.11.



Rys. 4.10: Wyniki eksperymentu testującego optymalną wartość współczynnika bezwładności na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy wartością parametru a błędem RMSE. W zakresie wartości $[0, 0.9]$ błąd oscyluje w granicach podobnych wartości z nieznaczną tendencją malejącą. Wyraźny skok na niekorzyść odnotowany jest dopiero przy momentum$= 1$.

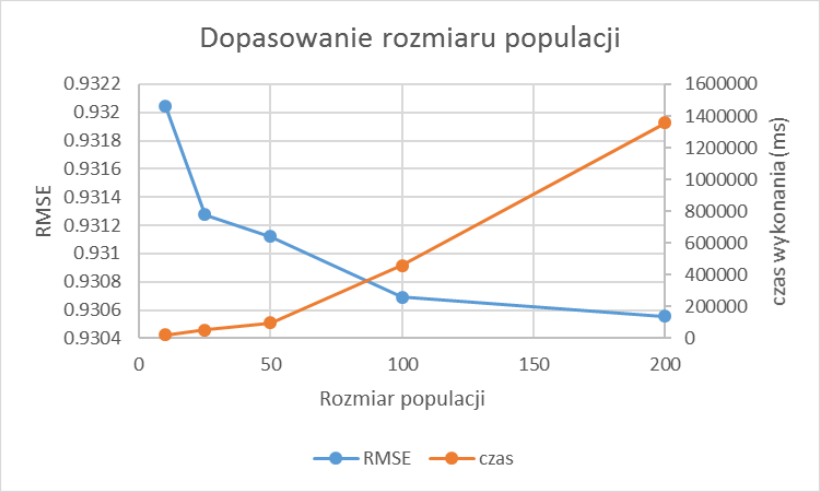
Momentum	**RMSE**	**MAE**
0	0.932621	0.712677
0.1	0.932369	0.711739
0.2	0.9366	0.71353
0.3	0.931794	0.718193
0.4	0.934477	0.716904
0.5	0.932799	0.717865
0.6	0.932026	0.709617
0.7	0.93268	0.714115
0.8	0.934683	0.714445
0.9	0.934906	0.716833
1	1.135473	0.8761

Tabela 4.11: Zależność błędu algorytmu RMSE od wartości współczynnika bezwładności (baza AmazonMeta).

Dopasowanie rozmiaru populacji

Eksperyment miał na celu zbadanie korelacji pomiędzy rozmiarem populacji a błędem algorytmu rekomendacji. Wyniki eksperymentu przedstawia wykres 4.11 i tabela **??**.

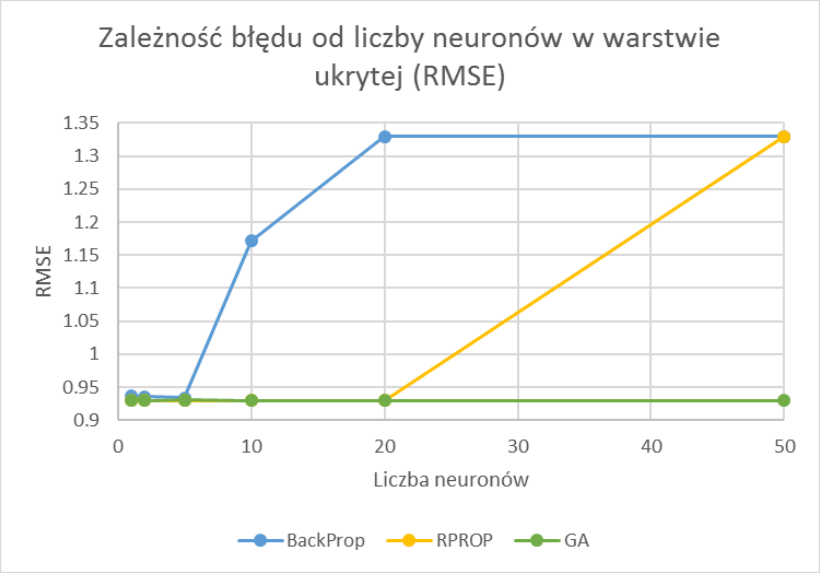
wstawić tabelkę



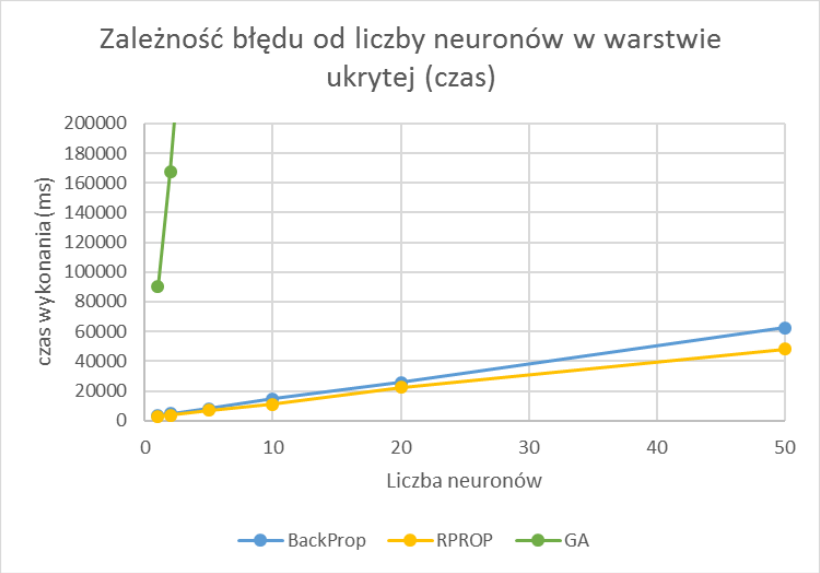
Rys. 4.11: Wyniki eksperymentu testującego optymalny rozmiar populacji na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy rozmiarem a błędem RMSE. Zgodnie z oczekiwaniami występuje dodatnia korelacja pomiędzy rozmiarem populacji a czasem wykonania i ujemna pomiędzy rozmiarem populacji a błędem algorytmu rekomendacji. Zależność pomiędzy rozmiarem populacji a czasem wykonania jest liniowa, natomiast błąd maleje wykładniczo gdy populacja osiąga większe rozmiary. Optymalnym zatem jest przyjęcie rozmiaru populacji równej 50 tak, aby czas wykonania nie przekraczał 200000ms.

Dopasowanie ilości neuronów w warstwie ukrytej

Eksperyment miał na celu zbadanie dla jakiej liczby neuronów w warstwie pośredniej algorytm osiąga optymalne rezultaty. Wyniki eksperymentu przedstawiają wykresy 4.12 i 4.13 oraz tabela 4.12.



