Elshiekh Ahmed

Razvoj softvera 2

12.01.2026

Opis implementacije recommendation sistema:

Sistem koristi Microsoft.ML Matrix Factorization algoritam za generisanje personalizovanih preporuka filmova. Sastoji se od dva ML modela:

1. Genre Model - Pronalazi slične filmove na osnovu zajedničkih žanrova. Računa Jaccard sličnost između filmova (broj zajedničkih žanrova / ukupan broj žanrova) i koristi Matrix Factorization za učenje latentnih faktora.

2. User Model - Generiše personalizovane preporuke za korisnika. Trenira se na osnovu korisnikovih recenzija (ocjene 1-5) i watchlist-e (pogledani filmovi dobijaju ocjenu 4, planirani 3).

Kada korisnik otvori Search ekran, na dnu se prikazuje sekcija "Preporučeno za tebe:" sa filmovima koji su preporučeni na osnovu njegovih prethodnih interakcija (recenzije i watchlist). Sistem kombinuje lokalne ML preporuke sa TMDb API preporukama za bolje rezultate.

Prilikom prvog pokretanja aplikacije vrši se treniranje modela, pa je potrebno sačekati određeno vrijeme da se isto završi. Modeli se čuvaju u fajlove (genre-model.zip i user-model.zip) za brže učitavanje pri sljedećim pokretanjima.

Putanja do source code-a:

\staGledas\staGledas.Service\Services\RecommenderService.cs

Printscreen glavne logike - Genre Model Training:

```
2 references
private void TrainGenreModel()
{
    var films = _context.Filmovi
        .Include(f => f.FilmoviZanrovi)
        .ToList();

    var data = new List<MovieEntry>();

    foreach (var film in films)
    {
        var filmGenreIds = film.FilmoviZanrovi.Select(fz => fz.ZanrId).ToHashSet();

        var relatedFilms = films
            .Where(f => f.Id != film.Id && f.FilmoviZanrovi.Any(fz => filmGenreIds.Contains(fz.ZanrId)));

        foreach (var relatedFilm in relatedFilms)
        {
            var relatedGenreIds = relatedFilm.FilmoviZanrovi.Select(fz => fz.ZanrId).ToHashSet();
            var sharedGenres = filmGenreIds.Intersect(relatedGenreIds).Count();
            var totalGenres = filmGenreIds.Union(relatedGenreIds).Count();
            var similarity = totalGenres > 0 ? (float)sharedGenres / totalGenres : 0f;

            data.Add(new MovieEntry
            {
                FilmId = (uint)film.Id,
                RelatedFilmId = (uint)relatedFilm.Id,
                Label = similarity
            });
        }
    }

    if (!data.Any())
    {
        data.Add(new MovieEntry { FilmId = 1, RelatedFilmId = 2, Label = 0.5f });
    }

    var trainData = _mlContext!.Data.LoadFromEnumerable(data);

    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = nameof(MovieEntry.FilmId),
        MatrixRowIndexColumnName = nameof(MovieEntry.RelatedFilmId),
        LabelColumnName = "Label",
        LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass,
        Alpha = 0.01,
        Lambda = 0.025,
        NumberOfIterations = 20,
        C = 0.00001,
        Quiet = true
    };

    var est = _mlContext.Recommendation().Trainers.MatrixFactorization(options);
    _genreModel = est.Fit(trainData);

    using var stream = new FileStream(GenreModelPath, FileMode.Create, FileAccess.Write, FileShare.Write);
    _mlContext.Model.Save(_genreModel, trainData.Schema, stream);
}
```

