Inledning

I detta projekt har jag byggt ett kollaborativt rekommendationssystem för filmer. Systemet använder tidigare betyg från användare för att förutsäga vilka filmer en användare troligen skulle gilla. Det går att tänka på det som ett slags ”Spotify för film” där systemet tittar på vad liknande personer har tyckt om, och sedan ger förslag.

Systemet används genom att man anger ett användar-ID. Modellen tar det ID:t, jämför det med datan från andra användare, och räknar ut vilka filmer som just den personen borde gilla bäst. Resultatet blir en lista med filmer som användaren inte har sett än, men som modellen tror att användaren skulle ge högt betyg.

Modellen

Jag använde TensorFlow (Keras) för att bygga modellen. Det var naturligt eftersom vi introducerats till det i modulen. Arkitekturen är ganska enkel men ändå kraftfull. Den består av följande delar:

- **StringLookup:** Användar-ID och film-ID omvandlas först till index så att modellen kan arbeta med dem.
- **Embedding-lager:** Embeddings ger varje användare och film en egen vektor i samma ”karta”. En användare hamnar nära de filmer som modellen tror att hen gillar, och längre bort från de filmer hen inte gillar.
- **Sammanfogning (Concatenate):** Användarens och filmens vektorer läggs ihop till en längre vektor som sedan används av modellen för att hitta mönster.
- **Dense-lager:** Sedan används några lager av noder med ReLU-aktivering för att hitta mer komplexa samband än bara linjära mönster.
- **Output (Dense(1)):** Slutligen finns det en nod som ger ett förutspått betyg för filmen. Värdet är kontinuerligt och ska helst hamna nära användarnas riktiga betyg.

Träningen gick till så att modellen fick in par av användare och film, med det faktiska betyget som mål. Förlustfunktionen var MSE (mean squared error) och jag mätte även RMSE (root mean squared error). Jag använde early stopping för att undvika överträning.

Motivering av metod

Jag valde kollaborativ filtrering eftersom:

- **Enklare fokus:** Uppgiften låg i kursdelen om TensorFlow och neurala nätverk. Kollaborativ filtrering med embeddings passade bra för att träna en modell med TensorFlow.
- **Mindre förarbete:** För att göra en innehållsbaserad modell hade jag behövt bearbeta textfält (t.ex. overview, tagline) och kategorisera data (t.ex. genre), vilket tar mycket tid och kräver mer feature engineering.
- **Stark metod för rekommendationer:** Kollaborativ filtrering hittar mönster i användarnas beteenden, vilket ofta fungerar bra i praktiken, även utan att använda filmens innehåll.
- **Utbyggbarhet:** Min modell kan i framtiden göras till en hybrid (kombination av kollaborativ + innehållsbaserad), men det var ett för stort steg för denna uppgift.

Jag valde att använda de mindre filerna (ratings_small.csv och links_small.csv) istället för hela datasetet. Detta på grund av att den fulla datan var väldigt stor och tog för lång tid att bearbeta och träna på min dator. För projektet räckte det lilla datasetet bra, eftersom syftet var att bygga en fungerande modell och förstå metoden, inte att optimera på hela datamängden.

Resultat

Efter träning fick modellen en validerings-RMSE på ungefär 0.9. Det betyder att den i genomsnitt ligger mindre än 1 betygssteg fel. Det är hyfsat bra med tanke på att betygsskalan går från 0,5 till 5.

När jag testade systemet såg jag att det ofta rekommenderade populära filmer som många användare gillar. Det var inte alltid jättestor variation mellan användare, men det fanns ändå skillnader. Jag testade även en modell som använde dot-produkten mellan användar- och filmvektorer. Det innebär att modellen beräknar ett slags likhetsmått. Ju högre dot-produkt, desto större chans att användaren gillar filmen. Den modellen fungerade okej, men jag tyckte att concat-modellen gav mer varierade rekommendationer och därför bättre för mitt projekt.

Potentiella problem

Trots att systemet fungerar finns flera svagheter:

- **Bias:** Systemet gynnar populära filmer. Om många användare har betygsatt en film högt är det stor chans att den dyker upp för nästan alla. Det gör att mindre kända filmer nästan aldrig rekommenderas.

- **Begränsad personalisering:** Vissa användare får rekommendationer som ser ganska lika ut. Det beror delvis på att datan är liten, men också på att modellen tränades på RMSE och inte på ranking. RMSE belönar att förutspå betyg rätt, men inte att göra listorna varierade.
- **RMSE vs verklig nytta:** Även om RMSE är låg betyder det inte alltid att rekommendationerna känns bra. En användare bryr sig mer om ifall listan är intressant än om det predicerade betyget är exakt rätt.
- **Skalbarhet:** Modellen fungerar bra på några tusen filmer, men skulle vara för långsam om man hade miljontals filmer och användare. Då hade man behövt optimera databehandlingen och kanske dela upp modellen i retrieval och ranking.
- **Cold start-problemet:** Ett annat problem med systemet är att det fungerar sämre för användare som har gett väldigt få betyg. Detta eftersom att modellen inte har tillräckligt med data om deras smak, då blir rekommendationerna mindre personliga. Detta kallas ofta för cold start-problemet.

Avslutning

Jag byggde ett fungerande rekommendationssystem baserat på kollaborativ filtrering i TensorFlow. Modellen använder embeddings för användare och filmer, concat och dense-lager för att kombinera dem, och en linjär nod som helt enkelt ger ett förutspått betyg som en siffra, utan att begränsas till ett visst intervall.

Systemet ger rimliga rekommendationer och RMSE är ganska lågt. Samtidigt såg jag att det ofta föreslog populära filmer och att variationen kunde varit bättre. Jag valde denna metod eftersom den var enkel att implementera, passade ett litet dataset och för att jag lärde mig grunderna i hur neurala rekommendationssystem fungerar.

Om jag skulle fortsätta utveckla systemet skulle jag vilja:

- Testa ett större dataset.
- Lägga till innehållsdata som exempelvis genre.
- Använda rankingmått istället för bara RMSE.

Projektet har gett mig en bra grundförståelse för hur man kan bygga rekommendationssystem och vilka utmaningar som finns.