Rys. 4.12: Wyniki eksperymentu testującego optymalny rozmiar ukrytej warstwy neuronów na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy ilością neuronów a błędem RMSE. W przypadku algorytmów propagacji wstecznej i RPROP występuje dodatnia korelacja pomiędzy rozmiarem warstwy ukrytej a błędem algorytmu rekomendacji. Dla algorytmu BackProp optymalne wyniki osiągnięte zostały dla zaledwie liczby neuronów w warstwie ukrytej nie przekraczającej 5, dla algorytmu RPROP dla nie przekraczającej 20. W przypadku algorytmu genetycznego, niezależnie od ilości neuronów wynik oscyluje na podobnym poziomie w zakresie testowanych wartości.



Rys. 4.13: Wyniki eksperymentu testującego optymalny rozmiar ukrytej warstwy neuronów na bazie AmazonMeta. Wykres przedstawia zależność pomiędzy ilością neuronów a czasem wykonania. Czas wykonania rośnie liniowo proporcjonalnie do ilości neuronów. W związku z tym oraz biorąc pod uwagę wykres 4.12 optymalnym jest ustalenie liczby neuronów w warstwie ukrytej na 1.

L. neuronów	Propagacja wsteczna		RPROP		Alg. genetyczny	
	RMSE	czas	RMSE	czas	RMSE	czas
1	0.936219	3817	0.930452	2854	0.930968	89939
2	0.935587	4680	0.930152	3629	0.930092	167708
5	0.933983	7947	0.929796	6977	0.930443	471038
10	1.171907	14826	0.930305	11182	0.930253	651918
20	1.32946	25576	0.930192	22295	0.930148	937888
50	1.32946	62574	1.32946	48140	0.92978	2055270

Tabela 4.12: Zależność błędu algorytmu RMSE i czasu wykonania od rozmiaru ukrytej warstwy neuronowej (baza AmazonMeta).

wstawić tabelkę

Dopasowanie współczynnika uczenia

(NA KOŃCU, BADANIA) Czy trzeba?

Dopasowanie wartości alfa

(NA KOŃCU, BADANIA) Czy trzeba?

Podsumowanie

Bazując na przeprowadzonych eksperymentach zostały zdefiniowane optymalne parametry algorytmu filtrowania z analizą zawartości. Zostały one przedstawione w tabeli 4.13.

Optymalna konfiguracja filtrowania z analizą zawartości dla bazy AmazonMeta	
Maksymalna ilość iteracji nauki:	1000
Współczynnik bezwładności:	0
Rozmiar populacji:	50
Ilość neuronów w warstwie ukrytej:	1
Współczynnik uczenia:	0.1
Parametr funkcji aktywacji α:	2.0

Tabela 4.13: Konfiguracja dla eksperymentu dopasowania rozmiaru ukrytej warstwy neuronów

4.4. Eksperymentalne porównanie zaproponowanych algorytmów hybrydowych z filtrowaniem z analizą zawartości i filtrowaniem kolaboratywnym

Eksperymenty miały na celu porównanie efektywności zaproponowanych algorytmów hybrydowych z samym filtrowaniem kolaboratywnym i z samym filtrowaniem z analizą zawartości.

Początkowo zostały przeprowadzone testy odrębnie dla filtrowania kolaboratywnego i dla filtrowania z analizą zawartości. Następnie dla takie samej konfiguracji parametrów zostały uruchomione algorytmy hybrydowe.

W pierwszym eksperymencie parametrem, który ulegał modyfikacji był parametr określający minimum powtarzających się cech ocenianych elementów. W drugim eksperymencie modyfikowana była liczba użytkowników brana pod uwagę przez filtrowanie kolaboratywne.

W każdym przypadku sieć neuronowa skonfigurowana została w sposób optymalny wynikający z eksperymentów przeprowadzonych w sekcjach 4.3.1 i 4.3.2.

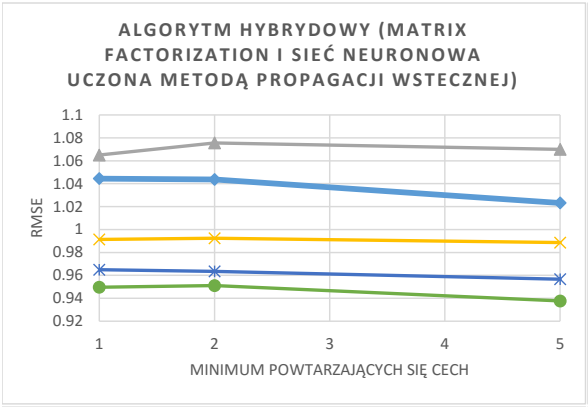
4.4.1. Eksperyment pierwszy - efektywność algorytmów w zależności od minimum powtarzających się cech elementów

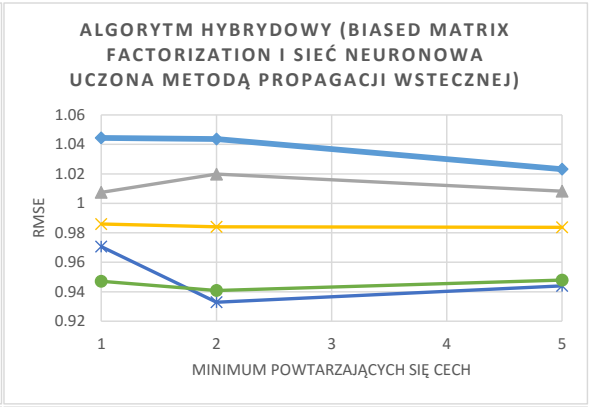
Badania na bazie AmazonMeta

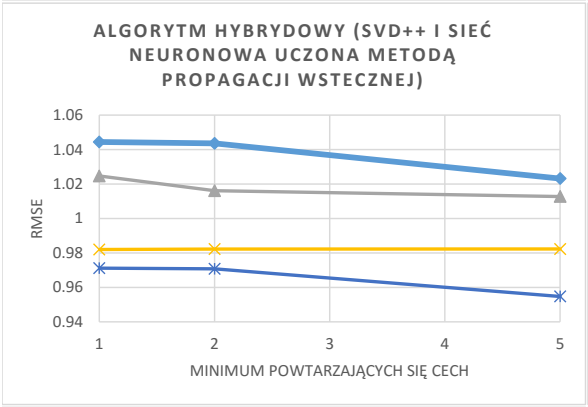
(BADANIA) Porównanie algorytmu hybrydowego z content-based i collaborative

