# System rekomendacyjny polecający książki w bibliotece

Patryk Nikonowicz,
Ewa Pasterz,
Piotr Skibiński,
Paulina Szostek

Kwiecień 2022

## Spis treści

# 1 Opis tematu

Wybrany przez nas temat polega na stworzeniu systemu rekomendacyjnego polecającego książki w bibliotece. Nasz system będzie zaimplementowany w formie strony internetowej połączonej z bazą danych. By otrzymywać rekomendacje trzeba będzie założyć indywidualne konto czytelnika. Dane o użytkowniku i książkach będą zapamiętywane między sesjami.

W naszym systemie będzie dostępnych ok. 1000 książek, które użytkownik będzie mógł oznaczyć jako przeczytane i je ocenić. Wszystkie książki posiadają informacje takie jak tytuł, autor, wydawnictwo, grupa docelowa czy język.

Do tworzenia rekomendacji przyjęliśmy podejście content-based. Książki będą polecane użytkownikowi na podstawie już wystawionych przez niego ocen i podobieństwa między ocenionymi książkami a tymi jeszcze nieprzeczytanymi.

# 2 Model danych

Nasze dane będą reprezentować rzeczywiste książki. Wyodrębniliśmy 13 atrybutów, które będziemy przetrzymywać dla każdej pozycji i zgodnie z którymi będziemy tworzyć rekomendacje. Poniżej wymieniliśmy atrybuty, które będziemy definiować:

1. Tytuł
2. Autor
3. Wydawca
4. Gatunki
5. Grupa docelowa
6. Poruszone tematy
7. Liczba stron
8. Język
9. Kraj pochodzenia

10. Data wydania
11. Ocena
12. Ogólna liczba wypożyczeń
13. Liczba wypożyczeń w miesiącu

Poruszane tematy są innym atrybutem niż gatunki, choć mogą wydawać się podobne. Gatunek to przykładowo romans czy horror, a poruszane tematy to wampiry czy wojna stuletnia. Ocena będzie wyliczana jako średnia ocen wystawionych przez użytkowników. Tytuł nie będzie uwzględniany w algorytmie rekomendacyjnym. Dane o książkach będą pobrane z m.in. Google Books API i ewentualnie uzupełnione o brakujące informacje.

# 3 Użytkownik

Każda osoba chcąca skorzystać z naszego systemu będzie musiała założyć swój indywidualny profil. Do założenia konta potrzebny będzie unikalny adres e-mail. Będzie istniała możliwość zarejestrowania się z użyciem już istniejącego konta Google.

Zalogowany użytkownik będzie mógł oznaczać książki obecne w naszej bazie jako przeczytane i będzie mógł im wystawić ocenę. Nasz system będzie zapamiętywał oceny użytkownika i na tej podstawie będziemy określać preferencje owego czytelnika.

# 4 Użyte technologie

Głowna implementacja modelu będzie z użyciem języka Python przy użyciu biblioteki Surprise, ewentualnie Scikit. Dane do biblioteki będą pobierane z dostępnych otwartych API - Google Books API, Goodreads API itd. W razie potrzeby dodatkowe dane mogą być pobierane z innych źródeł danych.

Frontend naszej aplikacji będzie wykonany jako strona internetowa przy użyciu Javascript. Biblioteka, której będziemy używać to React. Backend naszej aplikacji będzie wykonany w C# przy użyciu ASP.NET Core. Aby umożliwić logowanie się użytkowników będzie przeprowadzone połączenie z dostawcą tożsamości. Dostawca tożsamości będzie działał na podstawie biblioteki IdentityServer4 lub Keycloak. Przewidziana jest możliwość logowania

przez konta, które użytkownicy mogą założyć na stronie lub przez ich konto Google.

# 5 Algorytm

## 5.1 Opis

Nasz algorytm będzie przyjmował różne strategie dla nowych użytkowników i tych, którzy już mają ocenione książki. Gdy nowy użytkownik się dopiero zaloguje i nie ma jeszcze wystawionych ocen, będziemy polecać mu popularne i wysoko oceniane książki.

Nasza strategia dla użytkowników z większym stażem przyjmuje, że książki będą polecane na podstawię najwyżej ocenionych książek, które przeczytał użytkownik. W przypadku dużej liczby takich książek, będziemy brali te ostatnio przeczytane. Proces porównawczy będzie polegał na znalezieniu książek jak najbardziej podobnych do tych lubianych przez użytkownika, wykorzystując hybrydowo podobieństwo cosinusowe i odległość Hamminga. Odległość Hamminga będzie wykorzystana do określenia podobieństwa poruszonych tematów i kategorii, gdyż te atrybuty mogą zawierać w sobie wiele wartości. Podobieństwo cosinusowe będzie wykorzystane do określenia podobieństwa pozostałych wartości, z wyłączeniem oceny i częstości wypożyczeń. Wartości tych kolumn będą wykorzystywane do posortowania znalezionych rekomendacji, tak by najpierw pokazać książki najpopularniejsze.

Podobieństwo cosinusowe polega na obliczeniu cosinus kąta między dwoma wektorami. Funkcja cosinus może przyjmować wartości z zakresu [-1,1]. Im większa wartość tym większe podobieństwo między wektorami. Podobieństwo wylicza się z następującego wzoru:

$$D(x, y) = cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{1}$$

Wektory x i y będą przetrzymywać informacje o atrybutach książek. Atrybuty będą sprowadzone do postaci liczbowych, tak że dana wartość książki ocenionej przez użytkownika będzie równa 1, a każda inna będzie równa 0.

Odległość Hamminga natomiast określa liczbę różnych wartości pomiędzy dwoma wektorami. Gatunki i poruszane tematy będziemy zamieniać na cyfry 0 bądź 1, w zależności od tego czy występują w danej książce. Te cechy będą też miały zawsze stałą kolejność, na przykład: romans, horror, kryminał.

Kryminał, bez dodatkowych gatunków będzie miał wartość 001, a kryminał z elementami horroru: 011. Na takim ciągu liczbowym będzie liczona odległość Hamminga.

By policzyć finalne podobieństwo między książkami, będziemy brali pod uwagę podobieństwo cosinusowe oraz odległość Hamminga. Problemem jest to, że żeby książki były jak najbardziej podobne to potrzebujemy jak największego podobieństwa cosinusowego i jak najmniejszej odległości Hamminga, zatem nie możemy po prostu zsumować tych wartości. Dlatego zamiast zwykłej wartości Hamminga będziemy brali jej odwrotność, czyli liczbę elementów, których nie trzeba zmieniać - w takim wypadku im większa liczba tym lepiej. Patrząc na podany wcześniej przykład, z kryminałem i horrorystycznym kryminałem: klasyczna odległość Hamminga byłaby równa 1, a nasza „odwrócona" będzie równa 2.

Zaimplementujemy również lekko odmienną strategię. Wciąż wykorzystywane by były odległość Hamminga oraz podobieństwo cosinusowe, lecz zamiast liczyć je dla paru najwyżej ocenianych książek będziemy je rozważać dla ich „średniej". „Średnia z książek" to książka o cechach najczęściej występujących wśród ulubionych książek użytkownika. Użytkownik będzie miał możliwość wybrać, które z tych rekomendacji chce wyświetlić. Jako że odległość Hamminga i podobieństwo cosinusowe będą już zaimplementowane, użytkownik będzie mógł też wyświetlać rekomendacje na podstawie jednej, wybranej przez niego, książki.

## 5.2 Pseudokod

Poniżej znajduje się nasz algorytm zapisany w pseudokodzie. Nie uwzględniliśmy funkcji szukania najpopularniejszych książek oraz ulubionych książek użytkownika, gdyż sprowadzają się one jedynie do sortowania po ocenie i wypożyczeniach (w przypadku pierwszym) lub dacie przeczytania (w drugim przypadku).

```
Recommend(książka, n)
    x ← ToVector(książka, książka)
    xG, xT ← ToHemming(książka)
    similarity ← słownik przypisujący każdej książce
        stopień podobieństwa
    dla każdej nieprzeczytanej książki notRead
        y ← ToVector(notRead)
```

```
        yG, yT ← ToHamming(notRead)
        podobieństwo ← CosSimilarity(x,y) +
            HammingSimilarity(xG,yG) + HammingSimilarity(xT,yT)
        wstaw do similarity parę (notRead, podobieństwo)
    posortuj similarity malejąco
    zwróć n pierwszych książek z similarity

ToVector(x, comparison)
    wynik ← pusty wektor
    dla każdego a - atrybut książki (poza gatunkami i tematami)
        jeśli x.a == comparison.a to
            wstaw 1 do wynik
        w.p.p. wstaw 0 do wynik
    zwróć wynik
```

W powyższej funkcji należy zwrócić uwagę na porównanie atrybutów. Należy zaimplementować własne porównanie dat, by brać pod uwagę nie konkretne daty, tylko dłuższe okresy czasu, gdyż jeśli ktoś przeczytał książkę wydaną 03/12/2003, to nie interesują go inne książki wydane trzeciego grudnia tego roku, a raczej książki wydane w pierwszej dekadzie XXI wieku. My będziemy porównywali właśnie dekady.

```
ToHamming(x)
    wynikG ← pusty wektor
    wynikT ← pusty wektor
    dla każdego gatunku g
        jeśli x.gatunki zawiera g to
            wstaw 1 do wynikG
        w.p.p. wstaw 0 do wynikG
    dla każdego poruszanego tematu t
        jeśli x.tematy zawiera t to
            wstaw 1 do wynikT
        w.p.p. wstaw 0 do wynikT
    zwróć (wynikG, wynikT)

CosSimilarity(x,y)
    zwróć x·y / ||x||·||y||
```

W powyższej funkcji licznik jest iloczynem skalarnym wektorów, a mianownik iloczynem ich norm.

```
HammingSimilarity(x,y)
     i ← długość wektorów x i y
     wynik ← 0
     dopóki i>0
          jeśli x[i]==y[i] to
               wynik++
          i--
     zwróć wynik
```