Planering

Problemområde

Jag har valt att fokusera på FN:s 4:e mål, “God utbildning för alla” som fokuset i min examination. Min tanke är att skapa en hemsida som genom AI hjälper till att rekommendera böcker baserat på vilka kategorier en användare har valt.

Planerad lösning

Jag ämnar att använda google books som API för att kunna få tillgång till ett data-set att träna modellen. Därefter ämnar jag nyttja samma API för att kunna visa mer och tydligare info för användaren om den föreslagna boken.

Bidrag till Agenda 2030

Tanken med hur applikationen kan bidra till Agenda 2030 är att hjälpa till att öka intresset för läsning. Målet är att kunna bidra till mål 4, “God utbildning för alla”, genom att en användare som har svårt att veta vad den vill läsa kan välja vilka kategorier den har gillat sedan tidigare och då få förslag på nya liknande böcker. I förlängningen skulle applikationen potentiellt kunna visa var denna bok går att läsa gratis online eller om detta inte är möjligt hänvisa var den går att få tag på.

Etiska överväganden

Den första etiska svårigheten med denna applikation är att urvalet av den data som kommer att användas för tränande av modellen kommer att vara begränsad. Då jag ej har kapacitet att samla in ett mer komplett urval så kommer modellen inte att kunna ge några heltäckande förslag. För att mitigera detta kommer en så stor mängd som möjligt att samlas in.
Den andra etiska frågan som bör hållas i åtanke är de språkliga och kulturella problemen som en användare kan ha. Tas inte detta i åtanke blir modellen exkluderande och ej applicerbar på olika grupper. Detta är till stor del styrt av vilken träningsdata som används vid tränandet av modellen.

Implementation

Dokumentation av kod och process

Jag valde att använda NextJS som grund för applikationen då det ger en robust grund att stå på för vidare utveckling för hemsidan. Med NextJS behöver jag inte fokusera på att bygga API:er mellan frontend och backend vilket underlättar byggandet och simplifierar koden.

Jag valde att använda Material UI för att på ett enkelt sätt kunna bygga lite finare komponenter och gömma mycket komplexitet som annars skulle hamna i applikationen. Med hjälp av MUI kunde jag bygga ett finare och effektivare UI och då lägga mer fokus på AI-delen av projektet.

Jag valde att använda mig av google books API för att hämta data till träning av modellen. Google books api är öppet och ger mig enkelt tillgång till mycket data snabbt. Dessvärre finns det dock begränsningar som gör att jag endast kunde hämta 40 böcker åt gången. Jag valde därför att göra anrop fem gånger med olika sök-kategorier i ett försök att samla in mer användbar data för ändamålet.

Exempel på hur jag hämtar data:

```
async function getBooks() {
    const apiLink = "https://www.googleapis.com/books/v1/volumes"

    const categories = ["magi", "historia", "deckare", "romantik",
"fantasy"]
    const promiseList = categories.map(async (genre) => await
fetchData(`${apiLink}?q=${genre}&maxResults=40`))
    const data = await Promise.all(promiseList)

    const books = data.flatMap(books => books.items)
    return books
}
```

Jag valde att använda TensorFlow för att träna min modell då det är ett kraftfullt bibliotek för att bygga och träna modeller för bland annat prediktion. Genom att använda TensorFlow, kan man integrera maskininlärning i webbapplikationer utan att behöva förlita sig på externa servrar, vilket ger snabbare svarstider och möjliggör realtidsinteraktioner. TensorFlow gör det också möjligt att köra modeller på både klient- och serversidan, vilket ger flexibilitet och skalbarhet vilket skulle kunna vara användbart i framtiden om man skulle vilja vidareutveckla applikationen.

Testning och analys

Jag hade dessvärre inte tid att implementera någon testing i kod vilket skulle öka konfedensen i lösningen. Det som går att se är att varje gång man laddar om sidan så tränas en ny modell med ny data i och med hur google books api tycks fungera. Detta kopplat med användartester som jag gjort under utvecklingsprocessen har gett mig ganska hög konfidens i lösningen då den tycks ge rimliga resultat.

Med det sagt så skulle det första steget för vidare utveckling och ökad konfidens i produkten vara att implementera tester i kod. För att sedan fortsätta öka kvaliteten av produkten bör en ny och effektivare källa av träningsdata säkerställas. Efter det har man som utvecklare en stark grund att börja laborera med olika sätt att modifiera träningen av modellen.

Reflektion

Hållbarhet och etik

Lösningen bidrar till FN:s hållbarhetsmål genom att försöka öka läsintresset och på så sätt bidra till FN:s 4:e mål, “God utbildning för alla”. Tanken är att en användare som eventuellt har svårt att hitta böcker som denne skulle vara intresserad av kan välja vilka genrer som denne gillar och kan då bli rekommenderad ett antal liknande böcker.

Det finns ett antal etiska frågor som bör tänkas på med denna lösning. Den första är att kvaliteten på datan är ganska låg. Jag har försökt motverka detta med att dels samla ihop mer data och sedan städa den till viss del genom att exempelvis plocka bort böcker utan kategorier/genrar definierade. Det går dock inte att komma ifrån att med en högre kvalitet på input-datan så skulle lösningen kunna bli bättre och med mindre bias.

Den andra etiska frågan som bör beaktas är att det endast är svenska böcker som finns i biblioteket jag har använt. För att få denna lösning mer tillämpbar och rättvis bör litteratur från många fler länder och kulturer användas. Detta har varit svårt för mig att motverka i utvecklingen av applikationen då jag hade svårt att hitta bra databaser som är öppna mot internet för konsumtion.

Den tredje etiska frågan som bör tänkas på är hur kontroversiella böcker bör behandlas. I dagens politiska klimat är det mer relevant än någonsin att tänka på att böcker och rekommendationer av dem har betydelse. Oavsett om man väljer att exkludera vissa böcker eller inte så är det ett ställningstagande vilket är viktigt att tänka på.

Utvecklingsprocessen

Under utvecklingen gick det väldigt smidigt att använda Material UI för mina frontend-komponenter. På så sätt kunde jag spara mycket tid och ändå få ett okej UI.
Google books API underlättade mycket då jag inte behövde skapa en egen samling av böcker för träningsdata. Det var ganska lätt att konsumera och hade ett ganska högt maxtak på hur många böcker som kunde sökas samtidigt.
Även användandet av TensorFlow gick smidigt och jag kunde relativt snabbt få upp en fungerande lösning för modellen. Det finns en ganska bra dokumentation och sedan mycket information online för att hjälpa när man stötte på problem.

Jag stötte på tre stora problem under utvecklingen. Det första som skedde var att den senaste versionen av NextJS hade en bugg i sig som gjorde att jag hade stora problem med att få upp miljön. När jag till slut nedgraderade till den näst senaste versionen så hade jag fortfarande problem att få upp miljön tills jag upptäckte att det hade att göra med VS-codes inbyggda terminal. Försöker man köra denna applikation rekommenderar jag att man gör det med en extern terminal och inte VS-codes integrerade.
Den andra svårigheten jag fastnade ett tag med var att jag försökte att inte träna om modellen varje gång man laddar om sidan. Det bör teoretiskt gå men jag kunde dessvärre ej få det att fungera. Jag valde då att träna en ny modell varje gång man laddar om sidan. För att det ska gå behövde jag dock kalla på funktionen `tf.disposeVariables();` innan en ny modell ska tränas. Annars krockar variablerna i TensorFlow och användaren får ett felmeddelande.

Den sista buggen jag upptäckte som jag dessvärre inte haft tid att lösa är att när man klickar på knappen “ Recommen books” så hoppar listan med kategorier/genres ner en bit. Det är inte så fint men det påverkar inte funktionaliteten så jag valde att fokusera på annat i brist på tid.

Framtida förbättringar

För framtida utveckling av applikationen har jag identifierat ett antal förbättringsområden.
Det första som skulle göras är att med hjälp av Tensorflows inbyggda valideringsfunktioner undersöka modellens förlust (loss) och noggrannhet (accuracy). Detta skulle ge utvecklaren verktyg att kunna testa sig fram till en starkare och bättre modell.
Det andra som bör göras är att säkerställa en bättre och mer diversifierad källa för träningsdata. Detta är generellt det absolut svåraste i byggande av modeller. Det finns api:er som man kan betala för. Vill man bygga en robustare lösning så kan det vara en värd investering.
Det tredje man skulle kunna kolla på är att implementera fler språk för att göra lösningen mer universell. Detta skulle dock också krävas en större källa till träningsdata.
Det fjärde som skulle kunna implementeras är att om möjligt kunna hänvisa användaren till var denna kan få tag på böckerna. Detta skulle öka nyttan mycket för att bidra till FN:s 4:e mål
Det sista som jag tror skulle vara intressant att kolla på vore om man vid träningen inte bara kollade på böcker utan kanske utvecklade en “Bag of words”-lösning för sammanfattningen på böckerna och tränade modellen på det. Det skulle öka komplexiteten men även nyttan något enormt.