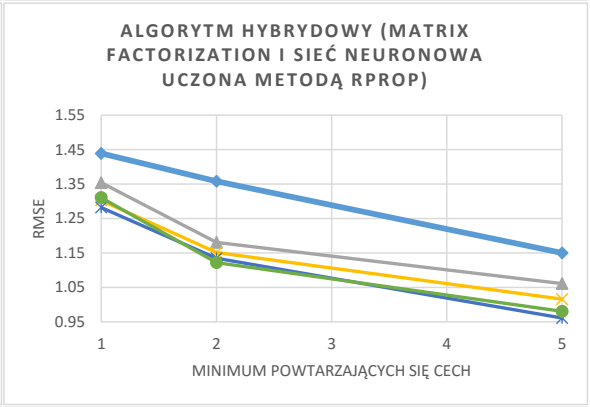
4.4.2. Analiza statystyczna wyników

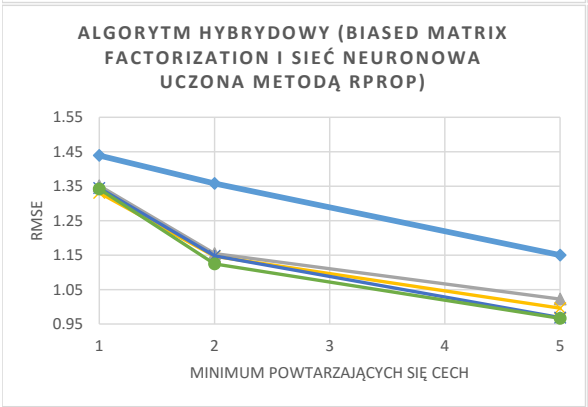
(PÓŹNIEJ) Analiza statystyczna wyników, PQStat

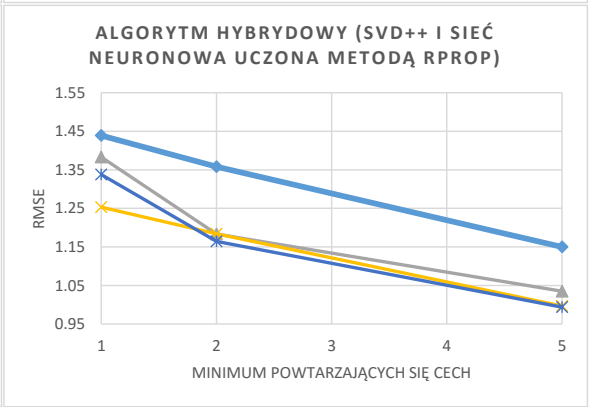

ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.1
1.08
1.06
1.04
1.02
1
0.98
0.96
0.94
0.92
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

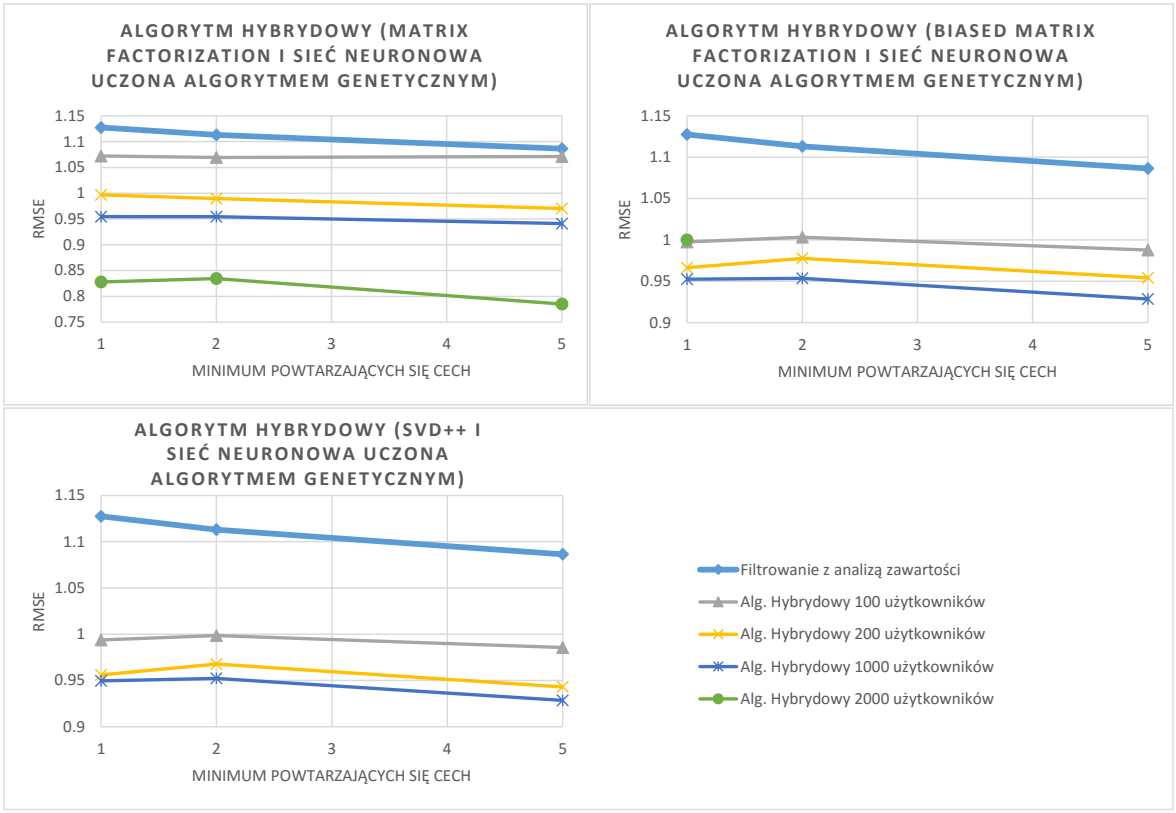

ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.06
1.04
1.02
1
0.98
0.96
0.94
0.92
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH


ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.06
1.04
1.02
1
0.98
0.96
0.94
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

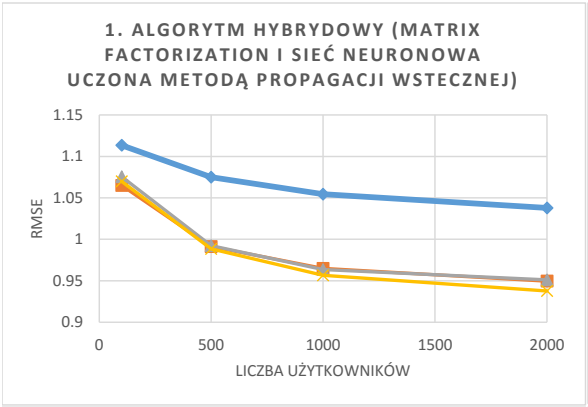

ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.55
1.45
1.35
1.25
1.15
1.05
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

