1. **Opis implementacije recommender sistema**

U svrhu aplikacije korišten je content based filtering. Cilj je čitaocu preporučiti knjige slične onima koje su oni već pogledali. Ova tehnika bazira se na analiziranju sadržaja (feature-a) svake knjige i kreiranju modela koji određuje koliko su knjige slične.

Podaci se sastoje od dvije glavne grupe:

* Korisnički interesi – lista knjiga koje korisnik pogledao
* Karakteristike knjiga – dostupni atributi knjiga: autori, žanrovi, jezik, ciljne grupe, vrste sadržaja, jezik.

ML.NET koristi tehnike ekstrakcije karakteristika kako bi predstavio svaku knjigu kao vektor u višedimenzionalnom prostoru. Svaki atribut knjige postaje dimenzija u tom prostoru. Ako korisnik voli knjige iz žanra „Fantastika“ koje su napisali „Autor A“ i „Autor B“, a u bazi postoji knjiga sa sličnim karakteristikama, model će izračunati visoku sličnost između te knjige i knjiga koje je korisnik već pročitao.

1. **Putanja i printscreen source code-a glavne logike recommender sistema**

eLibrary / eLibrary.Services / Recommender / MlRecommenderService.cs

public async Task<List<Model.KnjigeDTOs.Knjige>> GetRecommendedBooks(int citalacId)

{

var allBooksQuery = eLibraryContext

.Knjiges

.Include(x => x.KnjigaAutoris).ThenInclude(x => x.Autor)

.Include(x => x.KnjigaCiljneGrupes).ThenInclude(x => x.CiljnaGrupa)

.Include(x => x.VrstaGrade)

.Include(x => x.KnjigaVrsteSadrzajas).ThenInclude(x => x.VrstaSadrzaja)

.Include(x => x.Jezik)

.AsQueryable();

var readedBookIds = eLibraryContext

.CitalacKnjigaLogs

.Where(x => x.CitalacId == citalacId)

.Select(x => x.KnjigaId)

.Distinct()

.ToList();

var readedBooks = await allBooksQuery

.Where(x => readedBookIds.Contains(x.KnjigaId))

.ToListAsync();

var readedBookData = readedBooks.Select(k => new KnjigaData

{

KnjigaId = k.KnjigaId,

Autori = string.Join(", ", k.KnjigaAutoris.Select(ka => ka.Autor.Ime + " " + ka.Autor.Prezime)),

CiljneGrupe = string.Join(", ", k.KnjigaCiljneGrupes.Select(kcg => kcg.CiljnaGrupa.Naziv)),

VrsteGrade = k.VrstaGrade.Naziv

}).ToList();

var allBookData = await allBooksQuery.Select(k => new KnjigaData

{

KnjigaId = k.KnjigaId,

Autori = string.Join(", ", k.KnjigaAutoris.Select(ka => ka.Autor.Ime + " " + ka.Autor.Prezime.OrderBy(x => x))),

CiljneGrupe = string.Join(", ", k.KnjigaCiljneGrupes.Select(kcg => kcg.CiljnaGrupa.Naziv).OrderBy(x => x)),

VrsteGrade = k.VrstaGrade.Naziv,

VrsteSadrzaja = string.Join(", ", k.KnjigaVrsteSadrzajas.Select(kcg => kcg.VrstaSadrzaja.Naziv).OrderBy(x => x)),

Jezik = k.Jezik.Naziv,

Naslov = k.Naslov

}).ToListAsync();

var pipeline = mlContext.Transforms.Text.FeaturizeText("AutoriFeaturized", nameof(KnjigaData.Autori))

.Append(mlContext.Transforms.Text.FeaturizeText("CiljneGrupeFeaturized", nameof(KnjigaData.CiljneGrupe)))

.Append(mlContext.Transforms.Text.FeaturizeText("VrsteGradeFeaturized", nameof(KnjigaData.VrsteGrade)))

.Append(mlContext.Transforms.Text.FeaturizeText("VrsteSadrzajaFeaturized", nameof(KnjigaData.VrsteSadrzaja)))

.Append(mlContext.Transforms.Text.FeaturizeText("JezikFeaturized", nameof(KnjigaData.Jezik)))

.Append(mlContext.Transforms.Text.FeaturizeText("NaslovFeaturized", nameof(KnjigaData.Naslov)))

.Append(mlContext.Transforms.Concatenate("Features",

"AutoriFeaturized",

"CiljneGrupeFeaturized",

"VrsteGradeFeaturized",

"VrsteSadrzajaFeaturized",

"JezikFeaturized"));

var model = pipeline.Fit(mlContext.Data.LoadFromEnumerable(allBookData));

var predictions = new List<KnjigaPrediction>();

var predictionEngine = mlContext.Model.CreatePredictionEngine<KnjigaData, KnjigaPrediction>(model);

foreach (var book in allBookData)

{

if (readedBookData.Any(rb => rb.KnjigaId == book.KnjigaId))

continue;

var bookVector = predictionEngine.Predict(book).Features;

float totalScore = 0;

foreach (var readedBook in readedBookData)

{

var readedVector = predictionEngine.Predict(readedBook).Features;

totalScore += CalculateCosineSimilarity(bookVector, readedVector);

}

predictions.Add(new KnjigaPrediction

{

Id = book.KnjigaId,

Score = totalScore / readedBookData.Count

});

}

var predictionIds = predictions.OrderByDescending(x => x.Score).Take(5).Select(x => x.Id).ToList();

return mapper.Map<List<Model.KnjigeDTOs.Knjige>>(await allBooksQuery.Where(x => predictionIds.Contains(x.KnjigaId)).ToListAsync());

}

1. **Putanja i printscreen iz pokrenute aplikacije gdje se prikazuju preporuke**

**Prijaviti se kao čitalac na mobilnu aplikaciju (citalac1:test). Na početnoj stranici se odmah nalazi top 5 knjiga za tok čitaoca.**

1. A screenshot of a phone

   Description automatically generated