Sistem preporuke za aplikaciju eBiblioteke

Ovaj dokument opisuje način rada RecommenderService komponente u okviru sistema za biblioteku. Sistem koristi kombinaciju modela zasnovanih na sadržaju (content-based filtering) i korisničkog ponašanja da bi generisao personalizirane preporuke knjiga.

1. Opšti princip rada

Sistem za preporuku knjiga kombinuje analizu sadržaja knjiga i korisničkih interakcija kako bi predložio relevantne naslove. Svaka knjiga se pretvara u numerički vektor koji opisuje njen naslov, autora, žanrove i opis, što omogućava precizno mjerenje sličnosti među knjigama.

Korisnički profil formira se analizom posuđenih i ocijenjenih knjiga, pri čemu ocjene imaju veći utjecaj. Profil predstavlja sažetu sliku korisnikovih preferencija, a sistem ga koristi da pronađe knjige čiji su vektori najsličniji profilu korisnika. Knjige koje je korisnik već posudio ili ocijenio se isključuju kako bi preporuke bile nove i relevantne.

```
var reviews = await _db.Set<BookReview>()
    .Where(r => r.UserId == userId && r.IsApproved && !r.IsDenied)
    .Select(r => new { r.BookId, r.Rating })
    .ToListAsync(ct);

foreach (var r in reviews)
{
    if (_bookVectors.ContainsKey(r.BookId))
    {
        float w = Math.Clamp(r.Rating, 1, 5) / 5f;
        weights[r.BookId] = weights.GetValueOrDefault(r.BookId) + 2f * w;
    }
}

var loans = await _db.Set<BookLoan>()
    .Where(l => l.UserId == userId)
    .Select(l => new { l.BookId })
    .ToListAsync(ct);

foreach (var l in loans)
{
    if (_bookVectors.ContainsKey(l.BookId))
    {
        weights[l.BookId] = weights.GetValueOrDefault(l.BookId) + 1.5f;
    }
}
```

Dohvaćanje relevantnih podataka za kreiranje bolje preporuke

2. Izgradnja modela

Sistem koristi model koji se gradi samo jednom, čime se štedi vrijeme i resursi. Za svaku knjigu, tekstualni podaci poput naslova, imena autora, žanrova i opisa transformišu se u numeričke vektore koristeći ML.NET. Ovi vektori predstavljaju suštinu sadržaja knjige u višedimenzionalnom prostoru, omogućavajući da se knjige međusobno upoređuju po sličnosti. Proces izgradnje modela uključuje nekoliko koraka: prvo se tekstualni atributi feature-izuju, zatim se svi rezultati kombinuju u jedan jedinstveni vektor, a na kraju se primjenjuje normalizacija kako bi vektori bili usporedivi. Ovakav pristup omogućava da sistem stvori preciznu matematičku reprezentaciju svake knjige, što je temelj za kasniju analizu i generisanje personaliziranih preporuka.

```
private void BuildModelIfNeeded()
{
    if (_bookVectors != null) return;

    lock (_sync)
    {
        if (_bookVectors != null) return;

        _ml = new MLContext(seed: 1);

        var rows = LoadBookRows();
        if (rows.Count == 0)
        {
            _bookVectors = new();
            return;
        }

        var data = _ml.Data.LoadFromEnumerable(rows);

        var pipeline =
            _ml.Transforms.Text.FeaturizeText("TitleFeats", nameof(BookRow.Title))
            .Append(_ml.Transforms.Text.FeaturizeText("AuthorFeats", nameof(BookRow.Authors)))
            .Append(_ml.Transforms.Text.FeaturizeText("GenreFeats", nameof(BookRow.Genres)))
            .Append(_ml.Transforms.Text.FeaturizeText("DescFeats", nameof(BookRow.Description)))
            .Append(_ml.Transforms.Concatenate("Features", "TitleFeats", "AuthorFeats", "GenreFeats", "DescFeats"))
            .Append(_ml.Transforms.NormalizeLpNorm("Features"));

        var model = pipeline.Fit(data);
        var transformed = model.Transform(data);

        var vectors = _ml.Data.CreateEnumerable<VectorRow>(transformed, reuseRowObject: false);
        _bookVectors = vectors.ToDictionary(v => v.BookId, v => v.Features);
    }
}
```

Pipeline za transformaciju podataka o knjigama

3. Formiranje korisničkog profila

Korisnički profil predstavlja saženu sliku preferencija svakog korisnika u obliku numeričkog vektora. Sistem koristi informacije iz prethodnih interakcija, prvenstveno ocjene koje je korisnik dao knjigama i istoriju posudbi. Ocjene imaju veći utjecaj jer jasno odražavaju stav korisnika prema pročitanim knjigama, dok posudbe doprinose u manjoj mjeri, ali i one pružaju vrijedan uvid u interes

korisnika. Kombinovanjem težinskih doprinosa svih knjiga koje je korisnik ocijenio ili posudio, formira se težinski prosjek vektora koji čini korisnički profil. Ovaj profil potom služi za pronalaženje knjiga s najvećom sličnosti. Ako korisnik nema nikakvu prethodnu aktivnost, sistem automatski koristi cold start strategiju kako bi preporuke i dalje bile relevantne.

```
var loans = await _db.Set<BookLoan>()
    .Where(l => l.UserId == userId)
    .Select(l => new { l.BookId })
    .ToListAsync(ct);

foreach (var l in loans)
{
    if (_bookVectors.ContainsKey(l.BookId))
    {
        weights[l.BookId] = weights.GetValueOrDefault(l.BookId) + 1.5f;
    }
}

if (weights.Count == 0)
    return await ColdStartAsync(take, ct);

var profile = WeightedAverage(weights);
```

Kreiranje profila nakon pregledavanja interakcija iz baze (zadnja linija)

```
private static float[] WeightedAverage(Dictionary<int, float> weights)
{
    var firstId = weights.Keys.First();
    if (!_bookVectors.TryGetValue(firstId, out var first))
        return Array.Empty<float>();

    var sum = new float[first.Length];
    float wsum = 0;

    foreach (var (id, w) in weights)
    {
        if (!_bookVectors.TryGetValue(id, out var vec)) continue;

        for (int i = 0; i < vec.Length; i++)
            sum[i] += vec[i] * w;

        wsum += w;
    }

    if (wsum > 1e-6f)
        for (int i = 0; i < sum.Length; i++)
            sum[i] /= wsum;

    return sum;
}
```

Proces računanja težinskog prosjeka

4. Cold start

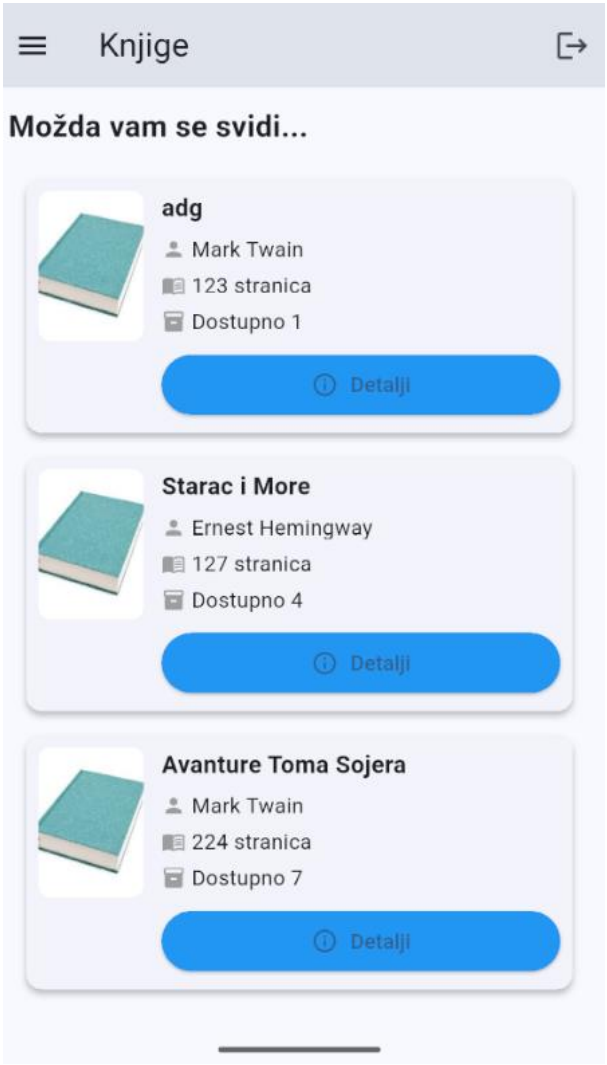
U situacijama kada model još nije izgrađen ili korisnik nema prethodnu historiju interakcija, sistem koristi cold start strategiju kako bi osigurao relevantne preporuke. Umjesto da korisnik dobije praznu listu, sistem se oslanja na globalne podatke iz baze. Prvo se uzimaju knjige s najboljim ocjenama, uzimajući u obzir prosječnu ocjenu i broj recenzija, jer one obično predstavljaju kvalitetne i popularne naslove. Ako i dalje nije moguće generisati dovoljan broj preporuka, sistem razmatra najčešće posuđivane knjige, koje odražavaju interes šire publike. Kao posljednji korak, ako ni to nije dovoljno, koristi se fallback lista prvih nekoliko knjiga po abecednom redu. Ovakav pristup osigurava da korisnik uvijek dobije korisne i relevantne prijedloge, čak i kada nema vlastite historije.

```
private async Task<List<BookDto>> ColdStartAsync(int take, CancellationToken ct)
{
    var topRated = await _db.Set<BookReview>()
        .Where(r => r.IsApproved && !r.IsDenied)
        .GroupBy(r => r.BookId)
        .Select(g => new { BookId = g.Key, AvgRating = g.Average(x => x.Rating), Count = g.Count() })
        .OrderByDescending(x => x.AvgRating)
        .ThenByDescending(x => x.Count)
        .Take(take * 2)
        .ToListAsync(ct);

    var ids = topRated.Select(x => x.BookId).ToList();

    var books = await _db.Set<Book>()
        .Where(b => ids.Contains(b.Id))
        .Include(b => b.Author)
        .Include(b => b.Genres)
        .Take(take)
        .ToListAsync(ct);

    if (books.Count > 0)
        return books.Select(b => _mapper.Map<BookDto>(b)).ToList();

    var mostBorrowed = await _db.Set<BookLoan>()
        .GroupBy(l => l.BookId)
        .Select(g => new { BookId = g.Key, Cnt = g.Count() })
        .OrderByDescending(x => x.Cnt)
        .Take(take)
        .ToListAsync(ct);

    ids = mostBorrowed.Select(x => x.BookId).ToList();

    books = await _db.Set<Book>()
        .Where(b => ids.Contains(b.Id))
        .Include(b => b.Author)
        .Include(b => b.Genres)
        .ToListAsync(ct);

    if (books.Count > 0)
        return books.Select(b => _mapper.Map<BookDto>(b)).ToList();

    var fallbackBooks = await _db.Set<Book>()
        .Include(b => b.Author)
        .Include(b => b.Genres)
        .OrderBy(b => b.Title)
        .Take(take)
        .ToListAsync(ct);

    return fallbackBooks.Select(b => _mapper.Map<BookDto>(b)).ToList();
}
```

Metodologija kreiranja preporuka ukoliko korisnik nema zabilježenih interakcija

5. Primjer iz aplikacije

Na idućoj slici možemo vidjeti kako izgleda recommender sistem u svojoj cjelosti. Ova sekcija se nalazi u listi knjiga, na samom dnu ekrana.



Recommender sistem izlistava 3 preporučene knjige