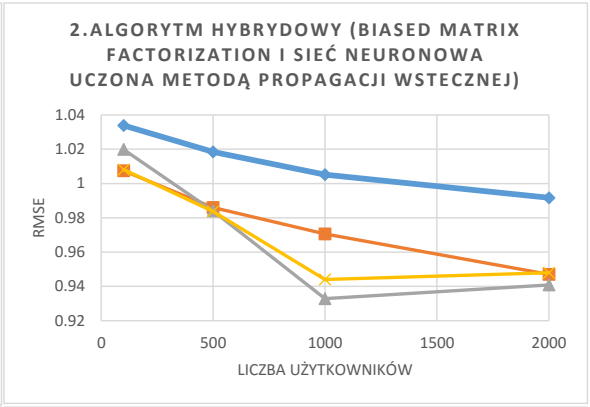

ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.55
1.45
1.35
1.25
1.15
1.05
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

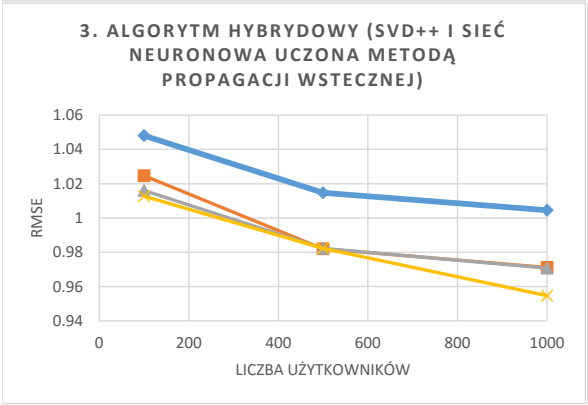

ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.55
1.45
1.35
1.25
1.15
1.05
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

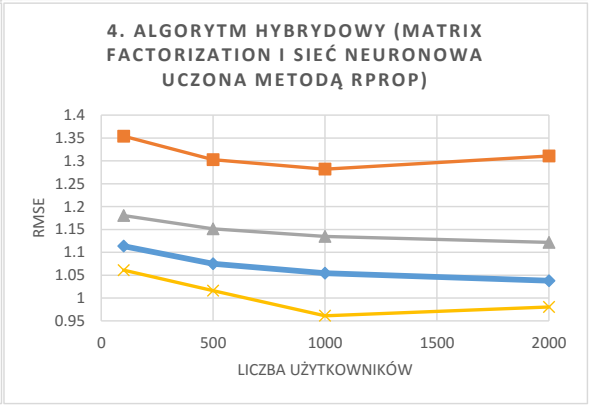


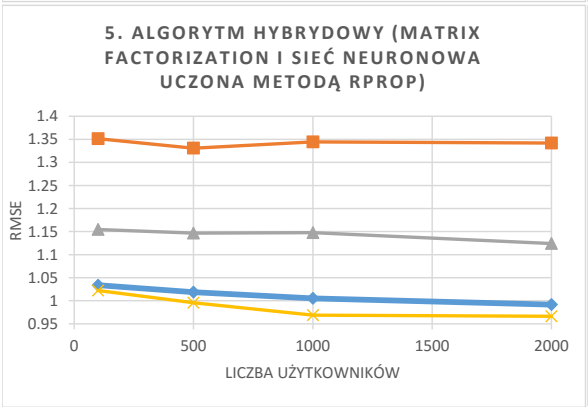
Rys. 4.14: Porównanie algorytmu hybrydowego z analizą zawartości na podstawie danych z bazy MovieLense. Skuteczność filtrowania z hybrydowego w zależności od liczby użytkowników i powtarzających się cech. Pogrubiona linia przedstawia działanie filtrowania z analizą zawartości.

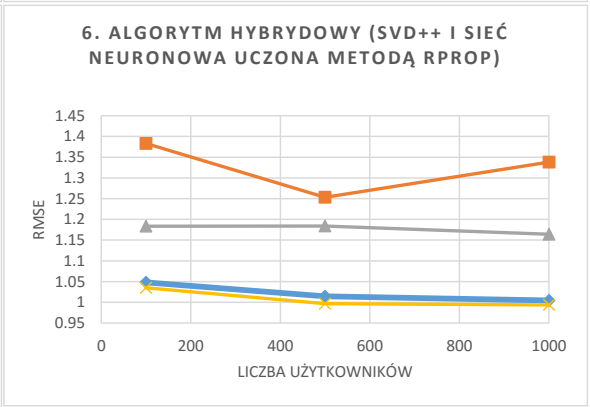

1. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.15
1.1
1.05
1
0.95
0.9
0
500
1000
1500
2000
LICZBA UŻYTKOWNIKÓW

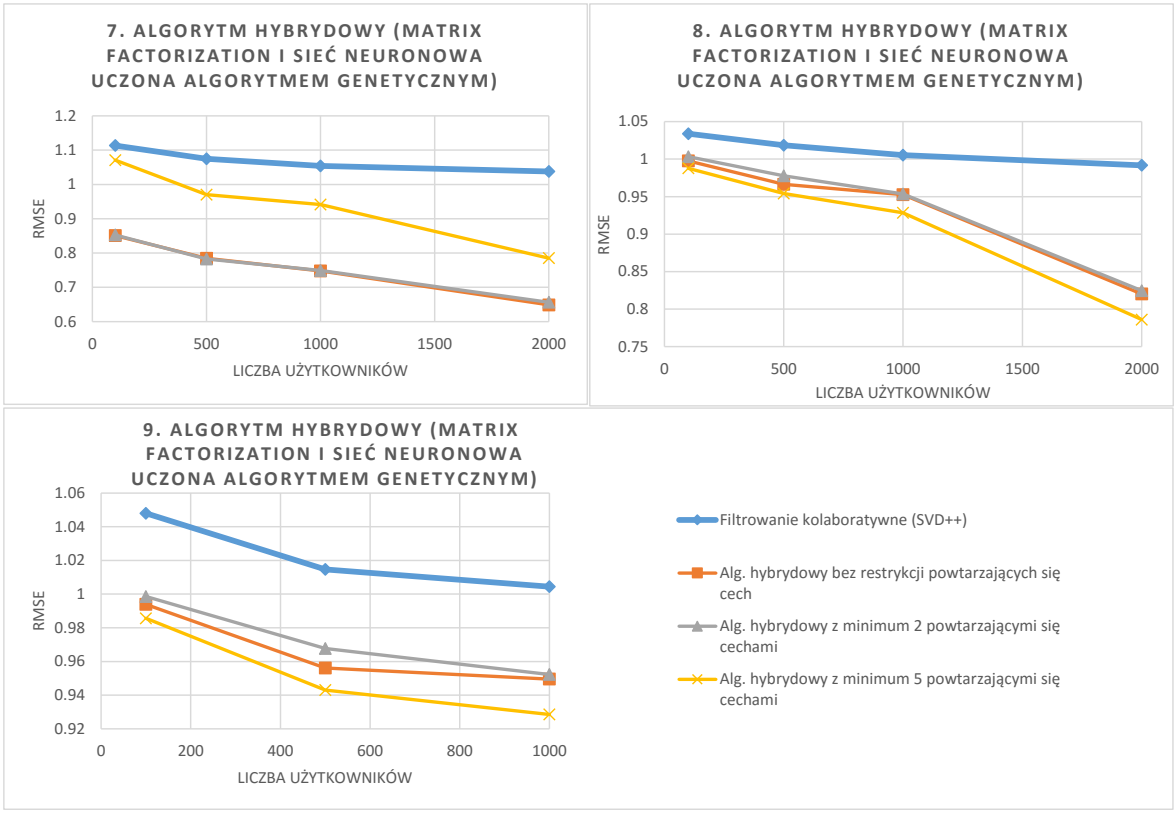

2.ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.04
1.02
1
0.98
0.96
0.94
0.92
0
500
1000
1500
2000
LICZBA UŻYTKOWNIKÓW


3. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.06
1.04
1.02
1
0.98
0.96
0.94
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW

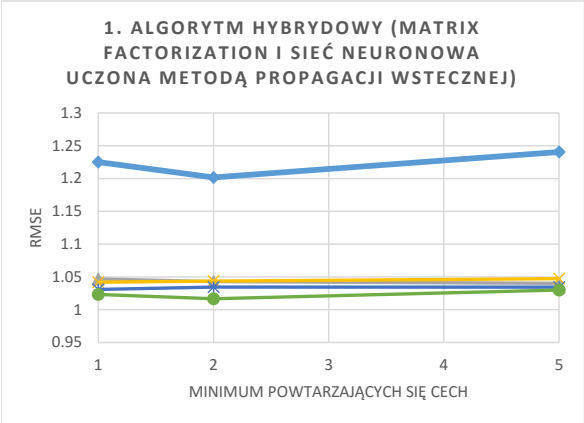

4. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.4
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
0
500
1000
1500
2000
LICZBA UŻYTKOWNIKÓW

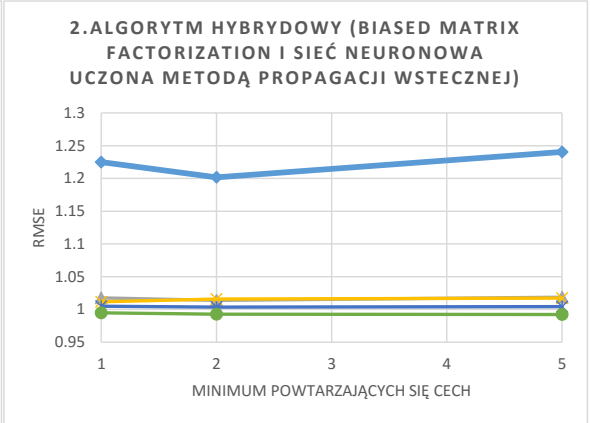

5. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.4
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
0
500
1000
1500
2000
LICZBA UŻYTKOWNIKÓW

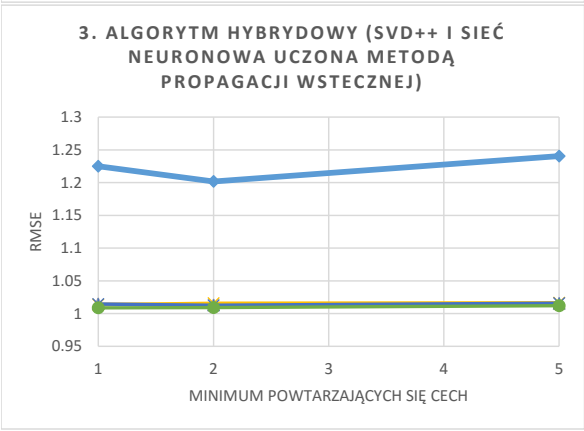

6. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.45
1.4
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW

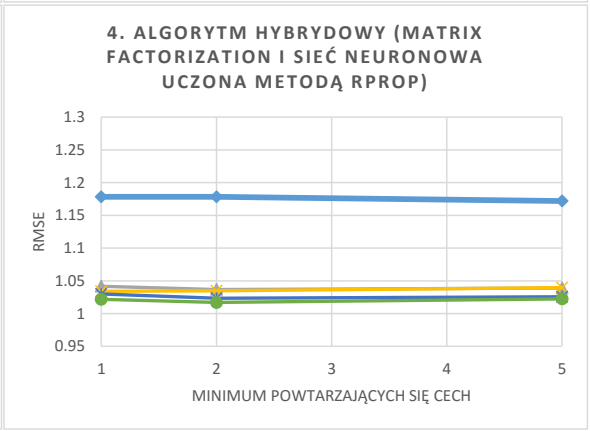


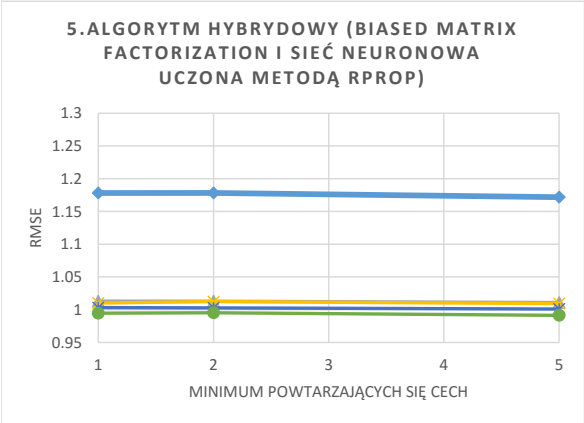
Rys. 4.15: Porównanie algorytmu hybrydowego z kolaboratywnym na podstawie danych z bazy MovieLense. Skuteczność filtrowania z hybrydowego w zależności od liczby użytkowników i powtarzających się cech. Pogrubiona linia przedstawia działanie filtrowania kolaboratywnego.

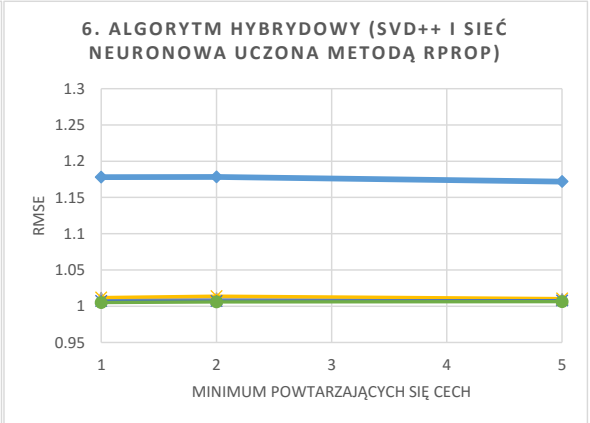

1. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

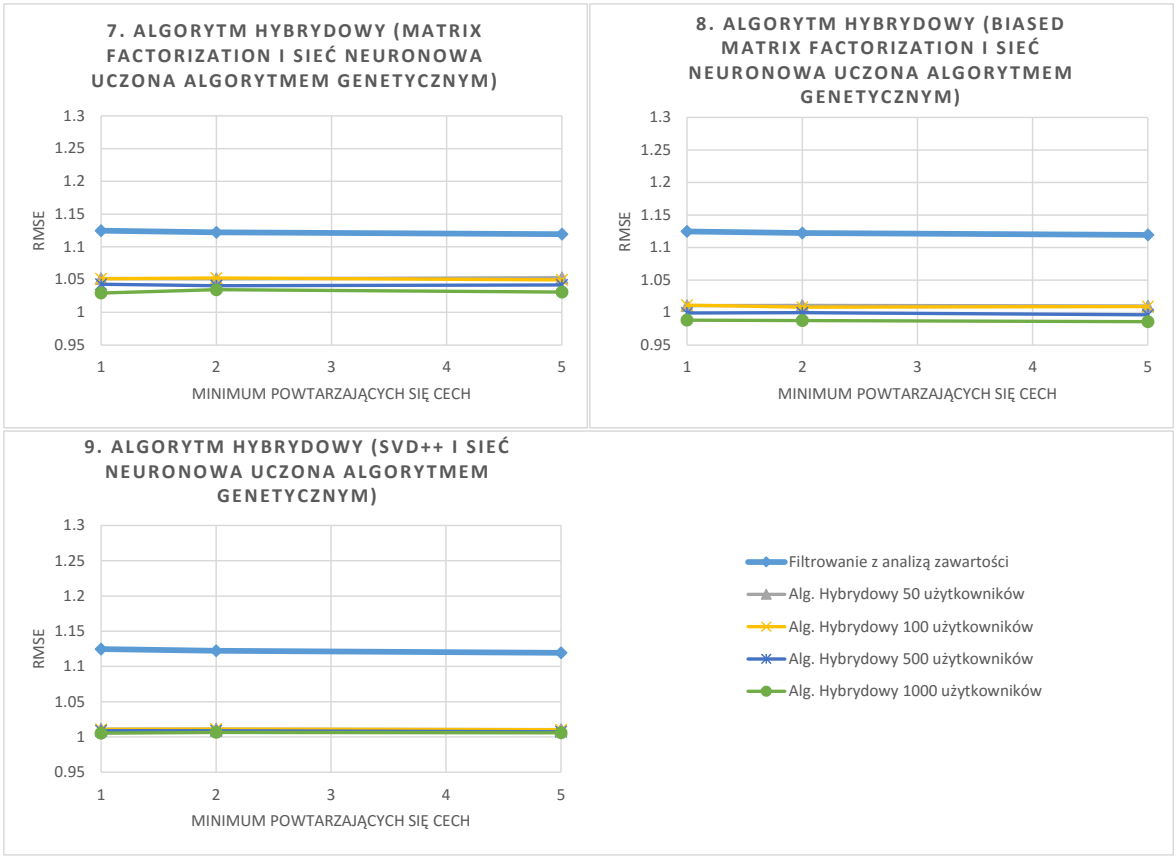

2.ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH


3. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

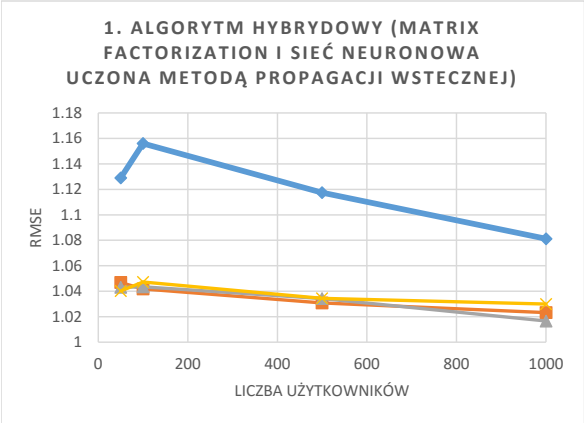

4. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

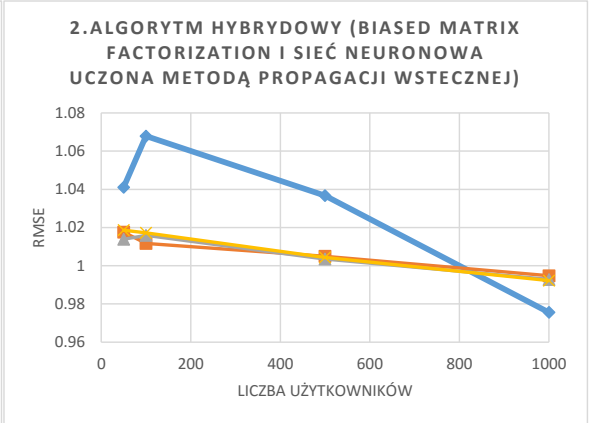

5.ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

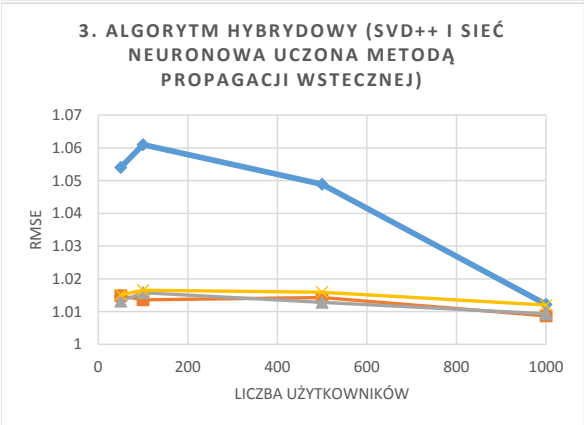

6. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
1
2
3
4
5
MINIMUM POWTARZAJĄCYCH SIĘ CECH

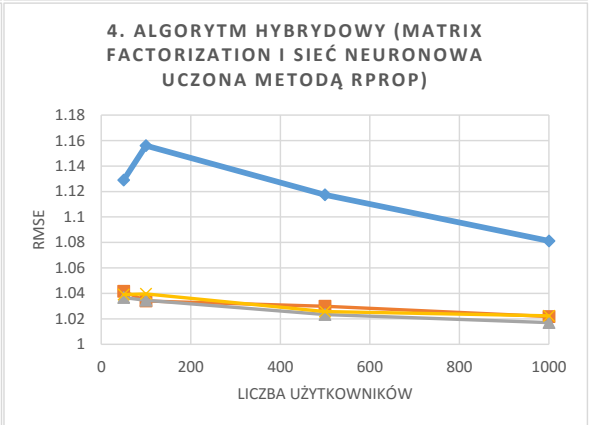


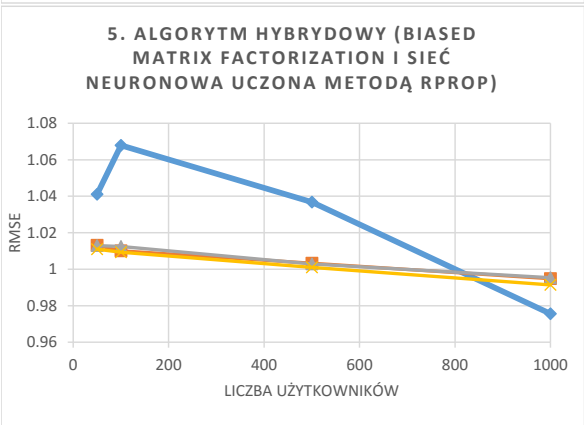
Rys. 4.16: Porównanie algorytmu hybrydowego z analizą zawartości na podstawie danych z bazy AmazonMeta. Skuteczność filtrowania z hybrydowego w zależności od liczby użytkowników i powtarzających się cech. Pogrubiona linia przedstawia działanie filtrowania z analizą zawartości.

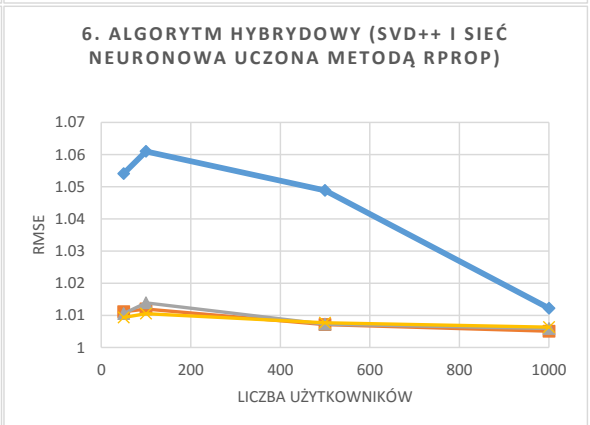

1. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.18
1.16
1.14
1.12
1.1
1.08
1.06
1.04
1.02
1
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW


2.ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.08
1.06
1.04
1.02
1
0.98
0.96
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW


3. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ PROPAGACJI WSTECZNEJ)
RMSE
1.07
1.06
1.05
1.04
1.03
1.02
1.01
1
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW


4. ALGORYTM HYBRYDOWY (MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.18
1.16
1.14
1.12
1.1
1.08
1.06
1.04
1.02
1
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW


5. ALGORYTM HYBRYDOWY (BIASED MATRIX FACTORIZATION I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.08
1.06
1.04
1.02
1
0.98
0.96
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW


6. ALGORYTM HYBRYDOWY (SVD++ I SIEĆ NEURONOWA UCZONA METODĄ RPROP)
RMSE
1.07
1.06
1.05
1.04
1.03
1.02
1.01
1
0
200
400
600
800
1000
LICZBA UŻYTKOWNIKÓW

Rys. 4.17: Porównanie algorytmu hybrydowego z kolaboratywnym na podstawie danych z bazy AmazonMeta. Skuteczność filtrowania z hybrydowego w zależności od liczby użytkowników i powtarzających się cech. Pogrubiona linia przedstawia działanie filtrowania kolaboratywnego.

Rozdział 5

Wnioski

Dodatek A

Appendix 1

Spis rysunków

Spis wzorów

Spis algorytmów

Bibliografia

[1] About the Music Genome Project. `http://www.pandora.com/about/mgp`. Data dostępu: 2016-07-19.

[2] Adomavicius G., Tuzhilin A. 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.

[3] Allegro – korzystanie z systemu rekomendacji. `http://faq.allegro.pl/artykul/27613/korzystanie-z-systemu-rekomendacji`. Data dostępu: 2016-07-19.

[4] Amazon meta. `https://snap.stanford.edu/data/amazon-meta.html`. Data dostępu: 2016-11-25.

[5] Avron H., Kale S., Kasiviswanathan S., Sindhwani V. Efficient and practical stochastic subgradient descent for nuclear norm regularization. `https://www.cs.cmu.edu/~yuxiangw/docs/SSGD.pdf`. Data dostępu: 2016-09-07.

[6] Basu C., Hirsh H., Cohen W. et al. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*, pages 714–720, 1998.

[7] Bell R., Koren Y., Volinsky C. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.

[8] Bottou L. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[9] Celma O. *The Long Tail in Recommender Systems*, pages 87–107. Springer-Verlag Berlin Heidelberg, 2010.

[10] Cheng J., Liu Y., Zhang H., Wu X., Chen F. A new recommendation algorithm based on user's dynamic information in complex social network. *Mathematical Problems in Engineering*, 2015, 2015.

[11] Claypool M., Gokhale A., Miranda T., Murnikov P., Netes D., Sartin M. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[12] Davidson J., Liebald B., Liu J., Nandy P., Van Vleet T., Gargi U., Gupta S., He Y., Lambert M., Livingston B. et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[13] Desrosiers C., Karypis G. *Recommender Systems Handbook*, chapter A Comprehensive Survey of Neighborhood-based Recommendation Methods, pages 107–144. Springer, New York Dordrecht Heidelberg London, 2010.

[14] Filmweb – najczęściej zadawane pytania. `http://www.filmweb.pl/help`. Data dostępu: 2016-07-19.

[15] Gantner Z., Rendle S., Drumond L., Freudenthaler C. Mymedialite recommender system library. `http://www.mymedialite.net/`. Data dostępu: 2016-09-05.

[16] Gantner Z., Rendle S., Freudenthaler C., Schmidt-Thieme L. Mymedialite: a free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[17] Gupta P., Goel A., Lin J., Sharma A., Wang D., Zadeh R. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 505–514. ACM, 2013.

[18] Harper F. M., Konstan J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[19] Haykin S. Neural networks: A comprehensive foundation: Macmillan college publishing company. *New York*, 1994.

[20] Hertz J. A., Krogh A. S., Palmer R. G., Jankowski S. *Wstęp do teorii obliczeń neuronowych*. Wydawnictwa Naukowo-Techniczne, 1993.

[21] Huttner J. From Tapestry to SVD: A survey of the algorithms that power recommender system. Master's thesis, Haverford College Department of Computer Science, 05 2009.

[22] Huynh T., Hoang K. Modeling collaborative knowledge of publishing activities for research recommendation. In *International Conference on Computational Collective Intelligence*, pages 41–50. Springer, 2012.

[23] Hyndman R. J., Koehler A. B. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[24] IMDb database statistics. `http://www.imdb.com/stats`. Data dostępu: 2016-07-19.

[25] Ji K., Sun R., Shu W., Li X. Next-song recommendation with temporal dynamics. *Knowledge-Based Systems*, 88:134–143, 2015.

[26] Kirillov A. AForge.NET framework. `http://www.aforgenet.com/framework/`. Data dostępu: 2016-09-05.

[27] Kirillov A. AForge.NET framework – evolutionarylearning class. `http://www.aforgenet.com/framework/docs/html/cc8bebc5-da54-5c56-6ddf-6a93aec7b9cd.htm`. Data dostępu: 2016-09-06.

[28] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[29] Koren Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1, 2010.

[30] Koren Y., Bell R. *Recommender Systems Handbook*, chapter Advances in Collaborative Filtering, pages 145–186. Springer, New York Dordrecht Heidelberg London, 2010.

[31] Koren Y., Bell R., Volinsky C. et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[32] Kwater T. Algorytmy uczenia sieci neuronowych. `http://www.neurosoft.edu.pl/media/pdf/tkwater/sztuczna_inteligencja/2_alg_ucz_ssn.pdf`. Data dostępu: 2016-09-06.

[33] Lemire D., Maclachlan A. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[34] Leskovec J., Adamic L. A., Huberman B. A. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

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

[36] Lops P., de Gemmis M., Semeraro G. *Recommender Systems Handbook*, chapter Content-based Recommender Systems: State of the Art and Trends, pages 73–100. Springer, New York Dordrecht Heidelberg London, 2010.

[37] Maleszka M., Mianowska B., Nguyen N. T. A method for collaborative recommendation using knowledge integration tools and hierarchical structure of user profiles. *Knowledge-Based Systems*, 47:1–13, 2013.

[38] Melville P., Mooney R. J., Nagarajan R. Content-boosted collaborative filtering for improved recommendations. In *Aaai/iaai*, pages 187–192, 2002.

[39] Montana D. J., Davis L. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

[40] Mymedialite: Example experiments. `http://www.mymedialite.net/examples/datasets.html`. Data dostępu: 2016-07-24.

[41] Netflix Prize (I tried to resist, but...). `https://www.snellman.net/blog/archive/2006-10-15-netflix-prize.html`. Data dostępu: 2016-07-08.

[42] Netflix Prize: forum. `http://www.netflixprize.com/community/viewtopic.php?id=1537`. Data dostępu: 2016-07-08.

[43] Netflix Prize Rankings. `http://www.hackingnetflix.com/2006/10/netflix_prize_r.html`. Data dostępu: 2016-07-08.

[44] Netflix Prize Rules. `http://www.netflixprize.com//rules`. Data dostępu: 2016-07-08.

[45] Osowski S. *Sieci neuronowe w ujęciu algorytmicznym.* Wydawnictwa Naukowo-Techniczne, 1996.

[46] Pariser E. *The filter bubble: What the Internet is hiding from you.* Penguin UK, 2011.

[47] Pena-Reyes C. A., Sipper M. Evolutionary computation in medicine: an overview. *Artificial Intelligence in Medicine*, 19(1):1–23, 2000.

[48] Pogue D. A Stream of Movies, Sort of Free. *The New York Times*, 2007.

[49] Rendle S., Schmidt-Thieme L. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[50] Ricci F., Rokach L., Shapira B. *Recommender Systems Handbook*, chapter Introduction to Recommender Systems Handbook, pages 1–35. Springer, New York Dordrecht Heidelberg London, 2010.

[51] Riedmiller M., Braun H. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference On*, pages 586–591. IEEE, 1993.

[52] Riedmiller M., Rprop I. Rprop-description and implementation details. 1994.

[53] Rohrmann T. Computing recommendations at extreme scale with apache flink. *http://data-artisans.com/computing-recommendations-at-extreme-scale-with-apache-flink*, 2015.

[54] Rubens N., Kaplan D., Sugiyama M. Active learning in recommender systems. In Kantor P., Ricci F., Rokach L., Shapira B., editors, *Recommender Systems Handbook*, pages 735–767. Springer, 2011.

[55] Salakhutdinov R., Mnih A. Probabilistic matrix factorization. In *NIPS*, volume 20, pages 1–8, 2011.

[56] Sarwar B., Karypis G., Konstan J., Riedl J. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[57] Schafer J., Frankowski D., Herlocker J., Sen S. *The Adaptive Web*, chapter Collaborative filtering recommender systems, page 291–324. Springer Berlin / Heidelberg, 2007.

[58] Sharma R., Singh R. Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey. *Indian Journal of Science and Technology*, 9(20), 2016.

[59] Timothy M. Sieci neuronowe w praktyce. *WNT, Warszawa*, 1996.

[60] Willmott C. J., Matsuura K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[61] Yahoo! music. `https://www.yahoo.com/music/`. Data dostępu: 2016-11-05.

[62] Zhang H.-R., Min F., He X., Xu Y.-Y. A hybrid recommender system based on user-recommender interaction. *Mathematical Problems in Engineering*, 2015, 2015.