Printscreen glavne logike - User Model Training

```
2 references
private void TrainUserModel()
{
    var interactions = _context.Recenzije
        .Select(r => new { r.KorisnikId, r.FilmId, Rating = (float)r.Ocjena })
        .ToList();

    var watchlistInteractions = _context.Watchlist
        .Select(w => new { w.KorisnikId, w.FilmId, Rating = w.Pogledano == true ? 4f : 3f })
        .ToList();

    var data = interactions
        .Select(i => new UserMovieEntry
        {
            KorisnikId = (uint)i.KorisnikId,
            FilmId = (uint)i.FilmId,
            Label = i.Rating / 5f
        })
        .Concat(watchlistInteractions.Select(w => new UserMovieEntry
        {
            KorisnikId = (uint)w.KorisnikId,
            FilmId = (uint)w.FilmId,
            Label = w.Rating / 5f
        }))
        .ToList();

    if (!data.Any())
    {
        data.Add(new UserMovieEntry { KorisnikId = 1, FilmId = 1, Label = 0.8f });
    }

    var trainData = _mlContext!.Data.LoadFromEnumerable(data);

    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = nameof(UserMovieEntry.KorisnikId),
        MatrixRowIndexColumnName = nameof(UserMovieEntry.FilmId),
        LabelColumnName = "Label",
        LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass,
        Alpha = 0.01,
        Lambda = 0.025,
        NumberOfIterations = 20,
        C = 0.00001,
        Quiet = true
    };

    var est = _mlContext.Recommendation().Trainers.MatrixFactorization(options);
    _userModel = est.Fit(trainData);

    using var stream = new FileStream(UserModelPath, FileMode.Create, FileAccess.Write, FileShare.Write);
    _mlContext.Model.Save(_userModel, trainData.Schema, stream);
}
```

Printscreen glavne logike - Get Recommendations For User

```
public async Task<List<TMDbMovie>> GetRecommendationsForUserAsync(int korisnikId, int count = 10)
{
    var userReviews = await _context.Recenzije
        .Where(r => r.KorisnikId == korisnikId)
        .Include(r => r.Film)
        .OrderByDescending(r => r.Ocjena)
        .ToListAsync();

    var matchlist = await _context.Watchlist
        .Where(w => w.KorisnikId == korisnikId && w.Pogledano == true)
        .Include(w => w.Film)
        .ToListAsync();

    var userLikes = await _context.FilmoviLajkovi
        .Where(fl => fl.KorisnikId == korisnikId)
        .Include(fl => fl.Film)
        .OrderByDescending(fl => fl.DatumLajka)
        .ToListAsync();

    var totalInteractions = userReviews.Count + matchlist.Count + userLikes.Count;

    if (totalInteractions == 0)
    {
        return await _tmdbService.GetTrendingMoviesAsync();
    }

    var likedMovies = userReviews
        .Where(r => r.Ocjena >= 4 && r.Film?.TmdbId != null)
        .Select(r => r.Film!)
        .ToList();

    if (!likedMovies.Any())
    {
        likedMovies = userLikes
            .Where(fl => fl.Film?.TmdbId != null)
            .Select(fl => fl.Film!)
            .ToList();
    }

    if (!likedMovies.Any())
    {
        likedMovies = matchlist
            .Where(w => w.Film?.TmdbId != null)
            .Select(w => w.Film!)
            .ToList();
    }

    if (!likedMovies.Any())
    {
        return await _tmdbService.GetPopularMoviesAsync();
    }

    var topMovie = likedMovies.First();
    var recommendations = await _tmdbService.GetTMDbRecommendationsAsync(topMovie.TmdbId!.Value);

    if (totalInteractions >= MinimumInteractionsForML && likedMovies.Count > 1)
    {
        var secondMovie = likedMovies.Skip(1).First();
        var moreRecs = await _tmdbService.GetTMDbRecommendationsAsync(secondMovie.TmdbId!.Value);

        var existingIds = recommendations.Select(r => r.Id).ToHashSet();
        recommendations.AddRange(moreRecs.Where(r => !existingIds.Contains(r.Id)));
    }

    var seenTmdbIds = userReviews.Select(r => r.Film?.TmdbId)
        .Union(matchlist.Select(w => w.Film?.TmdbId))
        .Union(userLikes.Select(fl => fl.Film?.TmdbId))
        .Where(id => id.HasValue)
        .Select(id => id!.Value)
        .ToHashSet();

    return recommendations
        .Where(r => !seenTmdbIds.Contains(r.Id))
        .Take(count)
        .ToList();
}
```

Putanja do code-a u aplikaciji gdje se poziva recommender sistem:

Flutter Mobile App:

\staGledas\stagledas_mobile\lib\screens\search_screen.dart

Backend Controller:

\staGledas\staGledas.API\Controllers\RecommenderController.cs

Printscreen iz pokrenute aplikacije gdje se prikazuju preporuke: