Henrik Holtvedt Andersen

Learning to Rank i folkebibliotek

Masteroppgave 2023

Master i Bibliotek og informasjonsvitenskap

OsloMet – Storbyuniversitetet, Institutt for Arkiv- bibliotek- og informasjonsfag

Sammendrag

Denne oppgaven har som mål å skape rangeringsmodeller via metoden Learning to Rank, og den vil undersøke hvordan de kan bidra til å forbedre rangeringen av dokumenter i trefflista i folkebibliotekkatalogen. Learning to Rank bruker maskinlæring til å trene opp en rangeringsmodell. Denne trenger merkelapper (training labels) som den trenes mot. Disse har vi skaffet via menneskeskapte relevansvurderinger av dokument-søketerm-par fra Deichmankatalogen - Norges største folkebibliotek. Sammen med data fra Deichmankatalogen, som er blitt representert på ulike måter via features, har vi trent opp noen modeller med algoritmene LambdaMART og Random forest. Vi evaluerte modellene ved å måle plasseringen av utvalgte dokumenters posisjon på trefflisten med og uten modellene. Resultatene viste at de fleste modellene klarte å forbedre rangeringen av trefflista. Modellen som presterte best var basert på en kombinasjon av popularitetsfaetures og tekst-matching-features. Resultatene viste at Learning to Rank er en lovende metode for å forbedre trefflista i folkebibliotek.

Summary

The goal of this thesis is to create ranking modeles based on Learning to Rank and determine if and how they can be used to improve the ranking of materials in a public library catalogue. Learning to Rank is an approach based on machine learning that need labeled data. We used training labels based on human relevance judgements of query-document pairs of materials in the Deichman collection - the biggest public library in Norway. Together with data from its collection, represented as features, we trained several ranking models using the algorithms LambdaMART and Random forest. We evaluated the models by measuring the ranking positions of certain documents with and without using the models. The results showed that most of the models succeeded in improving the ranking. The model with the best results proved to be a model based on a combination of popularity features and text matching. The results indicate that Learning to Rank is a method with much potential to improve the ranking of materials in a public library catalogue.

Problemstilling:

Hvorvidt kan maskinlæringsmetoden Learning to Rank (LTR) brukes til å forbedre rangeringen av dokumenter i trefflista i folkebibliotek?

Forskningsspørsmål:

Hvilke typer data, omsatt til features, kan bidra til å forbedre rangeringen av dokumenter i trefflista i folkebibliotek, og hvordan bør disse kombineres?

Takk til veileder Michael Preminger for god veiledning og sitt bidrag med koding.
Takk til Deichman for delingen av data som har blitt brukt i oppgaven,
og takk til alle assessorene (relevansvuderere) som har bidratt med å skape merkelapper til treningen av rangeringsmodellene.

Innholdsfortegnelse

1 Innledning

Algoritmer, maskinlæring og kunstig intelligens (KI/AI) er ord som blir stadig mer kjent. Det snakkes gjerne om google-algoritmen eller algoritmene som produserer feeden på sosiale medier. Disse er basert på maskinlæring og kan for eksempel favorisere innhold om hobbyene dine uten at du spesifikt har bedt om det, basert på dine brukerdata (Google, 2023). Maskinlæring og kunstig intelligens har i skrivende stund, våren 2023, fått en eksplosiv utvikling, hvor søkemotoren ChatGPT har blitt verdenskjent. Dette er et stadig mer intelligent digitalt verktøy og en søketjeneste for å skape og tilegne seg informasjon (https://openai.com/). Man kan diskutere fordeler og ulemper med denne utviklingen, men det er mye som tyder på at produkter av maskinlæring og kunstig intelligens har kommet for å bli, og vil fortsette å utvikle seg. Noen akademiske databaser og fagbibliotek bruker allerede søkealgoritmer og andre tjenester produsert via maskinlæring. Et eksempel på en akademisk database som bruker en maskinlært søkealgoritme er Semantic Scholar (https://www.semanticscholar.org/). ScienceDirect bruker maskinlæring som et verktøy til å generere en samling av forskningsemner som man kan navigere i (*ScienceDirect Topics*, 2023). Det finnes mye forskning på maskinlæring innen bibliometri og fagbibliotek-kontekst. For eksempel en artikkel som undersøker hvordan maskinlæring kan brukes til å forutse den fremtidige siteringsfrekvensen av nye forskningsartikler (Abrishami & Aliakbary, 2019). Søkealgoritmer og andre tjenester i norske folkebibliotek er ikke basert på maskinlæring enda, og det er lite forskning å finne om emnet. Denne oppgaven har som mål å bidra til slik forskning. Søkealgoritmene i folkebibliotek-katalogene i dag baserer seg på klassiske modeller for søk og gjenfinning. Denne oppgaven vil undersøke om en søkealgoritme produsert med maskinlæringsteknikken Learning to Rank vil føre til en forbedret rangering av trefflisten i folkebibliotek. Mer konkret utføres et LTR-eksperiment med et mål om å løfte rangeringen av noen populære dokumenter som er *for langt nede* på trefflisten i proporsjon med deres popularitet basert på klikkdata. I evalueringen måler vi om vi har klart å flytte disse dokumentene lenger opp på trefflista. Tanken er at hvis eksperimentet lykkes med det, vil vi kunne si at LTR er en lovende metode for å forbedre trefflisten i folkebibliotek, noe som vil kunne føre til økt utlån og mer fornøyde brukere av katalogen.

1.1 Deichman-data

Deichman bibliotek er Oslos folkebibliotek og består av 22 bibliotek, hvor Deichman Bjørvika er det største og mest kjente. Deichman har sendt oss en stor fil med titler som er ganske langt nede på trefflista, men som likevel genererer en del klikk fra brukerne, noe som indikerer at de er mer relevante for søket enn hva deres plassering skulle tilsi; en liste over dokumenter som er for dårlig rangert. Denne filen inneholder tittel, tilhørende søketerm, plassering på trefflista og antall klikk. Via *Elasticsearch,* gjenfinningsmodulen til Deichman, har vi tilgang på metadata fra alle dokumentene i Deichmandatabasen fra tidspunktet mars 2022. Derfra kan vi hente inn de dokumentene fra deres database som vi trenger å få vurdert av de menneskelige assessorene, og vi kan lage features basert på metadataene. Vi har også tilgang på utlånstall fra Deichman. Vi har ikke brukt - eller hatt tilgang på - mer avanserte brukerdata som for eksempel klikk og *dwell time* (hvor mye tid man har brukt inne på et dokument) eller annen mer sensitiv informasjon om brukerne til Deichman.

I denne oppgaven kalles alle søkbare enheter i Deichmankatalogen for *dokumenter*. Noen ganger kan de være omtalt som *kilder*. Dokumentene i Deichmankatalogen er varierte. De finnes i 39 ulike formater, fra *musikkinstrumenter* til *DVD. Bøker* utgjør størstedelen samlingen. De har dokumenter på 143 språk, og de er fordelt over 9 kategorier av målgrupper, for eksempel *0-2 år* og *voksne*. Samlingen er delt mellom fag og fiksjon og dekker 116 sjangre. Samlingen inneholder over 1,1 millioner bøker, tidsskrifter, musikk og filmer (*Om oss - Deichman.no*, 2023.).

1.2 Søk og gjenfinning i folkebibliotek

Søkesystemet til Deichman og andre norske folkebibliotek er basert på klassiske modeller for gjenfinning som ikke er produkter av maskinlæring. Rangeringsfunksjonen rangerer ikke basert på mønstergjenfinning i treningsdata som inneholder brukerdata og menneskeskapte relevansvurderinger; bibliotekbrukerne påvirker ikke rangeringen. Deichman rangerer basert på algoritmen BM25. De har mulighet til å justere parametere og tilpasse algoritmen og trefflisten etter behov.

Oppgaven tar utgangspunkt i en antakelse om at rangeringen av trefflisten kan forbedres, altså at dagens treffliste ikke er optimal. Rent intuitivt kan vi for eksempel se på trefflisten for søkeordet “harry potter” og vurdere dens relevans. I skrivende stund, mai 2023, finner vi bok nummer 1 i Harry Potter-serien, *De vises sten*, på 19. plass på trefflisten. Dokumentet øverst på trefflisten finner vi boken *Harry Potter - en filosofisk trollmann*. Dette er et sekundærverk utgitt i 2006 som filosoferer om Harry Potter-universet, ved å trekke inn klassiske filosofer og lignende, og målgruppen er voksne. Det er sikkert en interessant bok for noen, men det er trolig ikke det verket folk flest tenker på når de søker *harry potter* i Deichmankatalogen. Begge bøkene kan sies å være relevante for søket, og det er fint at de finnes i trefflisten, men denne oppgaven har som ambisjon å optimalisere rekkefølgen av dokumentene hvor de mest relevante kommer først. Dette er et konsept som innen informasjonsgjenfinning kalles *best match*, og som skiller seg fra det boolske konseptet om at et dokument kun er relevant eller ikke relevant, og om et dokument blir hentet eller ikke hentet til trefflisten.

Deichman har brukere og lånere i alle aldre. Deres ulike informasjonsbehov i forhold til søk, kan reduseres til noen få kategorier. En av dem er *known-item search* (Wildemuth & O’Neill, 1995). Her vet brukerne på forhånd hvilke dokumenter de skal ha; de ønsker seg for eksempel den siste krimboken til Jo Nesbø, og de søker etter dette i katalogen for å finne ut om den er ledig og hvor den står på hyllen. En annen kategori er *exploratory search*, hvor brukerne ikke vet helt hvilke dokumenter de er ute etter. Det kan hende de er interesserte i et fagområde, eller at de ønsker å bli overrasket med en god bok fra en ny forfatter. Marchionini (2006) kaller exploratory search for en av to hovedkategorier innen søk. *Lookup* er den andre, og *known-item*

search går inn under den. Særlig under *lookup* eller *known-item-search* er det viktig med en treffliste som er presis, hvor den konkrete tittelen er høyt oppe på trefflista. I disse situasjonene har nok brukeren ikke like stor tålmodighet og vilje til å scrolle og bruke tid på trefflista: da er det viktig med en rangeringsfunksjon som leverer god *presicion,* altså at en høy andel av trefflista er relevant. Med henhold til explorative search vil derimot ofte brukerne vise større evne til å scrolle og bruke tid på trefflista. Brukeren ønsker gjerne å vurdere et større antall relevante dokumenter, før personen treffer en beslutning om hvilke dokumenter som skal lånes. I dette tilfellet er det viktig at rangeringsfunksjonen leverer høy *recall*. Det vil si at trefflista ikke utelater noen relevante treff, slik at brukeren får en god og fullstendig oversikt over hvilke bøker som finnes innen fagområdet *astrologi*. Derfor kan man si at rangeringsfunksjonen må ta høyde for begge disse informasjonsbehovene.

Ifølge nasjonal bibliotekstrategi 2020-2023, under kapittelet formidling, står det at det er et mål å styrke bibliotekets og de bibliotekansattes arbeid med å formidle det store mangfoldet i bibliotekenes samlinger (Kulturdepartementet, 2019). Et spørsmål er da om den nye søkealgoritmen skal spille en rolle i dette og favorisere smal litteratur. Eller skal den være mer *nøytral*, og overlate formidlingen av smal litteratur til andre funksjoner og formidlingspraksiser? Trefflisten hos Deichman inneholder dokumenter som ansatte har merket med *anbefales* eller *vi liker*. Det er ikke godt å si om disse dokumentene faktisk favoriseres i trefflisten, men i teorien kan disse favoriseres og være et eksempel på en tilleggsfunksjon som kan bidra til å formidle smal litteratur. Det er ikke oppgavens formål å skape en søkealgoritme som formidler smal litteratur, og vi vil ikke evaluere modellen ut ifra det, men det er en interessant tanke som er verdt å nevne. Vårt hovedmål er å produsere en ny *baseline* modell, altså en mer generell rangeringsmodell som fungerer best mulig på hele samlingen, hvor *gjenfinning* er mer sentralt enn *formidling* av en viss type litteratur.

2 Teori

2.1 Informasjonsgjenfinning

Informasjonsgjenfinning er et fagområde som grunnleggende sett handler om å gi brukere rask og enkel tilgang den informasjonen de er ute etter. Informasjonsgjenfinning kan studeres fra et maskin- og systemteknisk perspektiv, eller fra et menneskelig perspektiv som også studerer menneskers informasjonsbehov og samhandling med informasjonssystemer (Baeza-Yates & Ribeiro-Neto, 2011, s. 1). Denne oppgaven vil først og fremst handle om informasjonsgjenfinning fra et systemteknisk perspektiv, men vi er også innom det menneskelige perspektivet i produksjonen av merkelappene (training lables), som vi kommer tilbake til. Generelt handler det om hvordan informasjonsobjekter er representert, indeksert, lagret og organisert for best mulig gjenfinning. I denne oppgaven er informasjonsobjektene alle søkbare kilder i Deichmankatalogen, hvor flesteparten er bøker. I oppgaven vil kildene ofte omtales som dokumenter. Et søk vil noen ganger omtales som en spørring eller en søketerm, eller det engelske ordet query. Modellering av en rangeringsfunksjon innen klassisk informasjonsgjenfinning kan reduseres til to arbeidsoppgaver. Den første er skapelsen av et regelbasert rammeverk over hvordan dokumenter og søketermer blir representert. Den andre er skapelsen av en rangeringsfunksjon som bestemmer rangeringen av hvert enkelt dokument ut fra en gitt søketerm (Baeza-Yates & Ribeiro-Neto, 2011, s. 57). De vanligste typene informasjonsobjekter som modeller baserer seg på er tekst, lenker og multimediale objekter (bilde, video og lyd). I denne oppgaven er det tekstbaserte modeller som er mest relevante. Tekstbaserte modeller kan brukes på informasjonsobjekter som bøker, bilder, video og lyd. De kan ikke alltid brukes direkte, men på tekstlige metadata knyttet til disse kildene. Disse kan igjen deles inn i noen klassiske hovedkategorier når søketermene er ustrukturerte: boolske modeller, vektor-modeller og probabilistiske modeller.

2.1.1 Den boolske modellen

Den boolske modellen er en enkel modell som ble brukt i tidlige biblioteksystemer, og boolsk logikk er fortsatt et relevant konsept innen søk og gjenfinning, men den boolske modellens evne til å produsere de beste trefflistene har i dag blitt forbigått av andre modeller og konseptet om termvekting. Den boolske logikken undersøker om det er samsvar mellom søketermene og de indekserte termene i dokumentene på en binær måte. Enten er søketermene der (1), eller så er de der ikke (0). Det spiller ingen rolle hvor mange treff søketermen har i dokumentene. Termene er ikke vektet etter hvor ofte de forekommer i dokumentet. Derfor predikerer den boolske modellen at et dokument er enten *relevant* eller *ikke relevant*. Det finnes ikke delvis relevante dokumenter. Det gir modellen lite rangeringsinformasjon, noe som kan føre til enten en veldig lang eller kort treffliste (Baeza-Yates & Ribeiro-Neto, 2011, s. 64-66), og trefflisten vil ikke være rangert etter hvilke dokumenter som er mest relevante. De lange trefflistene kan inneholde mange svakt relevante treff som lager mye *støy*, særlig ved korte søk. På den andre siden kan den strenge boolske logikken, hvor alle termer må være ganske nøyaktig representert i dokumentet, utelate mange relevante treff og gi kortere trefflister, særlig ved lengre spørringer. Vektor og probabilitetsmodellen vekter termene slik at det skapes mer rangeringsinformasjon. Derfor er disse mer populære enn boolske modeller (Baeza-Yates & Ribeiro-Neto, 2011).

2.1.2 Vektormodellen

Med vektormodellen, som også kan kalles vektorrom-modellen (Salton et al., 1975), kan hvert eneste ord i et dokument være sin egen vektordimensjon. Man kan måle likheten mellom denne og dimensjonen til spørringen eller andre dokumenter. Det er imidlertid mye å tjene på å vekte termene og redusere representasjonen av et dokument med utgangspunkt i et vektingsmål som TF-IDF. Da måles likheten mellom vektorene via cosinus-likhet mellom dokumentene og spørringen (Baeza-Yates & Ribeiro-Neto, 2011, s. 77-78). Dette gjør at alle dokumenter får verdier mellom 0 og 1, hvor 0 er dårligste match og 1 er beste match, og trefflisten rangeres basert på disse verdiene. Dette medfører at trefflisten inkluderer dokumenter som matcher delvis, i motsetning til den boolske modellens binære inndeling av relevante og ikke-relevante dokumenter. I vektormodellen er hver term er uavhengig av nabotermene;

det betyr at rekkefølgen av termene ikke tas med i betraktning. Denne egenskapen kalles *bag of words* (Zhao & Mao, 2018). Hvis du da søker *spill bibliotek*, så vil ikke dokumenter med frasen "*spill i bibliotek*" automatisk favoriseres over dokumenter hvor ordene opptrer langt unna hverandre og ikke danner den intenderte samlede betydningen fra søketermene. Det er imidlertid mulig å måle nærhet mellom søketermer via tilleggsteknikker som vi skal komme tilbake til senere i avsnittet om tekst-matching-features.

2.1.3 Probabilitetsmodeller

Det probabilistiske rammeverket baserer seg på ideen om at det finnes et sett av dokumenter som er relevante i forhold til en søketerm, og resten er ikke relevante. Dette er den ideelle gruppen av dokumenter. Utfordringen er å vite hvilke egenskaper denne gruppen har slik at vi kan bruke egenskapene som søketermer. Dette kan vi ikke vite på forhånd og derfor må vi gjette, noe som skaper en foreløpig beskrivelse av de relevante dokumentene, som gir en treffliste. Deretter bestemmer brukeren hvilke dokumenter som er relevante og hvilke som ikke er det, gjennom sin interaksjon med trefflisten. Systemet benytter informasjonen i denne interaksjonen til å finstemme beskrivelsen av de ideelle dokumentene for denne spørringen. Ved å gjenta denne prosessen mange ganger, er det ventet at beskrivelsen/representasjonen av de relevante treffene vil bli mer presise, noe som fører til bedre gjenfinning (Baeza-Yates & Ribeiro-Neto, 2011, s.80). Det finnes ulike måter å regne ut probabiliteten av relevans på, og noen metoder er ikke avhengig av interaksjon fra brukerne. Resultatet er uansett at dokumentene vil bli rangert i synkende rekkefølge etter *sannsynligheten* (probabiliteten) av at de er relevante.

Den boolske modellen er ansett for å være den dårligste av de klassiske modellene, på grunn av at den ikke tillater delvis matching. Det er delte meninger om hvilken modell som er best av de to andre. Vektormodellen fungerer godt på generelle samlinger og er derfor fremdeles en populær gjenfinningsmodel, og som stadig brukes som baseline modell for å evaluere nyere gjenfinningsmodeller (Baeza-Yates & Ribeiro-Neto, 2011, s.79). Probabilitetsmodellen BM25 øker imidlertid i anseelse, og noen mener den presterer bedre enn vektormodeller når parameterne er finjusterte (Baeza-Yates & Ribeiro-Neto, 2011, s.107).

2.1.4 TF-IDF

En vektormodell kan ta i bruk vektingsmålet TF-IDF: *Term frequency - inverse document frequency.* Den matematiske formelen kan ses fra figur 2.1. Dette vektingsmålet teller hvor mange ganger en søketerm finnes i et dokument og deler det på antall termer i dokumentet (TF). Videre regner den ut spesifisiteten av termene hvor termer som opptrer sjeldnere får mer vekt. Dette gjøres ved å dele den totale mengden dokumenter på mengden dokumenter som inneholder søketermen (IDF). Denne ganges med TF. Med TF-IDF skjer det en lengdenormalisering som gjør at korte og lange dokumenter kan sammenlignes, ellers ville lange dokumenter hatt en fordel, fordi flere termer gir flere muligheter for treff. TF-IDF tar ikke rekkefølgen av termer i betraktning og har dermed egenskapen av *bag of words.*

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Figur 2.1. Formelen for TF-IDF.

2.1.5 BM25

BM(*best matching*)25 er basert på den probabilistiske modellen. Et problem med termvektingen til den originale probabilistiske modellen var at den ikke lengdenormaliserte eller tok termfrekvens i betraktning. Den regnet kun IDF, som forklart i avsnittet om TF-IDF. BM25 er en utvikling av den probabilistiske modellen hvor lengdenormalisering og termfrekvens tas i betraktning i termvektingen, noe som forbedrer modellen betraktelig. BM25 er en mer komplisert modell enn TF-IDF. Fra formelen i figur 2.2 kan man se at den har noen parametere (k og b) som kan justeres. De kan optimaliseres på avanserte måter, men vanligvis er 1 en god verdi

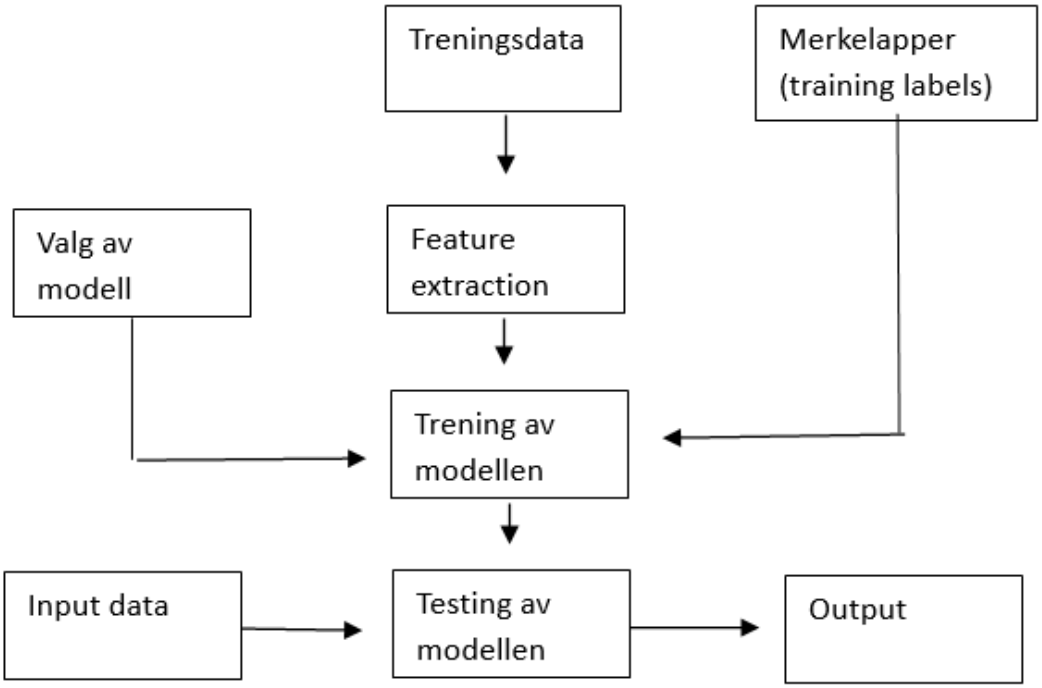
for *k1*, mens 0,75 er en god verdi for *b* (Baeza-Yates & Ribeiro-Neto, 2011, s.107).

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Figur 2.2. Formelen for BM25

2.2 Maskinlæring

Maskinlæring er et stort felt innen kunstig intelligens som produserer modeller som kjenner igjen mønstre i data og kan klassifisere eller predikere relevans av nye data basert på det (Baeza-Yates & Ribeiro-Neto, 2011, s.282). Opplæringen av maskinlæringsalgoritmer er i hovedsak enten veiledet, ikke-veiledet eller semi-veiledet. Veiledet læring går ut på at den lærer en klassifiseringsfunksjon basert på treningsdata som input. Målet er at klassifiseringsfunksjonen skal klassifisere ny informasjon med høy presisjon. Ikke-veiledet læring bruker ikke treningsdata med merkelapper (training labels) for hva som klassifiseres. Her er jobben til klassifikatoren å dele dokumentene inn i grupper basert på egenskapene til kilden, uten at mennesker har navngitt merkelapper eller gitt indikasjoner på hva som er en god gruppering. Dette kalles clustering. Semi-veiledet læring er en blanding av disse to, hvor man kombinerer en liten porsjon av data med merkelapper og en større porsjon data uten merkelapper. (Baeza–Yates & Ribeiro-Neto, 2011, s.283).
I denne oppgaven bruker vi veiledet læring, hvor vi skaffer merkelapper via menneskeskapte relevansvurderinger. Vi bruker treningsdata som trenes mot disse merkelappene med en algoritme, det skaper en modell som kan rangere nye dokumenter.
I figur 2.3 ser man en oversikt over fremgangsmåten ved maskinlæring. Oppgaven vil følge denne fremgangsmåten. Treningsdata kan være data fra Deichman, som bokomtaler eller utlånstall. Av dataene kan man lage feature-representasjoner som er bedre egnet for maskinlæring. Ved å ha en god forståelse av problemet man skal løse, og ved å studere treningsdataene og features, velger man en passende modell til treningen. Gjennom testing og evaluering kan man eksperimentere med ulike modeller og features for å se hvilke som gir best resultat. Til slutt er målet at den nye modellen skal gi best mulig *output,* som i vårt tilfelle betyr best mulig rangering.

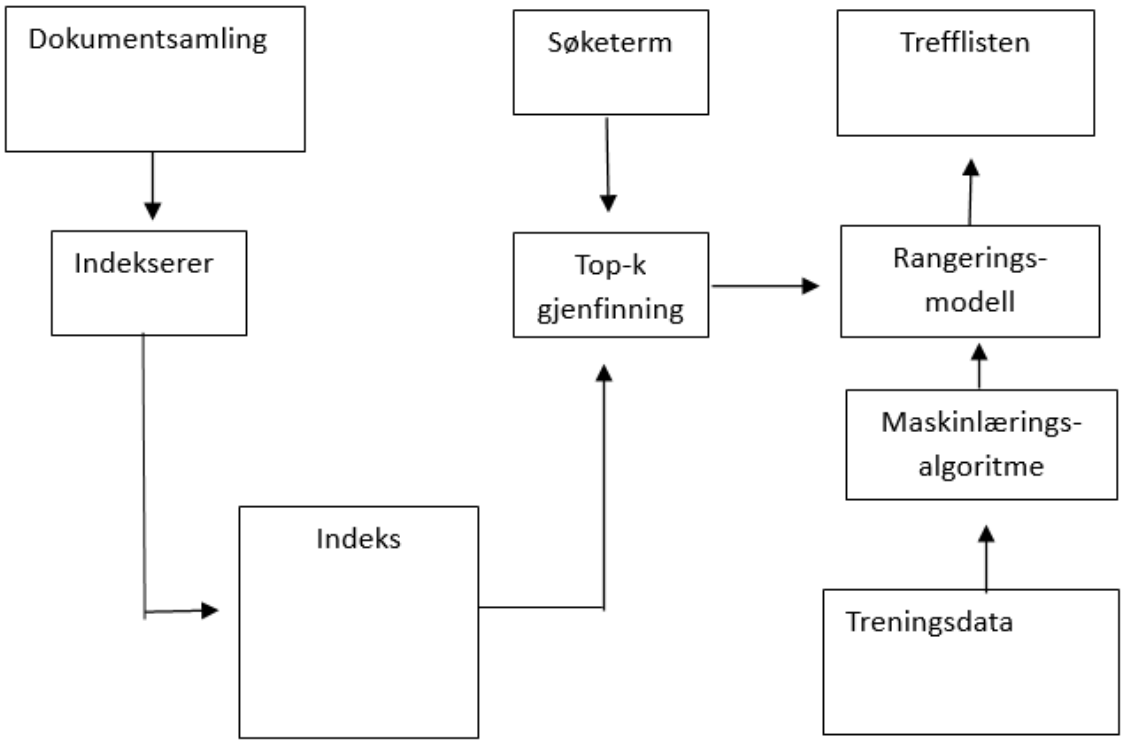


Figur 2.3. En oversikt over maskinlæringsprosessen

2.2.1 Learning to Rank (LTR)

Learning to Rank er en metode som bruker maskinlæring-algoritmer til å lære å rangere en treffliste av dokumenter. Det skaper en modell som blir til gjennom treningsdata som inneholder rangeringinformasjon og som presenteres for en maskinlært algoritme (Baeza-Yates & Ribeiro-Neto, 2011, s. 473).
Derfor bruker LTR veiledet læring. LTR er en type ML som alltid dreier seg om rangering, mens maskinlæring kan løse andre typer problemer også. I rangeringen er det ikke viktig å anslå en verdi for hvert enkelt dokument; det viktigste er at rangeringen av alle dokumentene er god (Li, 2011, s. 1856). LTR innebærer bruk av *loss function.* Hvis feature vektorene har høy verdi, altså godt samsvar mellom dokument og spørring, og samtidig er rangert høyt, er *loss function* liten (Li, 2011, s. 1857). *Loss function* kan beregnes på ulike måter. Tre vanlige metoder er *point-wise*, *pairwise* og *listwise*. Innen *point-wise* rangering blir oppgaven et regresjonsproblem hvor man trener en modell til å forutse scoren til et dokument. Med *pairwise* rangering blir oppgaven et binært klassifikasjonsproblem hvor modellen bestemmer at det ene dokumentet er mer relevant enn det andre. Med *listwise* rangerer man hele listen av dokumenter. Metoden regner ut *loss function* ved hjelp av evalueringsmål som f.eks. NDCG. Utregningen tar utgangspunkt i konseptet om at:

loss = 1 - NDCG (Li, 2011).



Figur 2.4. Oversikt over Learning to Rank

I figur 2.4 ser man en oversikt over hvordan maskinlæring kan brukes til et rangeringsproblem. I denne oppgaven vil dokumentsamlingen være Deichmankatalogen. Videre i figuren foregår en indeksering av samlingen. Kun metadata er søkbare i folkebibliotek. Brukerens spørring blir behandlet av søkesystemets modeller og gir en treffliste. Det som skjer på venstresiden i figuren er elementært for alle søkemotorer i norske folkebibliotek. Det nye med maskinlæringsrangering skjer på høyresiden. Her bruker vi treningsdata som kan være data basert på Deichmankatalogen, brukerens interaksjon med katalogen, eller relevante data fra andre kilder. Disse treningsdataene blir behandlet av en maskinlæringsalgoritme. Denne algoritmen får kun top-k resultatene fra spørringen, det vil si for eksempel de 10 eller 20 øverste treffene. Dette gjør at algoritmen kun rerangerer de øverste treffene. Dette er anbefalt fordi det vil være en krevende prosesseringsjobb for systemet å rangere hele samlingen. Det er raskere å

grovsortere via den ordinære modellen, basert på f.eks. TF-IDF, og foreta en LTR-rangering av de beste treffene. Deretter bruker man algoritmer på treningsdataene samme måte som forklart under avsnittet om maskinlæring.

2.2.2 Merkelapper (training labels)

I figur 2.3 har vi en boks som heter merkelapper. Dette gir dataene verdi, for eksempel ved at noe blir klassifisert som et eple istedenfor en banan, eller at noe blir klassifisert som *relevant* eller *ikke relevant*. Disse merkelappene blir målverdien (target value) som algoritmen trenes mot; algoritmen finner en ukjent funksjon fra disse eksemplene. Derfor er det viktig at merkelappene er gode. Ved å involvere mennesker som vurderere, særlig ekspertvurderere med domenekunnskap om det de vurderer, legger man til rette for en produksjon av merkelapper av god kvalitet. Vi kaller de som vurderer for assessorer. Det er også en fordel at flere vurderer de samme dataene slik at merkelappene oppnår god konsensus. Det er tid og ressurskrevende å skaffe menneskeskapte relevansvurderinger, særlig via ekspertvurderere. For å bøte med dette kan man skaffe merkelapper via crowdsourcing, hvor eksterne personer vurderer, enten betalt eller frivillig. Ulempen med crowdsourcing er at man ikke kan forsikre seg om påliteligheten til assessorene i samme grad som om det ville blitt gjort internt. Det kan prege kvaliteten av merkelappene. Det finnes imidlertid metoder for å bedre kvaliteten av disse vurderingene (Samimi & Ravana, 2014). Dette kommer vi tilbake til i kapittelet om relevansvurderinger. En enda mer tids og ressurseffektiv måte å produsere merkelapper på er via klikk-data. Da er tanken at kilder med høyt antall klikk får merkelappen *relevant*, mens kilder med et lavt antall klikk får merkelappen *ikke relevant*. Varianter av dette er å bruke utlånstall eller kjøpstall som relevansindikatorer. Kjell Arne Hellum (2017) gir et eksempel på dette, hvor han bruker kjøpstall og klikk som relevansindikatorer i sitt LTR-eksperiment innen netthandel. Dette er en indirekte måte å produsere merkelapper på. Den klikkbaserte metoden er imidlertid sårbar mot spambot-klikking, som kan gi gale relevansindikasjoner. En treffliste skaper i tillegg noe som kalles trust bias. Det betyr at man gjerne klikker på de øverste treffene fordi man ofte stoler på søkemotorens evne til å plassere de mest relevante treffene øverst, og dette kan gi misvisende relevansvurderinger (Joachims et al., 2007). Klikk-baserte vurderinger skaper mye støy, noe som gjør det utfordrende å skape gode relevansvurderinger. Likevel finnes

det metoder som gjør klikk-baserte vurderinger til en mulig fremgangsmåte i å skaffe merkelapper på. Joachims (2007) foreslår noen teknikker for å bøte på dette biaset hvor man regner en funksjon mellom de dokumentene som ble klikket på og de om ikke ble klikket på. Oppgaven vil ikke gå dypere inn i hvordan man kan produsere gode systemgenererte merkelapper, siden det er *menneskelige* relevansvurderinger som produseres i denne oppgaven.

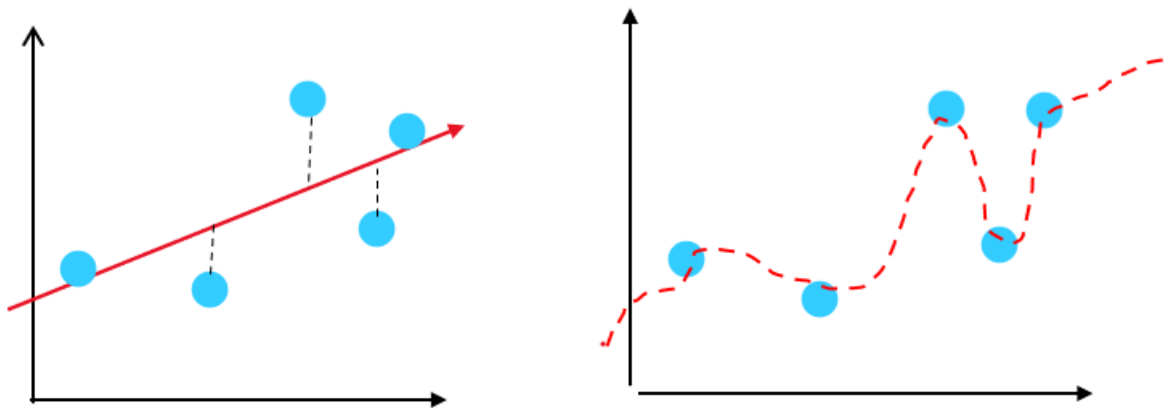
2.2.3 Modeller og algoritmer

Ordene modell og algoritme ligner hverandre i innhold i maskinlæringssammenheng. Som nevnt i innledningen har ordet algoritme blitt populært, hvor det er mer vanlig å si Google-søkealgoritmen enn Google-søkemodellen. Strengt tatt er det modell som er det riktige ordet å bruke om det overordnede sluttproduktet som rangerer trefflisten i Google og andre steder. En modell er sluttproduktet av å anvende en eller flere algoritmer på dataene. En algoritme er en trinnvis oppskrift på hvordan man skal løse et problem matematisk, og den trenger ikke å være et produkt av maskinlæring. Det finnes mange ulike algoritmer som kan trenes basert på treningsdataene og resultere i en modell. Valg av algoritme henger nøye sammen med hvilke data du har, og hvilket problem du skal løse. Innen veiledet læring finnes det to hovedgrupper av problemer som algoritmen skal løse: regresjon og klassifikasjon. Noen algoritmer kan brukes til begge problemene, mens andre kan kun brukes til det ene. Innen klassifikasjon er målet å predikere korrekte merkelapper (labels) på nye data. Dette gjøres basert på binære eller kategoriske datatyper. Med regresjon er målet å predikere korrekte numeriske verdier på ny data (output), basert på gammel data (input). Dette kan gjøres på kontinuerlige (continuous) datatyper, altså tall som ikke er begrenset til binære eller kategoriske verdier. Vi skal kort gjøre rede for noen vanlige algoritmer, men først presenteres noen konsepter knyttet til egenskapene og prestasjonen til ulike algoritmer.

2.2.3.1 Bias, varians og overtilpasning

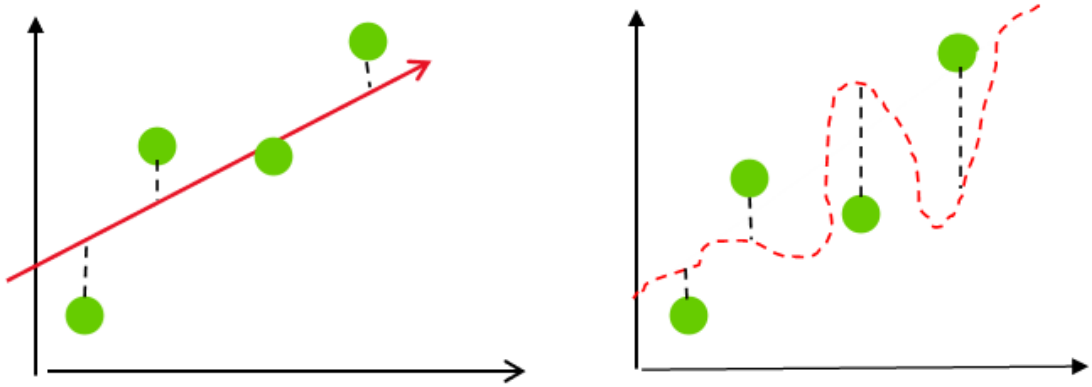
Til venstre i figur 2.5 ser du en algoritme av typen lineær regresjon. Denne linjen må være rett; den kan ikke bøye seg for å treffe de blå datapunktene. Avviket mellom linjen (prediksjonen) og datapunktene (treningssettet) kalles bias (Mehta et al., 2019). Derfor vil treningen av en modell basert på lineær regresjon føre til en del bias i dette eksempelet. Noe bias er helt vanlig med lineær regresjon, men hvis det blir veldig

høyt, vil den prestere dårlig i å predikere scoren av nye data, fordi den er dårlig til å fange opp mønsteret i treningsdataene. Det kalles undertilpasning (underfitting). Da bør man vurdere en annen algoritme som gir mindre bias og undertilpasning, hvis man vil ha presise prediksjoner. Til høyre i figuren har vi en ikke-lineær algoritme, som for eksempel Decision trees. Vi kan se at den er i bedre stand til å treffe de samme datapunktene. Avstanden mellom datapunktene og linjen er altså størst i modellen til venstre, og vi kan si at modellen til høyre er bedre trent og tilpasset treningsdataene.



Figur 2.5. Til venstre: lineær regresjon med moderat bias på treningsdata. Til høyre: Decision tree med ingen bias på treningsdata.

Vi har imidlertid ennå ikke vurdert hvordan modellene ovenfor presterer i å predikere scoren til nye data. Derfor er det viktig å teste algoritmen mot et testsett. Figuren under viser hvordan algoritmene presterer mot et testsett med nye grønne datapunkter.



Figur 2.6. Til venstre: lineær regresjon med moderat bias på testdata. Til høyre Decision tree med mye bias på testdata.

Her ser vi at modellen til venstre, basert på lineær regresjon, produserer omtrent like mye bias på testdataene som fra treningsdataene. Modellen til høyre, derimot, presterer mye dårligere og gir mer bias på testdataene i forhold til treningsdataene. Denne forskjellen i prestasjon som en modell har over ulike datasett, kalles varians. Derfor har modellen til høyre høyere varians enn modellen til venstre. Modellen til høyre gir faktisk mer bias enn den lineære modellen på testettet. Derfor kan vi si at den lineære modellen presterer bedre i å predikere scoren på nye data. Når predikeringen av nye dokumenter følger alt for tett opp til treningsdataene, slik som i modellen til høyre i figur 2.6, kaller vi det overtilpasning (overfitting) (Duboue, 2020, s. 14). Her *kopierer* modellen treningsdataene i stor grad, og det kan gi dårlige resultater når de nye dataene er ulike treningsdataene. Derfor er det viktig at en modell klarer å generalisere godt. Den perfekte algoritmen ville gitt lav bias og lav varians; den vil være presis og korrekt i forhold til treningssettet og testsettet. I praksis innebærer det imidlertid en trade-off mellom disse konseptene: blir det mer av den ene, blir det mindre av den andre. Valg av modell henger sammen med hvilke data den skal brukes på og hva som er formålet med gjenfinningen. Hvis modellen brukes på varierende og små datasett, er det bedre med en enkel modell som generaliserer godt (lav varians) (Mehta et al., 2019, s.10). Hvis modellen brukes på større, homogene datasett, er det bedre å velge en mer komplisert, ekspressiv og ikke-lineær modell som er bedre på å finne trendene og nyansene i treningsdataene. En modell som gir lav bias, og er tilbøyelig for overtilpasning, kan kalles en ekspressiv modell. Modeller med høy bias omtales som lite ekspressive. Kompliserte og avanserte modeller (*Complex models)*, som *neural networks, random forest* og *gradient boosting*, er ofte mer ekspressive enn enklere modeller (*simple models*) som lineær regresjon.

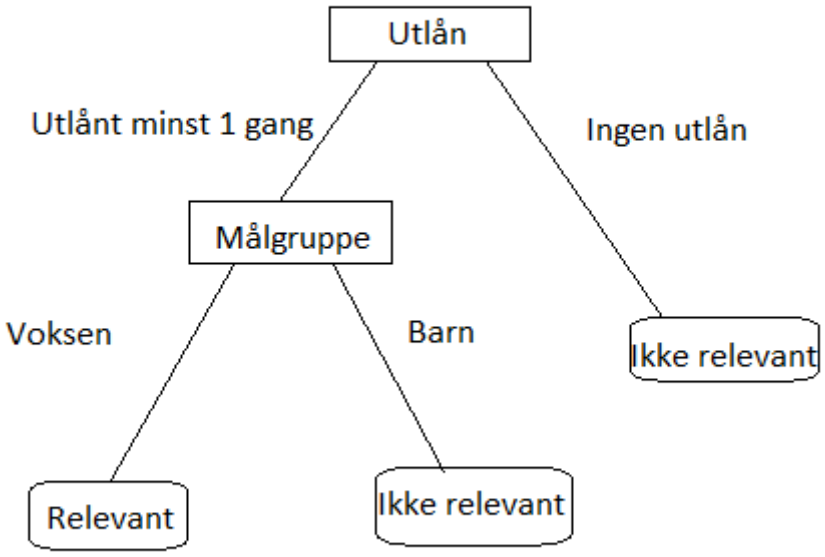
2.2.3.2 Regresjon

Lineær regresjon er en helt grunnleggende form for regresjon. Den undersøker forholdet mellom en avhengig variabel og en uavhengig variabel. Ved minste kvadraters metode (least squares) forsøker den å finne den beste rette linjen basert på sammenhengen mellom variablene. Metoden måler avstanden mellom datapunktene og den rette linjen og forsøker å redusere denne samlede avstanden mest mulig (G. James. et al., 2021, s. 62). Denne linjen kan brukes for å predikere nye verdier. Hvis man har utlånstall og sideantall på 10 bøker, kan man finne sammenhengen, den rette linjen, mellom disse variablene via lineær regresjon. Hvis man skal predikere utlånet av bok nr. 11, som er 50 sider lang, kan man følge linjens akse i linjediagrammet og enkelt finne den predikerte verdien for utlån, selv om verdien (prediksjonen) kan være noe unøyaktig.

Hvis vi skal undersøke forholdet mellom en avhengig variabel og flere uavhengige variabler, kalles det multippel regresjon. Dette gjelder hvis vi skal predikere utlånet basert på flere variabler som sideantall, antall eksemplarer i samlingen, antall likes, osv. Det er imidlertid ikke sikkert at alle variablene bidrar i predikeringen. Derfor er det viktig å sjekke variablene en etter en for å se om de korrelerer med den predikerte variabelen. Man kan måle korrelasjonen med f.eks. *adjusted* R^2 eller *Bayesian information criterion* (BIC). Ved metoden *forward selection* starter man på en null-modell (uten variabler) og legger til variabler hvor man starter med de som har størst statistisk betydning, helt til man når en grense man har satt. Det motsatte gjøres ved *backward selection* hvor man starter med alle variabler og tar bort en etter en etter hvilken variabel som har minst statistisk, eller prediktiv, betydning (G. James. et al, 2021, s. 79).

2.2.3.3 Decision trees

Decision trees kan brukes til både regresjon og klassifikasjon. Den bruker treningsdataene/featursene til å lage en trestruktur hvor verdiene for en feature forgrener seg nedover til man kommer til blader (leaf nodes) som er målverdien (target value). Dette lager klassifikasjonsregler som kan brukes til å klassifisere dokumenter utenfor samlingen (Baeza-Yates & Ribeiro-Neto, 2011, s. 294). I denne oppgaven kan problemstillingen være å klassifisere bøker i klasser som *relevant* eller *ikke relevant,* og målet er at modellen skal klare å predikere relevansen av nye dokumenter. Se figur 2.7 for hvordan det blir. Da kan en popularitetsfeature være *utlån den siste måneden*. Det kalles roten til treet. Da kan innholdet forgrene seg nedover som *ingen utlån* eller *utlån(>0)*, som betyr om bøkene har blitt utlånt den siste måneden eller ikke. Hvis vi da ser at alle bøkene som ikke har blitt utlånt er *ikke relevante*, vil algoritmen predikere at en ny bok som ikke er utlånt den siste måneden er *ikke relevant*, og da er den grenen ferdig. Hvis vi undersøker den andre grenen over bøkene som har blitt utlånt, og ser at den består av både *relevante* og *ikke relevante* bøker, må vi splitte/forgrene denne kategorien ytterligere, til vi står igjen med én verdi. Da kan vi velge en annen attributt/feature, for eksempel *målgruppe*. Da kan vi se at dokumentene forgrener seg mellom *voksen* og *barnelitteratur*. Hvis vi da ser at kun barnelitteratur-dokumentene er relevante, mens voksenlitteraturen er ikke relevant, betyr det at treet vil lære algoritmen å predikere at barnelitteratur som er blitt lånt den siste måneden er relevant, og voksenlitteratur som ikke har blitt lånt den siste måneden er ikke relevant.



Figur 2.7. Decision trees

Dette er det grunnleggende konseptet bak Decision trees. Det gir best resultat om forgreningen, eller splittingen, balanseres så godt som mulig, og om avstanden mellom roten og bladene ikke er lang (Baeza-Yates & Ribeiro-Neto, 2011, s. 296). Da er det lurt å velge attributter som bidrar til å redusere dette. I eksempelet ovenfor er veien fra roten til bladene 1 på høyre side, og 2 på venstre side, det er en liten ubalanse. Veien er kort mellom roten og bladene i begge tilfellene, noe som er positivt.

Den grafiske strukturen gjør det enkelt for mennesker å forstå klassifiseringen, og man kan se hvilke features som er viktigst, den foretar altså en implisitt *feature selection,* som handler om å velge ut viktige features, noe vi kommer tilbake til i kapittelet om features. Logikken i trestrukturen kan også enkelt konverteres til *if-then*-regler i et kodespråk (Rokach, 2016). Det å bruke enkle Decision trees gir derimot ikke like gode prediktive resultater som mange andre mer avanserte klassifisering- og regresjonsmetoder, og den kalles gjerne en *svak modell*. Hovedproblemet er at variansen gjerne er høy (James et al., 2021); den er tilbøyelig til overtilpasning og

generaliserer dårlig på nye data. En løsning er å kombinere et stort antall trær via metoder som *random forests* og *boosting.* Disse metodene forbedrer prediksjonene betraktelig. Det kalles *ensemble learning* når man kombinerer ulike metoder og (svake) modeller på denne måten (G. James. et al., 2021, s. 327). Vi skal kort forklare disse algoritmene nedenfor.

2.2.3.4 Random forest

Random forest bygger videre på en metode som kalles *bagging*. Her deles treningsettet opp, og det lages separate modeller/Decision trees av hver del. Deretter regner man gjennomsnittet av prediksjonene for hver del som blir grunnlaget for en ny modell (James et al., 2021). Dette kalles bagging og vil jevne ut den høye variansen man ofte oppnår ved enkle Decision trees. Dette er en forklaring av bagging for å løse et regresjonsproblem når man har flere treningssett, eller har mulighet til å dele det opp. Random forest lager også, i likhet med bagging, flere Decision trees basert på ulike deler av treningsdataene, men gjør det på en annen måte. Ved bagging vil de ulike trærne være ganske like fordi den sterkeste prediktoren i datasettet som oftest vil bli plassert som roten av treet (top split). Det fører til økt korrelasjon mellom de ulike trærne og små variasjoner i prediksjonene, noe som ikke bidrar like godt til å redusere variansen basert på gjennomsnittet av prediksjonene. Med Random forest vil derimot ikke den sterkeste prediktoren alltid utgjøre roten av treet. Kort fortalt sørger den for at det er mer tilfeldig hvilken variabel som blir roten til treet. Derfor får man prediksjoner av mer *tilfeldige trær*, derfor gir det mening å kalle modellen *random forest*. Logikken er at når det er mindre korrelasjon mellom trærne, er det også mindre samsvar mellom prediksjonene. Når disse kombineres via et gjennomsnitt, og det resulterer i en ny modell, vil variansen være mindre her enn ved bagging og enkle Decision trees.

2.2.3.5 LambdaMART

LambdaMART er en algoritme som *booster* Decision trees via ensemble learning. MART står for *multiple additive regression trees*, som bruker *gradient boosted Decision trees* for å predikere relevansen av dokumenter. Prinsippet bak gradient boosting er å sekvensielt legge til flere trær som stadig korrigerer prediksjonen til det foregående. Det nye treet trener på korreksjonen, som er avviket mellom den predikerte verdien og den reelle verdien, som det forrige treet har produsert. På denne måten lærer algoritmen sakte frem mot å gi bedre prediksjoner. Dette kombineres med en *cost function* fra LambdaRank (Burges, 2010).

Det er ikke oppgavens mål å gi en grundig oversikt over maskinlæringsmodeller og algoritmer, men å gi en kort presentasjon av noen hovedtyper som er relevante for denne oppgaven. Det kan nevnes at andre populære modeller er Navie Bayes (Lewis, 1998), Support Vector Machine (Joachims, 1998) og Neural Networks (Samek et al., 2021). Det krever relativt gode matematiske kunnskaper for å forstå hvordan noen av disse algoritmene fungerer på detaljnivå, og for å finjustere deres hyperparametre, om de har slike. Du trenger imidlertid ikke å være en ekspert innen matematikk for å bruke dem. Mange av algoritmene kan implementeres i Python med grunnleggende ferdigheter i programmering. Så lenge du vet hvordan du tester og evaluerer resultatet av algoritmene, er det altså fullt mulig å nyttiggjøre seg av dem til problemet du skal løse, uten å være en matematisk ekspert.

2.3 Evaluering

Eksperimentets målsetting er å forbedre rangeringen av trefflisten til Deichman ved å ta i bruk Learning to Rank. Da krever det at vi foretar en form for evaluering av resultatet for å vite om oppgaven har oppnådd dette. Dette skal vi ta for oss i et senere kapittel om fremgangsmåte. I dette kapittelet skal jeg kort redegjøre for noen grunnleggende og kjente evalueringsmetoder innen informasjonsgjenfinning.

2.3.1 Presisjon og fullstendighet

Presisjon og fullstendighet (precision og recall) er to grunnleggende evalueringsmål. Presisjon regnes som andelen relevante dokumenter av alle dokumenter på trefflisten. Så hvis vi har 5 relevante dokumenter i en treffliste på 10 dokumenter, hvor de andre 5 er ikke relevante, så har vi 50% presisjon. Fullstendighet er andelen relevante dokumenter som trefflisten genererte i forhold til det totale antallet relevante dokumenter. Så hvis det fantes 10 relevante dokumenter, men trefflisten kun genererte 5 av dem, så vil fullstendighet være 50%. Hvis trefflisten genererte alle, ville den vært 100%. Dette er et enkelt utgangspunkt for evaluering, og disse ordene kan også brukes mer uformelt og konseptuelt i å beskrive om en treffliste er presis og/eller uttømmende. Ulike informasjonsbehov vil stille ulike krav til presisjon og fullstendighet, hvor det i noen sammenhenger (known-item search) er viktigere å ha en presis treffliste, og andre ganger (exploratory search, research search) er det viktigere med en uttømmende treffliste. Mange av de andre evalueringsmålene bygger videre på presisjon og fullstendighet, for det er noen begrensninger knyttet til disse målene. For det første evalueres et dokument kun på en binær måte, som relevant eller ikke relevant, mens det i realiteten kan tenkes at et dokument er mer relevant enn et annet, selv om begge er relevante. En annen ting er at dokumentenes plassering på trefflisten ikke tas i betraktning i evalueringen. Hvis de 5 relevante dokumentene kommer først på trefflisten, vil det gi samme verdi for presisjon og fullstendighet som om de kom sist. I tillegg har vi ganske ofte ikke oversikt over hvor mange relevante dokumenter som finnes totalt sett til å kunne beregne en score. Derfor egner den seg dårlig til å evaluere systemer med lange trefflister, som vi ofte finner på weben (Baeza-Yates & Ribeiro-Neto, 2011, s.139).

2.3.2 Normalized Discounted Cumulative Gain (nDCG)

Normalized Discounted Cumulative Gain (nDCG) er et evalueringsmål som gir uttrykk for både ikke-binære vurderinger av relevans og et dokuments plassering i trefflisten. Det forutsetter at man har foretatt en gradering av dokumentene, for eksempel fra 0-3, hvor 0 er ikke relevant og 3 er veldig relevant. Disse verdiene akkumuleres ettersom man beveger seg nedover trefflisten. I det følgende eksempelet er tallene i parentes graderingene: en liste med 10 dokumenter hvor 5 dok. er *veldig relevant* (3), 2 dok. er *delvis relevant* (2), og resten *ikke relevant*(0). Det gir det en CG-score av 19 i siste posisjon. Fremdeles har vi ikke tatt dokumentenes plassering i betraktning, det er her *discounted* kommer inn. Denne virker slik at relevansverdien til et dokument dempes i forhold til dens plassering via en *discount factor*, som for eksempel en logaritme i base 2 av posisjonen til dokumentet, da deles discount gain med dokumentets plassering i trefflisten (Baeza-Yates & Ribeiro-Neto, 2011). Da blir DCG-scoren 4,24 istedenfor 19. For å sammenligne scoren mellom ulike systemer og samlinger er det til slutt nødvendig å normalisere scoren. Da kaller vi det nDCG, og resultatet er verdier mellom 0 og 1, hvor 1 er best resultat. Dette er et populært evalueringsmål, nettopp fordi det både uttrykker flere grader av relevans og tar et dokuments plassering på trefflisten i betraktning. Dette er imidlertid bare noen få av mange mulige evalueringsmål og evalueringsinnganger innen søk og gjenfinning.

2.4 Relevans og relevansvurderinger

Dette kapittelet har to hoveddeler. Den ene tar for seg teori knyttet til hva som menes med relevans innen søk og gjenfinning. Den andre delen tar for seg teori knyttet til hvordan man gjennomfører et eksperiment der det samles inn relevansvurderinger. Et steg i arbeidet med å lage en ny rangeringsfunksjon innebærer å produsere merkelapper for relevansen av søketerm-dokument-par. Vi skaffer merkelappene gjennom det vi kaller relevansvurderinger. Det er flere måter å skape dem på, som vi har diskutert tidligere i teoridelen. I denne oppgaven skal vi skaffe merkelappene basert på menneskeskapte relevansvurderinger (i motsetning til systemgenererte relevansvurderinger). Da er det store spørsmålet: hva menes med *relevans*? Og hva sier teorien om hvordan man bør produsere menneskelige relevansvurderinger av god kvalitet? Under følger relevant teori om disse spørsmålene.

2.4.1 Relevansbegrepet

Borlund (2003) skriver om relevansbegrepet i sammenheng med *evaluering av søk og gjenfinning*. Denne konteksten er viktig for å kunne evaluere prestasjonen av et søkesystem. Konteksten rundt relevansvurderingene er derimot ikke evaluering, men *trening* av en rangeringsmodell. Etter modellen er ferdig kan man derimot evaluere dens prestasjon i å rangere en treffliste. Likevel handler begge kontekster om det å vurdere relevansen av et dokument, selv om formålet er ulikt. Derfor er teorien i stor grad overførbar til relevansvurderingene de menneskelige assessorene står overfor. Borlund gir en oversikt over ulike typer av relevans. Hun viser til to hovedkategorier: *Objektiv* eller *systembasert* relevans på den ene siden, eller *subjektiv* eller *menneskebasert* relevans på den andre siden. Den systembaserte relevansen behandler relevans som et statisk og objektivt konsept, mens menneskebasert relevans behandler relevans som en subjektiv og rekonstruerende øvelse (Borlund, 2003). Denne subjektive relevansforståelsen innbefatter også et dynamisk og situasjonelt syn på relevans; at den er relativ og kan forandre seg ut ifra tid og kontekst.

Saracevic (1996) deler relevans inn i fem undergrupper. *System* eller *Algoritmisk* relevans går under hovedgruppen systembasert relevans og handler om samsvaret mellom søkeord og treffene det genererer i dokumentet; hvis søkeordet opptrer

mange ganger i et dokument, vil dokumentet ha en høy algoritmisk relevans-score. Dette er kontekstløs relevans, og det er et populært relevansmål innen informasjonsgjenfinning. *Topical* eller *subjektbasert* relevans handler om samsvaret mellom innholdet i søkeordet og innholdet i dokumentene. Dette handler om *aboutness* og transcenderer det *bokstavelige* samsvaret mellom søkeord og dokument fra algoritmisk relevans. Dette innebærer en subjektiv forståelse og vurdering av hva innholdet i spørringen og dokumentet er, og det tilhører dermed hovedkategorien subjektiv eller menneskebasert relevans. De tre andre typene: *kognitiv relevans, situasjonsbasert relevans og motivasjonsbasert relevans* tilhører også den subjektive og menneskebaserte hovedtypen for relevans. *Kognitiv relevans* tar brukerens kunnskap, perspektiv og informasjonsbehov med i vurderingen av relevansen mellom et søkeord og et dokument. *Situasjonsbasert relevans* handler om at brukere vurderer basert på hvordan de tolker oppgaven. Motivasjonsbasert relevans handler om brukerens motivasjon og målsetting i vurderingen.

2.4.2 Relevansvurderinger

Det er mange valg å ta når man skal lage et eksperiment som involverer menneskelig relevansvurdering. For det første bør problemstillingen og formålet med eksperimentet være tydelig, og på bakgrunn av den velger man et passende datasett, bestående av hva vi kaller dokumenter. Tilhørende til datasettet velger man passende *topics*, som i denne oppgaven vil være søketermer/spørringer. Du velger hvem som skal være vurderere, og videre er det viktig at assessorene gis en god og klar instruks slik at de skjønner oppgaven likt. Deretter er hvilke grader av relevans assessorene skal vurdere etter, et viktig valg. Det tradisjonelle målene presisjon og fullstendighet tillater kun binære grader av relevans, og i mange sammenhenger vil det være mulig, og behov for, å inkludere flere grader.

> [the] basic question about any given rating
> instrument is whether or not it has an optimum number of
> response categories or at least a number beyond which there
> is no further discrimination between the rated items . . . Too
> few response categories result in too coarse a scale and loss
> of much of the rater's discrimination powers. Conversely,
> too fine a scale may go beyond the rater's limited powers of

discrimination. (Jacoby & Matell, 1971, s. 495)

Dette sitatet uttrykker at det optimale antallet grader av relevans henger sammen med assessorenes evne til å skille mellom dokumenter, og dette kan variere fra kontekst til kontekst. Hvis assessoren (vurdereren) opplever å enkelt kunne skille dokumenter i 6 grader av relevans, bør det også være rundt 6 grader å velge mellom. Med en binær gradering i et slikt tilfelle vil nyansene forsvinne og informasjon vil gå tapt. På en annen side kan assessorene oppleve at det er vanskelig å skille mellom 6 grader av relevans i en vurdering. Kanskje klarer assessoren kun å skille mellom dokument hen liker og ikke liker, og at resten av skalaen dermed blir overflødig. I tillegg kan mange grader av relevans, som betyr flere mulige alternativer, føre til lengre betenkningstid og dermed lavere effektivitet i vurderingen. Tang (1999) sier at syv grader er optimalt, mens annen forskning (Iii, 1980) sier at det ikke finnes noen fasit som passer til alle situasjoner. Ifølge Borlund (2003) tar de fleste eksperimenter innen informasjonsgjenfinning i bruk tre grader.

Et annet valg å ta er hvilken rekkefølge dokumentene skal plasseres i; om den skal være tilfeldig eller ikke. Det kan tenkes at å plassere dokumentene tilfeldig vil fjerne et *trust bias* som forhindrer at assessorene tror at de første dokumentene er mer relevante. Sakai (2022) skriver at ved å ha en prioritert rekkefølge av dokumentene, hvor de mest relevante dokumentene (sett fra systemet, systembasert relevans) kommer først, skaper det en bedre forståelse hos assessoren av hva som er relevante dokumenter, noe som hjelper den videre vurderingen. Både med tanke på effektivitet og inter-assessor agreement (som handler om likhet mellom brukeres vurderinger) kom de frem til at det var lite forskjell på metodene. Men de kom frem til at det å bruke en prioritert rekkefølge skapte mer stabile og robuste resultater på tvers av ulike kilder og systemer, og derfor er en rangert treffliste å foretrekke (Sakai et al., 2022).

Vi har nevnt at det å bruke menneskelige relevansvurderere bidrar til relevansvurderinger av god kvalitet, sammenlignet med mer problematiske klikk-data, men menneskelige vurderere kan også gjøre feil, eller å sabotere bevisst. Relevansvurdering via *crowdsourcing* vil si å rekruttere eksterne og ukjente mennesker som enten frivillig eller imot betaling vil gjøre vurderingsjobben. Det

finnes plattformer som kan hjelpe deg med crowdsourcing. Dette er gjerne motsatsen til interne ekspertvurderere med god domenekunnskap om det de vurderer, og man tyr gjerne til crowdsourcing fordi det ofte er for dyrt og for ressurskrevende å få tak i ekspertvurderere. Ulempen med crowdsourcing er at man har mindre kjennskap til og kontroll over assessorene, og derfor øker sjansen for at de gir dårlige vurderinger. Det finnes imidlertid en mengde metoder som bidrar til kvalitetskontroll, eller som gjør dårlige assessorer og vurderinger synlige slik at de kan tas bort. Det er mulig å skaffe vurderinger av rimelig god kvalitet ved å benytte disse metodene (Samimi & Ravana, 2014). Vi nøyer oss med å nevne et par relevante mål som kan gi en indikasjon på kvaliteten av vurderingene. *Inter-rater reliability* er et mål over hvor likt assessorene har vurdert, mens *inter-rater agreement* er et mål over hvor likt en eller flere grupper har vurdert sammenlignet med andre grupper (Gisev et al., 2013). Inter-rater agreement er ekstra viktig når man evaluerer relevansvurderinger gjort via crowdsourcing, hvor man sammenligner den eksterne gruppen med en mindre ekspertgruppe som defineres som fasiten. Hvis resultatene fra crowdsourcing-gruppen er tilnærmet det av ekspertgruppen, øker det validiteten av resultatene. Innen ekspertgruppen er det interessant å undersøke inter-rater reliability, for å se hvor likt de har svart. Hvis svarene har en stor grad av homogenitet, vil det styrke og validere resultatene av vurderingene. Cohens kappa (McHugh M. L., 2012) er et kjent mål som kan gi gode verdier på inter-rater reliability.

2.5 Features

Vi har ikke funnet en norsk oversettelse av det engelske ordet *features* i maskinlæringsammenheng. *Egenskaper* eller *signaler* kunne vært kandidater, men vi har valgt å bruke det engelske ordet i denne oppgaven. Features er representasjoner av data som algoritmen trener på. Algoritmen kan noen ganger trene basert på rådata fra datainnsamlingen din. Du kan for eksempel ha en kolonne med verdier for *utlån* over en rekke bøker som er listet nedover i rader. Disse verdiene for utlån er rådata og noen algoritmer kan trene på disse, men det er ikke gitt at det vil resultere i en god modell å trene på disse dataene. Da kan det være smart å behandle rådataene, noe som kalles feature engineering. Når det kommer til tekst-matching som feature, kan du behandle hvert ord som en feature vektor, men da vil du ha svært mange feature vektor-dimensjoner, som vil være vanskelig å prosessere for vanlige systemer. En mulighet er da å lage features av disse rådataene via *featurization*, noe som vil redusere størrelsen på et dokument via en forenklet representasjon, som en TF-IDF-score for tekstlige data. Andre ganger kan du gjøre det motsatte, å utvide representasjonen ved å lage nye features basert på eksisterende features.

> Feature engineering is the process of representing a problem domain to make it amenable for learning techniques. This process involves the initial discovery of features and their stepwise improvement based on domain knowledge and the observed performance of a given ML algorithm over specific training data. (Duboue, 2020, s.7)

Feature engineering innebærer blant annet *feature generation,* hvor man lager features fra rådata, *feature transformation*, som handler om å utvikle eksisterende features, *feature selection,* som handler om å velge de viktigste featursene (Duboue, 2020, s. 8). Domenekunnskap handler om konteksten til dataene og målet med oppgaven. I denne oppgaven er konteksten biblioteket, og målet er å forbedre rangeringen av bibliotekmaterialet. En bibliotekar eller bibliotekarstudent kan sies å ha god domenekunnskap fordi hen ofte har god kjennskap til litteratur, søkesystem og bibliotekbrukere. Hen vil ha en fordel ved å analysere dataene gjennom *explorative data analysis,* som gjerne er første steg i å lete etter relevante features.

Det som menes med forbedring eller utvikling av features, kan innebære flere metoder. En av de mest vanlige er å normalisere verdiene via en skalering slik at verdiene blir mellom 0 og 1, eller -1 og 1. Dette er svært praktisk for å redusere variasjonen i verdier mellom ulike features. (Dubuoe, 2020, s. 36). Det er også mulig å vekte features ulikt. Det kan være smart hvis du forventer at visse features er mer informative enn andre og hvis maskinlæringen ikke plukker opp denne ulikheten basert på treningsdataene. Gode beslutninger rundt vekting bør komme fra domenekunnskap; altså kunnskap som ikke er tilgjengelig i dataene.

2.5.1 Kategorier og typer features

Det er vanlig å dele features inn i noen hovedkategorier. Features som er avhengige av søketermen kalles *query dependent features*. Dette kan være vektingsmodeller som BM25 som måler samsvaret mellom ord i dokumentet og ord i søketermen. Features som er uavhengige av søketermen kalles *query independent features*. Dette kan være en feature basert på dokumentets egenskaper, ofte representert via metadata. Et eksempel er en feature over hvor mange sider en bok inneholder. En tredje hovedkategori er brukeravhengige features som kalles *user dependent features*. Her baserer feature-scoren seg på informasjon om brukeren som interagerer med dokumentene. Vi kaller det også brukerdata.

Features kan ha ulike typer av verdier. Valg av modell henger sammen med dette, hvor noen modeller kun kan behandle visse verdityper. Den enkleste typen features er *binære*, da har de en av to mulige verdier. Disse kan brukes til å svare på ja og nei spørsmål, eller til å skille mellom to kategorier. Populære modeller som SVM (support vector machine) og logistisk regresjon kan kun predikere binære verdier (Duboue, 2020, s.28). En annen type features er *kategoriske*. Da er en feature en av flere forhåndsbestemte kategorier. Litterære sjangre som krim, fantasy og drama kan være eksempler på slike kategorier, gitt at vi har et fast antall sjangrer som tilgjengelige kategorier. Rekkefølgen betyr ikke noe. Modeller som løser klassifikasjonsproblemer egner seg godt til kategoriske feature typer. *Discrete* features er ordnede numeriske verdier i form av tall. Utlånstall er eksempel på en diskret verdi. De kan også mappes til å representere kategorier, enten ved at hver kategori får et tall eller at flere kategorier slås sammen under et tall. For eksempel

hvis årlig utlån for 3 bøker er henholdsvis 34, 58 og 99, er det mulig å mappe disse verdiene til tall fra 1-10 hvor de rundes opp eller ned til 3, 6 og 10. Rekkefølgen er ivaretatt, noe som skiller det fra kategorisering. Denne metoden kalles *discretization* og hjelper å forenkle dataene. *Kontinuerlige* verdier er ekte tall, ikke representasjoner slik som diskrete tall er, og det kan være distanse mål i kilometer. Det er ingen grense eller forhåndsbestemte kategorier for verdiene. Kontinuerlige features kan brukes i modeller som løser regresjonsproblemer. *Komplekse* verdier er kompliserte og unormale verdier. En dato, 20.03.2023, er et eksempel på en verdi som kan være vanskelig for en modell å prosessere og forstå meningen av. Andre eksempler er lister og datasett med verdier som henger sammen på andre måter. Ofte er løsningen å endre verditypene til noen av de som er nevnt ovenfor. For eksempel kan man endre den kompliserte verditypen for utgivelsesdato, 20.03.2023, til antall dager siden utgivelsen.

Gode features bør overholde noen kriterier. For det første bør de være informative og beskrive noe som gir mening for mennesker. Hvis man for eksempel lager en feature over antall lån av bøker, så vil det konseptet være forståelig for mennesker og dermed svare på det kriteriet. For det andre bør det forekomme mange verdier for en feature. Vi fortsetter med eksempelet om *utlån* av bøker som feature: hvis det kun er svært få bøker i samlingen som er utlånt (kanskje biblioteket og systemet er helt nytt), så vil antall utlån for de fleste bøkene være 0. Da vil ikke en slik feature være nyttig før utlånet tar seg opp og gir noen faktiske verdier. En feature med mange tomme felter gir mindre validitet og et dårligere treningsgrunnlag for algoritmen. Et siste kriterium er at det må være mulig å diskriminere mellom verdiene på en måte som er relevant for hva som skal predikeres (Duboue, s. 26). Hvis for eksempel alle bøkene i datasettet (merkelig nok) var utlånt 3 ganger, er featuren dårlig fordi det ikke er variasjon i verdiene til å diskriminere mellom bøkene i forhold til problemstillingen maskinlæringen skal løse.

2.5.2 Feature engineering

I dette avsnittet skal vi presentere noen kjente teknikker for hvordan man kan behandle features og gjøre dem klare for trening. Det finnes en stor mengde teknikker, og vi nøyer oss med å nevne et par av de mest brukte for å gi et inntrykk av hvordan denne prosessen er.

Normalisering

Et formål med normalisering er at meningsfylte variasjoner i featursene enklere skal kunne observeres (Duboue, 2020, s. 34). Normalisering av data brukes også til å fikse problemer med *outliers* og dominante features, hvor det reduserer deres negative effekt ved å omforme verdiene til en felles skala (Singh & Singh, 2022). *Outliers* er ekstreme verdier som stikker seg ut blant resten av verdiene. Dominante features er features som gir høyere scores enn andre features. De rå dataene kan være av ulike dimensjoner, som kg(vekt) og km(kilometer), som kan være vanskelige for algoritmer å håndtere godt. Rådata kan også ha store numeriske variasjoner som kan være vanskelige å tolke og sammenligne. For eksempel kan vi se på en feature over antall utlån av et dokument hos Deichman hvor verdien for et dokument er 50. Ved første øyekast, og uten domenekunnskap, er det vanskelig å vite om dette er en høy eller lav verdi, om det er en outlier eller dominant feature. Derfor er det ofte hensiktsmessig å normalisere verdier, men man må være bevisst på at det vil medføre at du mister informasjon som noen ganger er nyttig for algoritmen.
En enkel måte å normalisere på, er å skalere alle featursene, slik at de får verdier mellom 0-1. Da må man først finne maksimum og minimumsverdier for featuren. Så trekker du fra minimumsverdien av hver verdi og deler det på avstanden mellom maksimum og minimum (maksimum minus minimum). Med skalering må man imidlertid være bevisst på at *outliers* i dataene kan skape problemer. Hvis, f.eks. en bok ble lånt 10 000 ganger, mens ingen andre bøker ble lånt over 100 ganger, vil den nevnte skaleringsmetoden gjøre at de fleste verdier får svært lave verdier, noe som er uheldig for å få en god intuitiv forståelse og et godt sammenligningsgrunnlag av verdien. Duboue (2020) anbefaler å filtrere bort ekstreme verdier når man skal normalisere via den ovenfornevnte enkle skaleringsmetoden. Det bør imidlertid gjøres med omhu og basert på god domenekunnskap, ettersom det i noen kontekster er naturlig med outliers og noe modellen ønsker å ta til seg. Noen ganger vil

ekstreme outliers være et resultat av datafeil. 10 000 utlån av en bok ved et bibliotek er ganske urealistisk, så dette hypotetiske eksempelet ville tydet på at det var en datafeil.

One-hot encoding

Noen ganger har vi kategoriske features eller rådata som vi ønsker å bruke sammen med en algoritme som kun kan ta numeriske verdier. Kategoriske verdier, også kalt nominelle verdier, er verdier som målgruppen *barn, ungdom* og *voksen*. De har ingen rangering. Noen algoritmer kan ikke behandle kategoriske features på en god måte. De må gjøres om til numeriske verdier. One-hot encoding er en måte å gjøre dette på, hvor det skapes binære dummyvariabler 0 eller 1 for hver kategori. For å ta et eksempel: vi sier at Deichman opererer med 3 faste kategorier for målgruppe: *barn, ungdom* og *voksne*.

Bok	barn	ungdom	voksne
Pippi langstrømpe	1	0	0
Percy Jackson	0	1	0
Hamlet	0	0	1

Figur 2. 8: En matrise over One-hot encoding

I figur 2.8 ser man hvordan disse verdiene transformeres via One-hot encoding: Pippi (100), Percy Jackson (010) og Hamlet (001). Kun én kategori kan være riktig til enhver tid. Du vil alltid kun se et ett-tall og resten nuller, men ett-tallet vil alltid være på en unik plass slik at verdiene blir unike. Verdien vil da bestå av like mange tall som antall kategorier. Poenget er at hvis en bestemt verdi er mer betydningsfull enn de andre, gjør denne representasjonen det enklere for algoritmen å lære hva som er relevant (Duboue, 2020, s. 64).

2.5.3 Tidligere forskning om features

I denne delen går vi mer detaljert til verks og ser nærmere på forskning knyttet til features. Det er lite forskning om features benyttet i folkebiblioteksammenheng, men det finnes en del forskning om features brukt til søk og gjenfinning i fagbibliotek og i netthandel. Med utgangspunkt i disse skal jeg presentere og diskutere noen features som kan være relevante å bruke til treningen av en ny søkealgoritme i folkebibliotek.

Dirk Lewandowski (2009, 2010) gir en kategorisk oversikt over faktorer det er fornuftig å rangere etter i fagbiblioteksammenheng. Han deler de inn i 4 kategorier: *text matching* (tekstlige data), *popularity* (popularitet), *freshness* (Nyhetsverdi) og *locality* (plassering) og *other*. Jeg skal ta for meg hver av disse kategoriene og gi noen eksempler på features som kan være aktuelle i folkebiblioteksammenheng.

Tekst-matching

Tekst-matching handler om samsvar mellom søkeord og tekst tilknyttet dokumentene. Tekst-matching-features kategoriseres som query-dependent, fordi rangeringen er avhengig av søketermen. Tekst-matching kalles også statistiske mål av relevans (Behnert & Lewandowski, 2015). Det kan måles tekstlig samsvar mellom hele dokumentet(fulltekstsøk) og søketermen, eller man kan begrense matchingen til ulike bibliografiske felter. De ulike feltene kan vektes ulikt slik at match i noen felter gir en høyere score av relevans enn andre. Match i dokumentets tittelfelt regnes vanligvis som mest verdifullt. TF-IDF og BM25 er to av de vanligste tekst-matching-algoritmene som blir brukt. Lewandowski går ikke detaljert til verks i hvordan man skal lage features basert på tekst-matching, men sier det er avgjørende å kombinere mange faktorer (features) for at rangeringen blir best mulig (Lewandowski, 2009). Macdonald går derimot mer konkret til verks og viser til hvordan kombinasjonen av flere tekstlige vektingsmål, både TF-IDF og BM25, gir bedre resultater enn å kun bruke en av dem (Macdonald et al., 2013). Han forklarer resultatene ved å påvise tre effekter: *skimming effect,* som betyr at ulike modeller kompletterer hverandre i å hente forskjellige relevante treff. *Chorus effect*, som henter de samme relevante treffene, noe som bekrefter de relevante treffene. Og *the dark horse effect,* en feature som kun tidvis er brukbar. I tillegg viser han hvordan man kombinerer vektingen av ulike felter på best mulig måte. De kritiserer det som i hvert fall den gang var en

vanlig måte å gjøre det på: å måle TF-IDF for hvert informasjonsfelt. Dette kaller han *single-field* models.
Han kritiserer denne metoden og viser til at Robertson (Robertson et al., 2004) fant en annen og forbedret måte å summere vektingen av ulike felter sammen på. Dette var riktignok utenfor LTR-sammenheng. Macdonald går videre i å undersøke hvordan det fungerer i LTR-sammenheng, og han finner at det gir best resultat også der. Løsningen er å summere antall ord i et felt i proporsjon med feltets vekt, uten å lengdenormalisere for hvert felt, og deretter blande alle feltene i et ustrukturert samlet dokument og foreta vektingen (TF-IDF) deretter. Hvis for eksempel tittelfeltet er vektet med 5, og ordet *håp* står én gang i tittelfeltet, betyr det at *håp* repeteres fem ganger i den endelige TF-IDF vektingen av det ustrukturerte dokumentet, selv om det bare er ett tilfelle av ordet i hele dokumentet. Dette kaller han *field models,* og formelen er slik:

$$score(d, Q) = \sum_{t \in Q} w \left(\sum_{f} w_f \cdot tf_f \right),$$

Figur 2.9: Field model

Derfor viser de at det gir best resultat å kombinere flere tekst-matching-modeller, og i tillegg utføre vektingen av hvert informasjonsfelt slik som beskrevet i formelen til *field models*.

Behnert og Lewandowski (2015) skriver om flere tekstlige features som er query dependent. De henter inspirasjon fra features som er brukt på weben, og undersøker hvordan de kan brukes i biblioteket. Disse er:

-rekkefølgen av søketermene (*search term order)*
-distansen mellom søketermene (search term distance)
-søketermens posisjon (position of search term)
-dokumentlengden (document length)
-vektlegging av termer innen dokumentet (emphasis on term within document)

Spørsmålet er om disse kan være nyttige features i folkebiblioteksammenheng. En stor forskjell mellom dokumentene i folkebibliotek og på weben, og fagbibliotek, er at folkebibliotekdokumenter ikke har fulltekstsøk. Det søkes kun i metadata om dokumentene. Derfor er det mye mindre tekst å søke og matche i, sammenlignet med konteksten på weben og i fagbibliotek, hvor gjerne hele nettsiden eller forskningsartikkelen indekseres og søkes i. Derfor kan man argumentere for at slike mer finurlige og detaljerte tekst-matching-features vil være mindre nyttige i vår sammenheng enn på weben og i fagbibliotek. Men selv om forutsetningene ikke er like gode, så er det fortsatt fullt mulig at de kan bidra til å forbedre rangeringsmodellen i folkebiblioteksammenheng.

Distansen mellom søketermer handler om å gi en verdi på distansen mellom søkeordene i dokumentet/metadataene. Tanken er at hvis avstanden er kort, kan det tyde på at søkeordene i dokumentet henger semantisk sammen, på samme måte som de gjør i søkeordet, noe som vil være positivt. *Rekkefølgen av søketermer* handler om å lage en feature som gir en verdi over hvor godt søketermenes rekkefølge er ivaretatt i dokumentene. Det kan argumenteres på samme måte som for *distansen mellom søketermer* at et slikt mål vil si noe om hvor godt det samlede innholdet i søketermen er representert i dokumentet, noe som vil kunne favorisere dokumenter med best integritet i forhold til søketermen. Dokumentlengde kan være en indikator for relevans som er interessant å lage features av. Hva som er en ideell dokumentlengde, vil avhenge av konteksten. Hvis konteksten er fagbibliotek og forskningssammenheng er nok ideal-lengden av dokumentene på rundt 10-30 sider, fordi det meste av relevant forskningslitteratur befinner seg i det spennet. I folkebibliotek har vi ikke fulltekstsøk som i fagbibliotek, derfor vil disse søkene foregå på metadataene. Lengden på metadatainformasjon kan ved samme logikk (som fulltekstsøk) være en feature, kanskje ser man et mønster hvor korte eller ufullstendige beskrivelser av et dokument får en lavere score av relevans. *Emphasis on term within document* betyr å skape en featureverdi over match mellom søketermer og termer med en spesiell formatmessig utforming i dokumentet. Det kan være treff på ord med fet, kursiv eller understreking. I metadataene til Deichman bruker de understrekinger (lenker) i metadataene, for eksempel for forfatternavn og utgivernavn. Match mellom søketermer og disse understrekningene kunne da blitt brukt som feature. Dette kommer fra en antagelse om at formaterte ord på weben

ofte er viktigere enn uformaterte ord. Om det forholder seg slik i Deichman-metadataene, er derimot ikke like sikkert, så det er ikke sikkert dette vil være en nyttig feature i folkebiblioteksammenheng.
I websammenheng er antall lenker inn og ut av en side en kjent indikator for relevans som kan brukes som feature, uten at det er tekstlig match mellom lenkene og søketermen. I folkebiblioteksammenheng kan man telle disse lenkede understrekningene i metadataene fra Deichmankatalogen på samme måte, selv om søkeordet ikke engang matcher understrekningen, men da snakker vi ikke lenger om rangering basert på tekst-matching, da snakker vi om rangering basert på dokumentets egenskaper eller popularitet, og dette er features som er query independent. Videre skal vi snakke mer om denne kategorien features.

Popularitet

Rangering basert på popularitet er query independent. Popularitet er knyttet til menneskers bruk av dokumentet, altså deres brukeratferd. Kategorien bygger på et prinsipp om at det som er relevant for mange, er relevant for den enkelte, ref. konseptet *wisdom of crowds* (Sourowiecki, 2005). Det er imidlertid ikke alltid så enkelt, eller helt treffende, ettersom flertallet i teorien kan ta feil, men popularitetsmål fungerer som regel godt i praksis (Behnert & Lewandowski, 2015). Popularitetsmål trenger ikke bare å være basert på brukerdata fra bibliotekbrukerne. Det kan også være basert på hva som er populært blant bibliotekarene. Deres innkjøpshistorikk kan gi en indikasjon på det. Hvis de har kjøpt, katalogisert og stilt ut mange eksemplarer av et verk kan det bety at det er populært blant bibliotekarer, uavhengig om brukerne låner verket eller ikke. Antall eksemplarer av et verk kaller Behnert og Lewandowski *popularty based on authority* (2015). De antyder at dette tallet har en viss autoritet, at bibliotekarenes valg av innkjøp også indikerer kvalitet. Både Lewandowski og Behnert gir eksempler på popularitetsmål knyttet til fagbibliotek, for eksempel siteringsdata og nedlastinger. Slik bibliometrisk data kan vi ikke rangere etter i norske folkebibliotek fordi sitering og nedlastinger ikke er mulig eller relevant i denne konteksten, men vi tar med oss tankegangen og ser om det er andre popularitetsmål man kan bruke som features i folkebiblioteket.

Utlån er et av hovedmålene til folkebiblioteket. Derfor er utlånstall en naturlig kandidat som popularitetsfeature. Jamfør logikken fra *wisdom of the crowds*

(Sourowiecki, 2005), vil en bok med høyt utlån sannsynligvis være et interessant utlånsobjekt for andre. Utlånstall kan sammenlignes med en feature for antall kjøp, noe som er en vanlig og viktig popularitetsfeature innen netthandel. Antall fornyelser er også et mål som kan indikere at en bok fortsetter å være relevant for lånerne over lengre tid. På den andre siden blir de aller mest populære bøkene ofte ikke fornyet fordi det er venteliste på dem. Antall reserveringer er et annet popularitetsmål som kan indikere relevans. Her er det heller ingen grense for hvor mange reserveringer en bok kan ha. Tallet over antall utlån og fornyelser kan på sin side stoppe opp når antall eksemplarer ikke svarer til etterspørselen, eller når en låner beholder dokumentet over forfallsfristen. Det er også gjerne begrensninger på hvor mange ganger en låner kan fornye et lån. Antall reserveringer møter ingen slike hindringer, og derfor kan dette antallet bli svært høyt for nye og populære bøker. På en annen side kan det være mange bøker som aldri blir populære nok til at de blir reservert, selv om de har blitt lånt flere ganger. Det kan føre til en del tomme verdier for *reservert* som feature.

Antall klikk på dokumentet kan ses på som en indikator på relevans, fordi det viser at kilden er interessant nok til å bli klikket på. De dokumentene med flest klikk for en treffliste oppfattes gjerne som mest interessante for brukerne, og det kan brukes som feature hvis man har tilgang på klikk-data. Det er imidlertid sårbart for spam-klikking, hvor man ved ulike metoder kan skape kunstige høye klikktall. For å bøte på det er det en god ide å fjerne outliers (ekstreme verdier) i datagrunnlaget før treningen starter. Klikkbaserte relevansdata er også forbundet med det som kalles *trust bias*, hvor personer har en tilbøyelighet til å stole blindt på at søkesystemet plasserer de mest relevante treffene øverst på trefflisten (Joachims et al., 2007). Dermed oppnår de høyest rangerte dokumentene allerede en kunstig fordel i antall klikk. Joachims (2007) viser til to strategier for å løse dette problemet. Begge tar utgangspunkt i at man også skal ta i betraktning de dokumentene som *ikke* ble klikket på, hvor man regner ut en relasjon mellom klikkede og uklikkede dokumenter. De viser til at denne metoden gir et mer korrekt tall på relevans knyttet til antall klikk. Dette tallet kan brukes som en feature.

Dwell time måler hvor lang tid en bibliotekbruker tilbringer inne på dokumentsiden, etter hen har klikket på dokumentet. Det å fange opp dwell time kan være en god

indikator på relevans. I utgangspunktet er logikken at dokumentet blir mer relevant jo lenger tid som er brukt på dokumentet. Dette er også noe sårbart for trust bias, men i mindre grad enn klikkdata fordi brukeren vil klikke seg ut av dokumentsiden og etterlate lite dwell time om hen finner ut at dokumentet ikke er relevant. Derfor er det en feature som kan fungere godt sammen med klikk og motvirke klikk-biaset. Dwell time er også utsatt for spam hvor dokumentsiden kan stå åpen og generere høye verdier. Agichtein anbefaler å bruke en rekke varianter av dwell time (Agichtein et al., 2006). Disse er ment for weben, men det er interessant å la seg inspirere og se om noen av disse kan overføres til folkebiblioteksammenheng. For å nevne noen gjelder det time on page, time on domain, time on short url, cumulative time on page, average dwell time og dwell time deviation. Sistnevnte dwell time deviation er et mål som kan bøte på spamverdier hvor store avvik (outliers) blir oppdaget. Dwell time on page, cumulative time on page og average dwell time er nok de som har mest potensiale i folkebibliotek hvor det ville vært mulig å måle disse verdiene for en brukers interaksjon med dokumentsiden. Ifølge undersøkelsene i artikkelen til Agichtein (2006), gir dwell time features (en gruppe han kaller browsing features) en treffliste med bedre presisjon enn tekstbaserte og klikkbaserte features.

På nett kan en feature over *antall lenker inn og ut* være en god indikator på relevans. Hvor mange hyperlenker som peker inn mot en kilde kan altså være en god indikator på popularitet og relevans. I Deichmanpostene er blant annet forfatternavn og utgivere lenket slik at man kan trykke på dem og se en forfatters eller utgivers produksjon. Det er usikkert om vi kan overføre dette som en nyttig feature i folkebiblioteksammenheng. I tilfellene hvor postene har flere lenker enn de fleste andre, noe som vil være tilfellet med en post med mange forfattere (forfattere er lenket), er det en tvilsom logikk i at et verk med mange forfattere skal være mer relevant enn et verk med få forfattere.

En siste bemerkning angående popularitetsfeatures er at man i teorien kan hente indikasjoner om dette fra mange hold. Dette kan for eksempel gjelde bøkenes rating på *Bokelskere* (et nettsamfunn for bøker og lesere) eller salgstall fra forlagene. En utfordring er å få tilgang på disse dataene. Flere av de nevnte popularitetsfeatures er basert på brukerdata. Dette er ofte sensitive data, og man må være sikker på at det

ikke bryter noen retningslinjer knyttet til personvern før man benytter dataene til et Learning to Rank-eksperiment.

Nyhetsverdi (Freshness)

Med freshness mener Behnert og Lewandowski (2015) faktorer som går på hvor aktuell en kilde er. I mange sammenhenger er nyere kilder mer relevante enn eldre kilder. Dette er særlig nyttig innen vitenskapelige forskningsfelt som utvikler seg fort, men mindre viktig innen humaniora, hvor historiske kilder ikke utdateres på samme måte. I folkebiblioteksammenheng kan vi på samme måte si at dens vitenskapelig faglitteratur er mer sårbar for å utdateres, sammenlignet med historiske kilder eller skjønnlitteratur som et klassisk skuespill av Shakespeare. Features knyttet til freshness kan være forskjellige, noen eksempler er: første gang utgitt, nye utgivelser, dato for når kilden ble lastet opp i katalogen, eller siste gang utlånt. Datoer er kompliserte featuretyper som ikke kan brukes av mange algoritmer. Da må man omgjøre verdiene ved teknikker som har blitt nevnt i avsnitt 2.5.2 om feature engineering.

Plassering og tilgjengelighet (Locality og availability)

Locality kan oversettes med plassering, og availability med tilgjengelighet. Hvor kilden er plassert kan være en faktor for dens relevans. Mange biblioteker har magasin eller samlinger og avdelinger som ikke henger sammen med hovedsamlingen/hovedbiblioteket. Man kan lage en feature som sier om dokumentet er en del av hovedsamlingen eller står i magasin. Mange bibliotek samarbeider med andre bibliotek og kan tilby deres dokumenter via fjernlån. Brukerne foretrekker å få tak i dokumentene så fort som mulig, derfor er det en fordel med dokumenter som er tilgjengelige i en hovedsamling nær deg. Hvis man har tilgang på søkets geografiske posisjon (IP-adresse), kan man måle avstanden mellom søket og dokumentene. Denne verdien kan brukes som en feature, og denne informasjonen kunne i teorien vært presentert for assessorene, slik at de tok informasjon om tilgjengelighet og plassering i betraktning når de relevansvurderte dokumentene.

Tilgjengelighet vil i biblioteksammenheng handle om hvorvidt dokumentet er ledig eller utlånt. Det kan også handle om hvorvidt det er åpen tilgang eller en barriere for tilgang, noe som er mer aktuelt i fagbibliotek og dokumenter som vitenskapelige

artikler. Dette kunne vært en enkel binær feature som ledig/utlånt. Hvis assessorene vurderer dokumentet som ikke-relevant, fordi det ikke er tilgjengelig, og vi bruker det som feature, vil den nye algoritmen ha en tendens til å favorisere tilgjengelig litteratur.

Dokumentegenskaper (Content properties)

Det kan oversettes som egenskaper ved kilden. Det er mulig å lage features basert på egenskapene til kildene i bibliotekkatalogen. Hvis det gjelder kategoriske egenskaper ved dokumentet, kan de bli omgjort til numeriske maskinlesbare verdier via *one-hot encoding*. Det kan være hvilket format kildene har: bok, CD, DVD, lydbok, spill, osv. Formater som DVD og CD blir stadig mindre relevante, og kanskje er det et mønster algoritmen fanger opp ved å bruke slike features. Andre egenskaper som er relevante for et folkebibliotek kan være målgruppe. Noen bøker er relevante for barn, andre for voksne eller ungdom. Det er mulig å lage kategoriske features basert på dette. Det samme gjelder for sjanger. Språk er også en egenskap man kan rangere etter. Det er generelt naturlig å tenke at søketermer skrevet på et bestemt språk også foretrekker treff på samme språk, dette vil uansett tekst-matchingen mellom søketermen og dokumentet i stor grad sørge for.

Brukerdata (User background/ userdata)

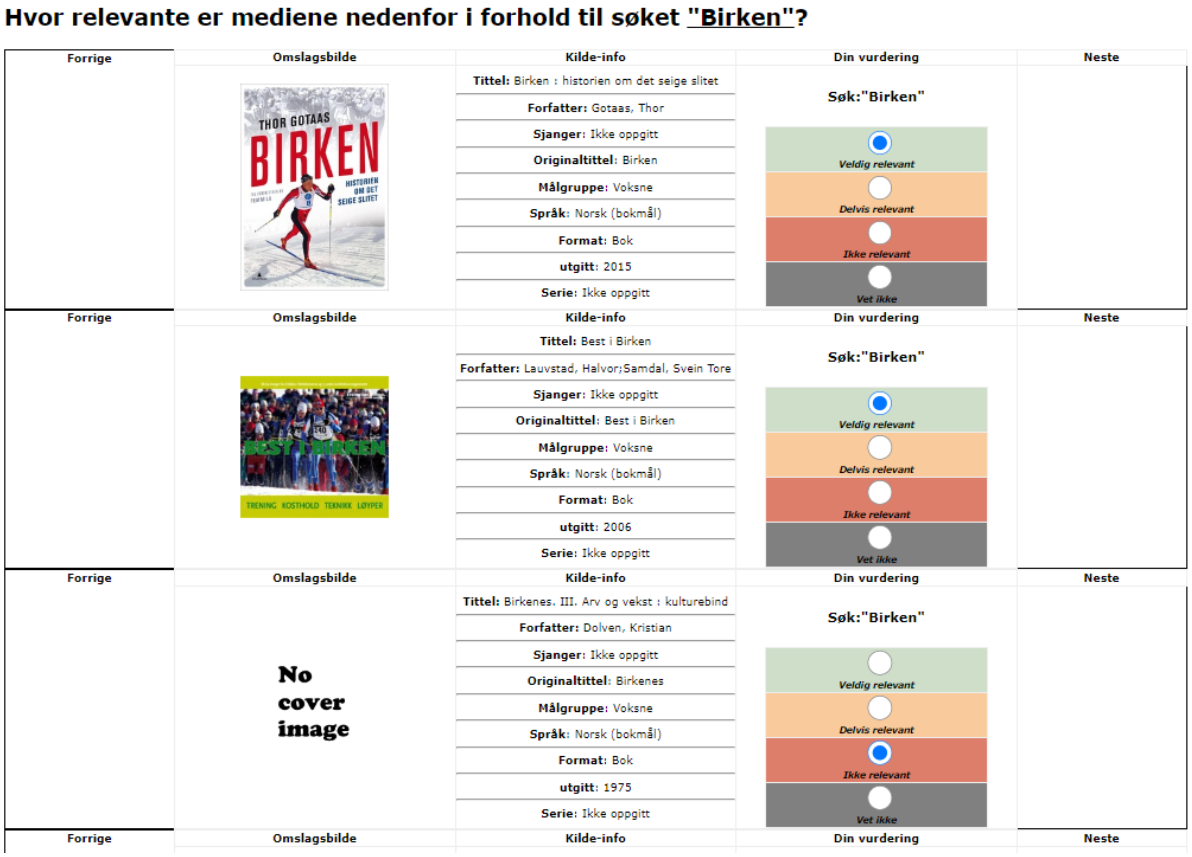
Denne kategorien kan kalles brukerdata på norsk. Det er vanlig på weben å bruke brukerdata til å gi personaliserte trefflister, anbefalinger og reklamer. Der godtar man gjerne at nettsidene benytter brukerdataene dine ved å akseptere cookies. I biblioteksammenheng er det i teorien mulig å gjøre det samme, men for å ha rett til å bruke brukerdata må vi ha samtykke fra brukerne. Via autorisasjon og innlogging hadde det vært mulig å samle brukerdata som klikk- og browsingatferd, eller, hvis grensesnittet tillater det, lister av favorittkilder. Det er mulig å lage features basert på disse brukerdataene, og i tillegg gi brukerne personaliserte trefflister. Denne oppgaven vil derimot ikke gå innpå dette i detalj. Dette prosjektet vil ikke bruke brukerdatabaserte features på denne måten. Vi har ikke rettigheter og tilgang på brukerdata fra Deichmans brukergruppe, men det ville vært interessant å undersøke i en annen oppgave så lenge det gjøres på en forskningsetisk forsvarlig måte. Vi har riktignok utlånsdata som er tilknyttet hvert eksemplar, som kan kalles brukerdata, men dette er helt anonyme data som ikke kan knyttes til spesifikke brukere.

3 Fremgangsmåte

I dette kapittelet om fremgangsmåte forklares hvilken metode som har blitt brukt for å skaffe relevansvurderinger. Det forklares hvilke metoder som har blitt brukt for å velge features og behandle features, og hvilke metoder vi har brukt for å trene, teste og evaluere resultatet av de nye rangeringsmodellene. Den overordnende metoden for prosjektet følger rammeverket for Learning to Rank (figur 2.3 fra teoridel), som inneholder rammeverket for hvordan man produserer modeller basert på maskinlæring (figur 2.4 fra teoridel).

3.1 Relevansvurderinger

Denne delen av masteroppgaven forklarer fremgangsmåten i å skaffe merkelapper (labels) via menneskelige relevansvurderinger. Vurderingene er utført på søketermer og dokumenter fra Deichmankatalogen, som ble gjort tilgjengelig via et grensesnitt med *Elasticsearch*. Dette er Deichmans gjenfinningsmodul, og vi bruker en snapshot av katalogen fra mars 2022. I utformingen av eksperimentet for relevansvurderingen er det mange valg å ta. I figur 3.1 ser man resultatet av grensesnittet hvor vurderingene fant sted. Mange av valgene begrunnes i tråd med teori som ble gjennomgått i teoridelen.



Figur 3.1: Screenshot av grensesnittet for relevansvurderingene.

3.1.1 Valg av innsamlingsmetode og assessorer

Vi valgte å gå for *menneskeskapte* relevansvurdereringer som innsamlingsmetode av merkelapper. De som relevansvurderer kalles assessorer. Motsatsen til dette er systemgenererte relevansvurderinger, som man oppnår ved å for eksempel bruke klikk-data som relevansindikatorer. Disse er i og for seg også menneskeskapte (mennesker har klikket på dokumentene), men ikke på en like direkte og intendert måte som ved vår metode. Menneskeskapte relevansvurderinger legger til rette for at hver vurdering er av høyere kvalitet, selv om det er krevende å skaffe en like stor mengde vurderinger som ved å høste klikk-data.

Vi rekrutterte personer i all hovedsak tilknyttet ABI - Arkiv bibliotek og informasjonsvitenskap ved OsloMet. Disse personene kan sies å ha bedre kunnskap om søk og gjenfinning og bibliotekdokumenter enn folk flest. Disse personene kan kalles interne, i og med at vi er knyttet til samme institusjon. Dette gjør vurderingene mer pålitelige. Et problem med crowdsourcing, som nevnt i teoridelen, er nettopp

upålitelige og slurvete vurderinger. Et internt utvalg reduserer sjansen for at det skjer, men det gir ingen garanti for at det ikke kan forekomme. For å gardere oss bedre mot slurv, misforståelser og generelt *dårlige vurderinger*, la vi opp til at grupper på syv personer skulle utføre de eksakt samme vurderingene. Slik kunne vi regne ut gjennomsnittsvurderinger, noe som bidro til å gi resultatene bedre konsensus og validitet. I tillegg bidro et høyere antall assessorer til å uttrykke ulike informasjonsbehov i vurderingene.

3.1.2 Søketermene

Deretter bestemte vi hvilke søketermer som skulle vurderes. Søketermene fra Deichmanfilen, som vist i figur 3.2, varierer mellom termer som refererer til navn, tittel, emner osv. Dette er typiske søkeord, og vi ville gjenspeile denne variasjonen i utvalget. Vi bestemte at hver assessor skulle vurdere 23 søketermer. Basert på filene vi fikk tilsendt av Deichman, utformet vi noen utvalgskriterier som vi diskuterer nedenfor.

Av søketermer inkluderte vi kun de som var skrevet i fritekst og som søker i hele dokumentet (all metadata). Det vil si at vi ekskluderte feltsøk som *tittel: "dragonball"*, eller *emne: "arkitektur"*.

	Tittel langt ned på trefflista	søkeord	plass nr	antall
2	Harry Potter og de vises stein	harry potter	21	9
3	Dragonball : 3-in-1. 34, 35, 36	tittel:"Dragonball"	12	8
4	Bittas votter : strikking hele året	tittel:"Strikking"	15	8
5	Metode og oppgaveskriving	fy	12	7
6	Karsten og Petra hilser på kongen	tittel:"Karsten og Petra"	18	6
7	Karsten og Petra på bondegård	tittel:"Karsten og Petra"	16	6
8	Den store Brillebjørn-boka : de fire første bøkene i én!	tittel:"Brillebjørn"	18	5
9	Probleme të shqipes standarde në Kosovë	albansk bok	14	5
10	Den store, grønne Brillebjørn-boka	tittel:"Brillebjørn"	16	5
11	Sinus matematikk 2P : lærebok i matematikk for vg1 : studieforberedende program	tittel:"Matematikk 2P"	12	5
12	Harry Potter og Føniksordenen	tittel:"Harry Potter"	12	5
13	Livets fem store : en historie om å lede seg selv og andre	emne:"Ledelse"	19	5
14	Slik timer du markedet : 50 signaler som har vist seg å fungere	Aksjer	12	5
15	Bleach. B.39. El verdugo	tittel:"Bleach"	13	5
16	Mira	Mira	13	5
17	Mira : #kreativklubben #familie #kyss	Mira	14	5
18	Den store boken om følelser	Følelser	30	5
19	Krokodillevokteren	Krim	22	5

Figur 3.2: Deichman-filen med titler, søkeord, plassering og antall.

Basert på Deichman-filen inkluderte vi kun søketermer med tilhørende dokument på en plassering mellom 10-30. Det er altså dokumenter vi har som målsetting å flytte

høyere opp på trefflisten. Vi ekskluderte topp 10 fordi de allerede er ganske godt rangert; brukere ser gjerne på de 10 øverste dokumentene**.** Vi ekskluderte søketermer med dokumenter som var plassert 30+ på trefflista for en søketerm, fordi det innen LTR handler om å rerangere trefflisten etter top-k-retrieval, for eksempel de 20 eller 30 beste treffene. I tillegg er det ifølge teori om feature engineering (Duboue, 2020) en god ide å fjerne *outliers* i datagrunnlaget for best mulig trening av rangeringsmodellen.

I utvelgelsen favoriserte vi dokument-søketermpar som hadde høyere antall klikk, ref. kolonnen ved siden av plasseringsdataen, fordi det er en indikator på hvor populært søket er. Vi ser det som ekstra nyttig for læringen av rangeringsfunksjonen å lære av populære søk fremfor mindre populære søk, selv om en viss variasjon er ønskelig, slik at læringen også baserer seg på mer perifere søk. Vi må derimot prioritere på grunn av kapasitetsbegrensninger.

Det er i utgangspunktet ønskelig at datagrunnlaget for læringen er så bredt som mulig med tanke på typer søketermer og dokumenter, for at den nye modellen skal fungere godt i flest mulige søkesammenhenger. Vi valgte derfor varierte søketerm-dokument-par fra Deichmanfilen som representere både skjønn og faglitteratur, og litteratur for de fleste målgruppene. Som nevnt i innledningen er det over en million dokumenter i Deichmankatalogen, blant annet fordelt over 39 formater og 116 sjangertyper. Oppgavens rammevilkår ga noen utfordringer med å legge til rette for en fullkommen variasjon mellom disse typene, hvor det ville blitt svært tid og ressurskrevende å få til. Variasjonen i dokumenter og søketermer er rett og slett for stor, og igjen måtte vi prioritere. Vi prioriterte formatet *bøker*. De andre formattypene som for eksempel spill, CD'er og DVD'er, har en svak representasjon i treningsdataene. I tillegg er noen aldersgrupper underrepresentert, særlig bildebøker og pekebøker for de minste. Vi ekskluderte for eksempel navn på regissører, og andre personer som har andre roller enn forfatter som søketermer til relevansvurderingen. Metadata med regissørnavn (illustratør og en mengde lignende tilfeller) ble ikke med i grensesnittet for vurderingen, og da kan assessorene ikke vite om en film ved navn *The Birds* er relevant for søketermen *Hitchcok*, med mindre de vet fra før at dette er Hitchcoks film, fordi navnet hans ikke var representert som metadata i grensesnittet. Vi tok denne avgjørelsen fordi vi ikke kunne overlesse

assessorene med fullstendig bibliografisk informasjon om hvert dokument i grensesnittet hvor de vurderer. Da ville assessorene måtte ta inn mye informasjon og bruke lang tid på hver vurdering. Med kun 30 rekrutterte assessorer var vi avhengig at de vurderte hurtig og produserte et tilstrekkelig antall vurderinger for å ha nok merkelapper til maskinlæringen (nok treningsdata). Vi så det nødvendig med liten trade-off i kvalitet (variasjon) for kvantitet.

3.1.3 Dokumentene

Per søketerm ba vi assessorene vurdere 16-20 dokumenter. Grunnen til variasjonen er at ikke alle søketermer produserer en treffliste på 20 dokumenter. Noen ganger stopper den litt før. Ellers, hvis trefflisten består av 20+ dokumenter, som den vanligvis gjør, vil de vurdere 20 dokumenter. Trefflisten som de vurderer, ligner den rangerte trefflisten man ville funnet hos Deichman mars 2022. Men vi har modifisert den litt, hvor kun ett uttrykk fra hvert verk er representert. Tanken var at vi var mer interesserte i å fange opp forskjellene i relevans mellom verk snarere enn mellom forskjellige versjoner av det samme verket. Vurderingen av relevans av ulike uttrykk kan likevel bli fanget opp på tvers av verk. For eksempel hvis boken (verket) *Sult* på norsk får en god score for relevans, mens boken (verket) *Min Kamp* på italiensk får en dårlig score for relevans, kan det tyde på at det er det italienske *uttrykket* som gir dårligere score, noe algoritmen kan plukke opp. Det ville naturligvis vært enklere for algoritmen å plukke opp dette mønsteret om *Sult* på italiensk ga dårlig score sammenlignet med den norske utgaven, men vi har nedprioritert å fange opp dette som følge av den begrensede kapasiteten til assessorene.
Ifølge Sakai (2022), som nevnt i teoridelen, er det en liten fordel at trefflisten er rangert. Derfor har vi også rangert trefflisten som de skal vurdere. Det innebærer at hvis det dokumentet (fra Deichmanfilen) som skal løftes befinner seg mellom 20.-30. plass, vil det ikke bli vurdert, men det er heller ikke viktig for treningen at akkurat det dokumentet blir vurdert.

3.1.4 Gradering

Assessorene fikk velge mellom tre grader av relevans: *ikke relevant*, *delvis relevant* og *veldig relevant*, i tillegg til *vet ikke*. Vi tenkte at en binær gradering mellom *ikke relevant* og *relevant* ble for begrensende, at en bibliotekbruker ofte kan komme over kilder på trefflisten som hen tenker er i gråsonen mellom noe hen kunne ha lånt og ikke lånt. Derfor introduserte vi en mellomverdi som vi kalte *delvis relevant*. Tanken er at det skal representere dokumenter som er av en viss interesse, men ikke fullt så interessant som det beste; dokumenter som ville blitt studert nøyere og klikket inn på, og kanskje til og med lånt. Det er dokumenter som er lenger nede på trefflista, ca. nedre halvdel. De *veldig relevante* representerer de aller beste treffene på øvre halvdel av trefflisten. De som blir klikket på og lånt. De *ikke relevante* dokumentene er uaktuelle som utlånsobjekt og bør være nederst eller helt borte fra trefflista. Vi vurderte å ha enda flere graderinger. Det kunne vært aktuelt med en verdi *relevant* mellom *delvis* og *veldig relevant*. Spranget mellom de graderingene, særlig når de er kalt delvis og veldig, kan virke stort, og det kunne virket intuitivt riktig å fylle det med *relevant*. Det kunne gitt grunnlag for å rangere trefflisten enda finere. Vi fikk også en tilbakemelding fra en vurderer som etterlyste dette. Vi holdt oss til tre grader delvis fordi det kan gi raskere vurderinger å velge mellom tre grader enn fire, og vi var opptatt av kvantitet. Teorien viser også til at tre vurderinger er vanlig innen informasjonsgjenfinning. Samtidig tenker vi at tre grader i de fleste tilfeller vil være tilstrekkelig for konteksten av å vurdere bibliotekdokumenter. Hvis man for eksempel skal vurdere søketermen *Jørn Lier Horst* (krimforfatter), kan det være utfordrende å dele dokumentene inn i fire kategorier basert på relevans. Det kan derimot tenkes at to eller tre kategorier er nok for å skille mellom de ulike krim-bøkene hans. For andre søketermer, som “selvhjelpsbøker”, kan det derimot lettere tenkes at resultatlisten kan inneholde bøker som er mer eller mindre relevante i forhold til søket og emnet. Vi tenker likevel at tre grader virker passende i konteksten av folkebibliotek. I tillegg kan det øke effektiviteten i produksjonen av vurderingene med færre svaralternativer. I etterbehandlingen konverterer vi derimot resultatet inn i fem grader. Det gir algoritmen et bedre grunnlag for å finrangere trefflisten. Dette har vi gjort via *rounding*, som er en enkel måte å transformere et ekte tall til heltall på. Da har vi *rundet* gjennomsnittet (mean grade) av vurderingene av hvert søketerm-dokument-par slik: rounded = round((2*meangrade)).

3.1.5 Instruksen

Det er viktig å gi assessorene en form for instruks før de vurderer. Denne bør være tydelig formulert slik at alle forstår oppgaven, og at de forstår den likt. Instruksdokumentet består av flere deler. Først er det en del som kort forklarer problemstillingen til oppgaven og hva som er formålet med relevansvurderingene knyttet til den. Slik får assessorene en bedre forståelse av hva vurderingene deres bidrar til. Denne informasjonen kan bidra til å samle assessorene rundt en felles situasjon og motivasjon for vurderingene, som beskrevet i teoridelen som *situational* og *motivational relevance*. En annen del av instruksen gir teknisk informasjon om grensesnittet og omfanget av vurderingene. Deretter inneholder instruksen en veiledning og definisjon av hva som menes med relevans, og en forklaring av de ulike gradene av relevans. Instruksen foreslår at assessorene velger seg et alminnelig informasjonsbehov for hver søketerm. Det gis noen eksempler på dette: hvis søketermen er “Hamsun”, så vil et alminnelig informasjonsbehov for eksempel være å ville ha originalverkene hans. Da vil biografier og lignende sekundærlitteratur bli *ikke relevant* for det informasjonsbehovet. Det betyr at relevansvurderingene vil være subjektive vurderinger. Assessorene fikk tilgang på instruksen og grensesnittet, og ble gitt 3 uker på å svare. Relevansvurderinger er bare et ledd i LTR-prosessen. Videre i metodekapittelet beskrives prosessen rundt utvelgelsen, behandlingen, testingen og evalueringen av features.

3.2 Fremgangsmåte for feature engineering, trening og evaluering

I teorikapittelet ble det presentert og diskutert forskning om features som kan være relevante å bruke i folkebiblioteksammenheng. I tillegg har vi intuitivt gått gjennom tilgjengelige data og sett på hva som kan brukes som fornuftige features. Dette intuitive arbeidet kalles *explorative data analysis* (EDA) (Duboue, 2020). Vår intuisjon er forankret i god domenekunnskap om søk og gjenfinning i folkebibliotek. På denne måten har vi kommet frem til flere features vi ønsker å teste ut.

3.2.1 Features

Her er våre features inndelt i fem kategorier:

- **Innholdstype**
    - Fag, fiksjon
    - Match på søkeord fiksjon
    - Roman, ikke-roman
    - Match på søkeord roman
    - Spenningslitteratur
- **Tekst-matching**
    - McDonalds field model
        - TF-IDF
        - BM25
    - TF-IDF
        - Tittel
        - Forfatter
        - Beskrivelse
    - BM25
        - Tittel
        - Forfatter
        - Beskrivelse

Popularitet

- Antall eksemplarer
- Utlån

    - Antall fornyelser
- **Nyhetsverdi**
    - 0-2 år
    - 3-5 år
    - 6-10 år
    - Eldre
- **Brukere (målgruppe)**
    - Voksen
    - Ungdom
    - Barn

Innholdstype

Innholdstype er en query independent kategori. Dette er kategoriske data som vi har nødt til å gjøre om til numeriske data for at de skal bli leselige for algoritmene som brukes i dette eksperimentet (de presenteres i neste kapittel, 3.2.2). Dette har vi løst ved å lage binære dummyvariabler 0 og 1 for hver feature. For feature *fag* blir verdien 1 hvis dokumentet er faglitteratur, 0 hvis ikke. For feature *fiksjon* blir verdien 1 hvis dokumentet er fiksjon, 0 hvis det ikke er det.
Videre har vi en feature for om boka er en roman eller ikke, og vi brukte samme metode som ved *fag* og *fiksjon.*
Vi ville ha features for sjangre, helst en eller flere som fanget opp alle. Det finnes imidlertid 116 ulike sjangertyper i Deichmankatalogen, og vi har ikke et datagrunnlag (relevansvurderinger) som representerer alle disse sjangrene. Derfor var det ikke nyttig at vi tok med alle, og vi valgte noen få som representerte dataene våre godt. Vi laget en feature som heter spenningslitteratur. Denne består av flere sjangertyper som vi anser for å være spenningslitteratur. Det gjelder: krim, spenning, thriller, grøss og detektivfortellinger. Igjen brukte vi dummyvariabler slik at hvis et dokument utgjør noen av disse sjangrene, vil verdien være 1, hvis ikke, vil verdien være 0.
I tillegg har vi noen ekstra features som gir verdiene 1 eller 0 for om søkeordene matcher navnene på kategoriene. Hvis noen søker *roman* vil det bety 1, ellers 0. Dette gjelder for *roman* og *fiksjon*. Tanken er noen kan finne på å søke *hamsun roman*. Dette vil gjøre at modellen har mulighet til å favorisere Hamsuns romaner fremfor ikke-romaner.

Tekst-matching

Tekst-matching er en query dependent featurekategori som gir en score for samsvaret mellom søketermen og dokumentet. Vi brukte både vektingsmålene BM25 og TF-IDF for å oppnå en skimming, chorus og dark horse effekt, som beskrevet i teoridel (Macdonald et al., 2013). Da matchet vi i tittelfeltet, forfatterfeltet og beskrivelsefeltet for både BM25 og TF-IDF. Det regnes ut en score for BM25 og TF-IDF for hvert felt. Dermed er hvert felt en feature, og det er hva Macdonald kaller *single field model*.
Vi brukte også hans *field model* med ulike varianter med ulik vekting av feltene. Det regnes ikke en score for hvert felt separat, slik som med metoden nevnt ovenfor, men de vektede verdiene behandles i et ustrukturert dokument med BM25 og TF-IDF. For eksempel hvis søkeordet er *Hamsun,* og vi får treff på det søkeordet i forfatterfeltet med variant 1, så vil termen *Hamsun* bli representert 5 ganger (fordi vekten er 5) i det ustrukturerte dokumentet som blir behandlet av BM25 og TF-IDF. Derfor gir det to features per variant (en for BM25 og en for TF-IDF).

Variant 1:
forfatter = 5
tittel = 3
beskrivelse = 1

Variant 2:
tittel = 5
forfatter = 3
beskrivelse = 1

Variant 3:
emne = 5
tittel = 4
forfatter = 3
beskrivelse =1

Variant 4:

tittel = 5
emne = 4
forfatter = 3
beskrivelse = 1

I testingen kombinerte vi alle features. Macdonald viser til at *field models* gir best resultat. For å holde oss mest mulig tro til Macdonalds konklusjon om field model, skulle vi kun brukt de fire variantene til tekst-matchingen og ikke kombinert den med *single field model*. Likevel gikk vi for en kombinasjon som er et kompromiss, hvor metoden field model påvirker resultatet av tekst-matchingen, men vi brukte den ikke på en rendyrket måte slik han anbefaler i forskningsartikkelen. Vektingstallene vi har kommet frem til med *field model* kan også avvike noe fra hans foreskrevne metode.

Popularitet

Vår verdi av *antall eksemplarer* er tall på hvor mange eksemplarer det finnes av hver manifestasjon av *ett* uttrykk av et verk. For eksempel hvis verket *Sult* av Knut Hamsun har 5 uttrykk (på fem ulike språk), og hvert uttrykk har 2 manifestasjoner, (1.utg. og 2. utg.), og hver manifestasjon har 2 eksemplarer (fysiske bøker), da blir featureverdien 4.

Utlån er et annet viktig popularitetsmål. Våre tall er fra mars 2022, og vi målte totale lån for hvert dokument fra alle filialer og alle år. Vi slo sammen eksemplarer på samme måte som vi gjorde med feature *antall eksemplarer*: ett uttrykk av et verk, alle manifestasjoner av ett uttrykk, og alle eksemplarer av alle manifestasjoner. Dette kan føre til svært høye utlånstall for de mest populære bøkene, og av eldre bøker hvor utlånet har akkumulert seg over flere år.

Antall reserveringer er et popularitetsmål som vi brukte som feature. På samme måte som ved *antall utlån* og *antall eksemplarer* telte vi alle eksemplarer fra alle manifestasjoner av *ett* uttrykk av et verk.

For å unngå at et dokument med 100 utlån anses for å være dobbelt så relevant som et dokument med 5 utlån, eller 100 ganger så relevant som et nytt dokument med kun 1 utlån, log-transformerte vi alle popularitetsdata slik: log10 av 1+antall utlån.

Denne transformasjonen vil dempe effekten av høye verdier, hvor log10 av 100 er 2, mens log10 av 50 er 1,69. Da ser man at økningen ikke lenger er dobbel. Man kan argumentere for at denne mer moderate økningen samsvarer bedre med menneskelig forståelse av relevans. Imidlertid beholdt vi også rådataene som features, slik at vi gjorde et slags kompromiss med to sett av popularitetsfeatures. Det gjør at vi til sammen har 6 features i kategorien popularitetsdata (3 log-transformerte, 3 av rådata).

Nyhetsverdi

Vi ønsket å lage features basert på dokumentets nyhetsverdi, eller freshness, som det også kalles. Datoverdier som 02.04.2019 er kompliserte featuretyper som må gjøres om til binære, kontinuerlige eller diskrete verdityper for å bli leselige i de fleste kontekster av maskinlæring. Dette løste vi med å lage binære features av noen datointervaller. Vi lagde en binær feature hvor dokumenter som er gitt ut de siste to årene fikk verdien 1, resten 0. Vi brukte en versjon av katalogen fra mars 2022 og regnet to år tilbake fra det tidspunktet. Vi målte kun utgivelser av nye verk og uttrykk, så vi målte ikke utgivelsen av en ny manifestasjon. De andre datointervallene er 3-5 år, 6-10 år, og *eldre*. Sistnevnte kategori inneholder alle verk som er utgitt før de siste ti årene.

Brukere (målgruppe)

Det finnes 9 kategorier *målgrupper*: 0-2, 3-5, 6-8, 9-10, 11-12, 13-15, 16-17, barn og ungdom, voksne som Deichman bruker i katalogiseringen sin. Vi forenklet dette og lagde en binær feature for om dokumentet er kun for voksne eller ikke. Hvis dokumentet kun har målgruppeverdien *voksne* blir featureverdien 1. Hvis dokument har andre målgruppeverdier, selv om *voksne* også opptrer (man kan ha flere målgruppeverdier per dokument), blir featureverdien 0. Vi brukte samme metode for ungdom, det vil si dokumenter som inneholder målgruppene 11-12, 13-15, 16-17 og barn og ungdom. Hvis noen av disse verdiene opptrer i dokumentet, og ingen verdier utenfor opptrer, blir featureverdien 1, ellers blir verdien 0. Vi brukte samme metode på barnelitteratur og målgrupper fra 10 år og nedover.

3.2.2 Algoritmer

Vi trenet algoritmene LambdaMART og Random forest. Disse er beskrevet i teorikapittelet. Vi kjørte alle tester via begge algoritmene. På den måten kunne vi sammenligne resultatet fra de ulike algoritmene og konkludere med hvilken som presterer best i å løfte titlene fra Deichman-filen. Algoritmene er ferdige og tilgjengelige og ble implementert med kodespråket Python.

3.2.3 Trening

Når featursene er ferdigbehandlet, merkelappene for relevans er produsert, og algoritmene er valgt, da er alt klart for å starte treningen. Implementeringen av alle disse bestanddelene ble gjort via programmering i kodespråket Python i Jupyter notebook. Vi vil ikke presentere kode i denne oppgaven, men forklare hvordan vi har gått frem, og senere presentere resultatene.

For å evaluere modellen må man spare en del av treningsdataene som kun brukes til testing. Vi sparte 20% av treningssettet og trente med resten. Vi trener opp flere forskjellige modeller og sammenligner resultatene. Vi trener modeller basert på hver enkelt featuregruppe, alle featuregrupper samlet, og alle kombinasjoner av featuregrupper. Vi trente de ulike kombinasjonene med både LambdaMART og Random Forest. Evalueringsmålet nDCG evaluerte resultatet av testsettet. Den femtedelen (20%) man velger som testsett, kan i teorien være en dårlig representasjon av datasettet. Kanskje er en annen femtedel bedre egnet som testsett, eller kanskje er den femtedelen du velger som testsett spesielt viktig for treningen og bør være treningsdata istedenfor. En måte å løse denne problematikken på er å kryssvalidere dataene. Når man har fem deler, vil det bety at det trenes fem ganger, hvor den femtedelen som brukes som testdata, er en ny del av treningssettet for hver test, slik at man får trenet på alle delene. Deretter regnes det ut et gjennomsnitt av resultatet. Dette gir en mer pålitelig evaluering, og man får trent på alle delene av dataene.

Alle features ble inndelt i tre grupper. En gruppe er *popularitetsfeatures*. Denne utgjør de 6 popularitetsfeatursene som ble beskrevet tidligere i kapittel 3.2.1. En annen gruppe kaller vi *kategoriske features*. Det gjelder alle features innen litteraturtype,

målgruppe, og nyhetsverdi. Den siste gruppen av features er features knyttet til tekst-matching. Den utgjør alle tekst-matching-features som tidligere er presentert.

Vi kommer ikke til å teste hver enkelt feature, og vi vil ikke analysere *feature interactions* - synergieffekter hvor flere features fungerer spesielt godt sammen om å predikere riktig merkelapp. Dette hadde imidlertid vært interessant å undersøke, men vi har begrenset med treningskapasitet innenfor rammene for denne oppgaven.

3.2.3 Evaluering

Hovedevalueringen er å evaluere hvordan dokumentene fra Deichmanfilen har forflyttet seg på trefflisten med den nye rangeringsmodellen basert på maskinlæring. Målet er at de skal forflytte seg oppover trefflisten. Vi kan måle plasseringen av kildene med de nye rangeringsmodellene som vi lager, og vi kan sammenligne denne med deres tidligere plassering. Vi lager kode i Python som gjør at vi får verdier på hvor mange dokumenter som har flyttet seg oppover trefflisten, hvor mange dokumenter som har flyttet seg nedover trefflisten, og hvor mange som ikke har flyttet seg. Vi regner ut en verdi *totaldiff* som forteller oss hvor mange plasser dokumentene har flyttet seg både opp og ned. Minusverdier av totaldiff er det vi ønsker fordi det betyr hvor mange plasser dokumentene har beveget seg mot 1, som er beste plassering på trefflista. Dette blir det endelige evalueringsmålet som forteller oss hvorvidt vi har lykkes i arbeidet med å lage en ny maskinlært rangeringsmodell via LTR som løfter dokumentene fra Deichmanfilen.

Det å skape gode merkelapper (labels) er en viktig del av arbeidet med å produsere de nye rangeringsmodellene. Derfor undersøkte vi også resultatene av relevansvurderingene. Vi så blant annet etter om resultatene ga en viss variasjon, noe som skaper et rangeringsgrunnlag. Hvis alle dokumenter er rangert veldig likt – i få kategorier - gir det dårligere rangeringsgrunnlag. Det ble også tatt noen stikkprøver for å se om de dokumentene som befinner seg i Deichmanfilen har blitt klassifisert som *veldig relevant*, *delvis relevant*, eller *ikke relevant*. Det er positivt om dokumentene fra Deichmanfilen har blitt klassifisert som *veldig relevant*, fordi det legger til rette for at deres egenskaper blir favorisert av algoritmen og at de blir rangert høyere. Det er imidlertid ingen automatikk i det, eller en nødvendighet for et

godt resultat, for de er avhengige av gunstige resultater fra de andre vurderingene også for at algoritmen skal påvirkes i deres favør.

4 Resultater

Først presenteres resultatene fra relevansvurderingen. Deretter presenteres resultatet av testingen av features, og til slutt undersøkes det om den nye modellen klarer å løfte dokumentene fra Deichman-filen høyere opp på trefflista.

4.1 Resultatet av relevansvurderingen

Eksperimentet var designet slik at grupper på syv skulle vurdere 23 søk med 16-20 dokumenter. Det var i praksis mulig for assessorene å avbryte underveis. Av de 31 rekrutterte assessorene, var det 22 som vurderte alt eller nesten alt av materialet. I gjennomsnitt ble hver søketerm vurdert av 5 assessorer. Til sammen ble over 6597 dokumenter vurdert og 100 søketermer vurdert.

sÃ¸k	numdocs	num_vurderi	meangrade	ikke_relevant	delvis_releva	relevant	vurd_0	vurd_1	vurd_2	vet_ikke
gustav lorentzen	20	7	1.026195	1	14	5	30	60	32	18
JÃ¸rgen Brekke	17	7	1.847335294117647	0	0	17	7	4	107	0
knausgÃ¥rd	18	7	1.481466666666667	0	6	12	20	22	82	2
profesjon	20	7	1.2428499999999998	0	8	12	29	46	63	2
Sykepleie	20	7	1.4283350000000001	0	6	14	14	46	68	12
Ville vekster	20	7	1.47857	2	3	15	20	32	87	1
adhd	20	6	1.45167	0	6	14	11	40	62	3
Colette	20	6	0.97	5	6	9	44	30	42	3
Cyber	20	6	0.6666599999999998	0	18	2	62	36	22	0
dag solstad	20	6	0.910005	1	13	6	44	36	32	4
Demens	20	6	1.19166	0	13	7	21	49	43	0
Engelsk	20	6	1.2666600000000001	3	2	15	29	28	55	0
the secret	20	6	0.108335	18	1	1	112	1	6	0

Figur 4.1 Resultater fra relevansvurderingen

Her er et utsnitt med tall fra relevansvurderingen. Til venstre har vi søketermene. Kolonne nummer to er antall dokumenter som skal vurderes. Neste kolonne er antall assessorer, og tabellen er rangert fra høyest til lavest antall assessorer. Kolonne *meangrade* er en verdi på hvor relevant vurderingene ble vurdert på en skala fra 0-2. Det gir en lettfattelig verdi på summen av vurderingene, men sier ikke så mye om trefflistens kvalitet, fordi en god treffliste kan bestå av noen få veldig relevante treff og flere mindre relevante treff, så lenge de er riktig rangert.

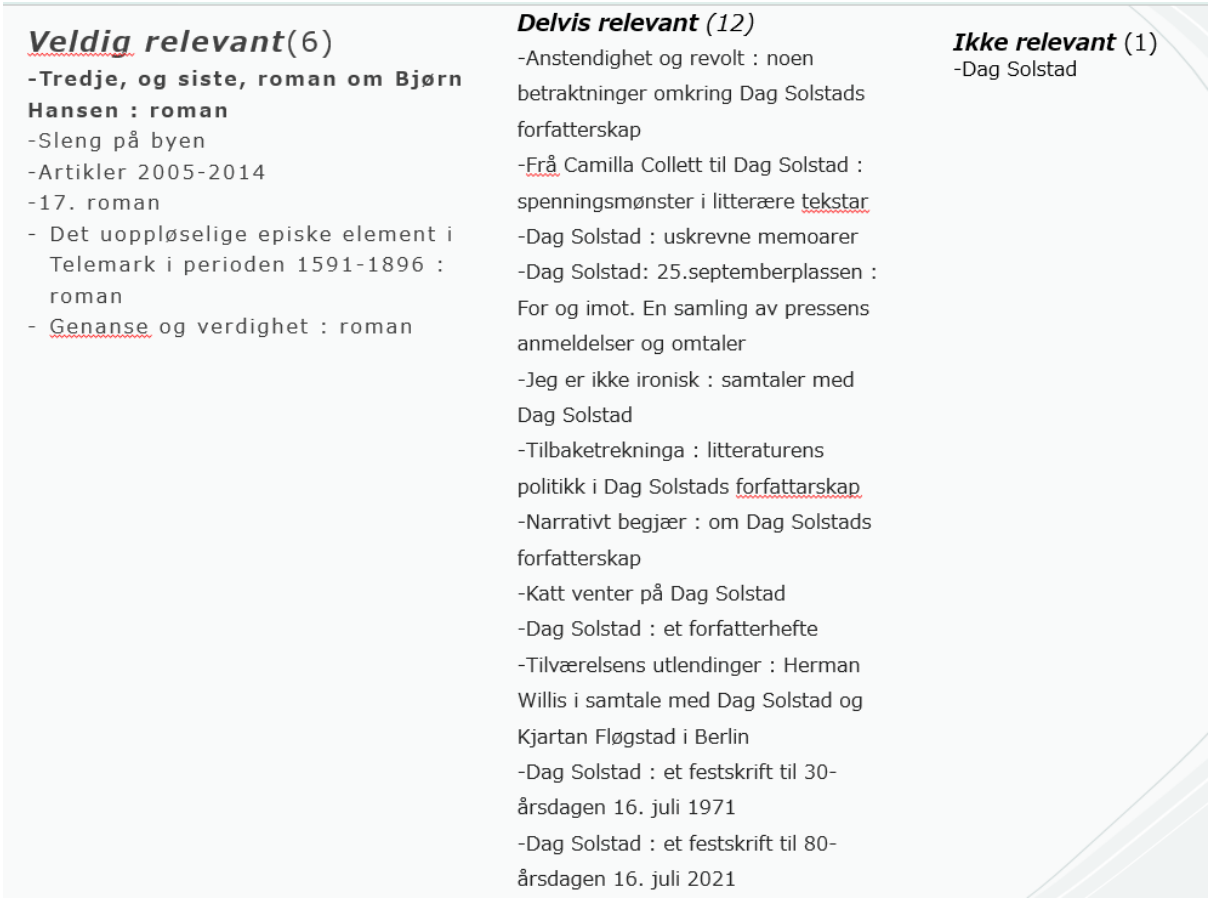
De neste kolonnene merket med gult, er den gjennomsnittlige vurderingen av hvert dokument. Vi kan gjerne kalle det klassifiseringer snarere enn vurderinger.

Enkeltvurderingene kommer til uttrykk i de blå kolonnene ved siden av. For søketermen *the secret* er gjennomsnittsvurderingen at 18 av 20 dokumenter er *ikke relevante*, 1 er *delvis relevant* og 1 er *veldig relevant*. Det kan tyde på at alle assessorene hadde det samme informasjonsbehovet og har kjennskap til boken *The Secret,* og anser den som veldig relevant. Resten vurderes stort sett som *ikke relevant*. Dette kan sies å være en enkel og entydig søketerm, altså det som kan kalles *known-item search* (Wildemuth & O'Neill, 1995). Andre søketermer har derimot gitt mer tvetydige resultater.

Resultatene viser en klar overvekt av relevante vurderinger. Det er ikke så rart siden trefflisten allerede er rangert basert på BM25. Det er ikke et problem så lenge vi har litt variasjon, men det er uheldig om variasjonen er veldig liten, og spesielt om alle dokumenter klassifiseres likt. For søketermen *Jørgen Brekke* er alle dokumenter klassifisert som veldig relevante. Det er uheldig fordi det ikke gir grunnlag for å diskriminere mellom dokumentene på trefflisten for søket *Jørgen brekke*. Det gjør det vanskeligere å *forbedre* rangeringen av disse dokumentene, men selv disse entydige vurderingene gir oss informasjon om hva som er relevant. Algoritmen lærer og mønstergjenkjenner de gode kvalitetene ved disse dokumentene og deres forbindelse til søketermen, som modellen til syvende og sist bruker til å rangere alle dokumenter. Resultatene av vurderingen av en søketerm påvirker altså rangeringer av dokumenter for andre søketermer. Derfor vil vurderingen av andre søketermer også påvirke hvordan dokumentene for søketermen *Jørgen Brekke* blir rangert, slik at denne i teorien også kan bli forbedret.

De blå cellene viser det totale antallet vurderinger i hver kategori. For "dag solstad" er det 44 vurderinger som er *ikke relevant*, 36 vurderinger som er *delvis relevant*, 32 vurderinger som er *veldig relevant* og 4 vurderinger som sier *vet ikke*. Da er det lett å tenke at mange dokumenter vil ha gjennomsnittsvurdering som "ikke relevant", men det gjelder faktisk kun ett dokument. Det kan forklares med at det er stor variasjon mellom dokumentene som vurderes som *ikke relevante*, foruten et dokument som de fleste er enige om. Det er mulig at noen har valgt et informasjonsbehov av å finne bøker *om* Dag Solstad, og dermed vurderer biografier *veldig relevant*, mens andre kan ha et informasjonsbehov om å finne skjønnlitterære bøker av Solstad. Da vil både biografien og romanene få motstridende relevansvurderinger, noe som gjør at

det gjennomsnittlig er færre dokumenter i yttergradene og flere dokumenter i midten som klassifiseres som *delvis relevant.* Etter å ha undersøkt hvilke dokumenter som utgjør de ulike gradene av relevans, har de fleste valgt et informasjonsbehov som favoriserer romaner. Kun romaner er klassifisert som *veldig relevant*, mens biografier og andre tekster er klassifisert som *delvis* og *ikke relevant*. De ikke-skjønnlitterære dokumentene har imidlertid fått nok *veldig relevante* og *delvis relevante* vurderinger til at de ikke gjennomsnittlig klassifiseres som *ikke relevant*, foruten det ene dokumentet. I Deichmanfilen over dokumenter som burde være høyere oppe på trefflista, fant vi Dag Solstads roman *Tredje, og siste, roman om Bjørn Hansen: roman*. Fra figur 4.1 kan man se at dette dokumentet er blant de 6 dokumentene som i gjennomsnitt har blitt vurdert som *veldig relevant*. Det er et positivt delresultat som vil gjøre modellen tilbøyelig til å favorisere egenskapene til dette dokumentet. Det er imidlertid bare 1 av tusenvis av vurderinger som påvirker modellen, så det er ingen automatikk i at dette resultatet er nok til å løfte dokumentet i sluttevalueringen av trefflista.



Figur 4.2 Resultatet av relevansvurderingene for søket “dag solstad”

I instruksen fikk assessorene beskjed om å skape seg et alminnelig informasjonsbehov for hver søketerm. Derfor la vi opp til å vurdere basert på *subjektiv relevans,* en relevans som ikke er statisk som sin *objektive* motpart. Angående de fem kategoriene til Saracevic: systembasert, subjektbasert, kognitiv, situasjonsbasert, motivasjonsbasert, kan flere av kategoriene passe. Hvis vi ser på resultatet fra figur 4.2, ser vi at dokumentet med tittel *Dag Solstad* er klassifisert som *ikke relevant* for søketermen "dag solstad". Dermed har ikke assessorene vurdert dette søketerm-dokumentparet ut fra systembasert relevans, fordi samsvaret mellom søketermen og tittelen er 100%, og likevel vurderes den som *ikke relevant*. Subjektbasert relevans handler om å se forbi søketermens bokstavelighet og vurdere samsvaret mellom innholdet i søketermen og dokumentet. Den samme tittelen "Dag Solstad" er et hefte på 22 sider skrevet av Hans Skei. Den *handler om* Dag Solstad, likevel er den vurdert *Ikke relevant*. Derfor kan man si at for dette eksempelet er heller ikke subjektiv relevans alene avgjørende for vurderingen. Kognitiv relevans tar brukerens kunnskap, perspektiv og informasjonsbehov med i beregningen. Siden dette heftet ikke er relevant mens noen av hans mest kjente romaner er *veldig relevant* (for eksempel *Genanse og Verdighet),* tyder det på at et informasjonsbehov er bestemmende for vurderingen; et behov for hovedsakelig romaner. Samtidig kan denne forskjellen i vurderingen mellom et lite hefte og en klassiker tyde på at assessoren tar med seg sin kunnskap og sitt bibliotekariske perspektiv (for alle er tilknyttet ABI) med i vurderingen og tenker at klassikeren kommer brukeren mest til gode. Derfor tyder resultatene på at assessorene vurderer basert på kognitiv relevans, noe de også ble instruert i å gjøre. Det betyr ikke at de andre typene av relevans er helt irrelevante for vurderingen, men at kognitiv relevans er den typen som stikker seg ekstra ut.

4.2 Resultatet av modellene

Vi begynner med å vise resultatet av modellene som er basert på alle featuregruppene separat. Deretter viser vi resultatet av alle featuregruppene samlet. Til slutt presenterer vi resultatet av de ulike kombinasjonene av featuregrupper.

4.2.1 Resultat av featuregrupper separat

Popularitetsfeatures

	LambdaMART		Train	Test
1	1	NDCG@10	0.7737	0.6990
2	2	NDCG@10	0.7813	0.6621
3	3	NDCG@10	0.7779	0.6823
4	4	NDCG@10	0.7715	0.6704
5	5	NDCG@10	0.7803	0.6358
6	AVG	NDCG@10	0.7769	0.6699

Eval 113 queries: improved=43, declined=32, unchanged=38, totaldiff=-152

Figur 4.3: Popularitetsfeatures med LambdaMART

	Random forest		Train	Test
1	1	NDCG@10	0.7428	0.6828
2	2	NDCG@10	0.7396	0.6313
3	3	NDCG@10	0.7356	0.7027
4	4	NDCG@10	0.7373	0.6999
5	5	NDCG@10	0.7537	0.6361
6	AVG	NDCG@10	0.7418	0.6706

Eval 113 queries: improved=33, declined=40, unchanged=39, totaldiff=-95

Figur 4.4: Popularitetsfeatures med Random forest

Vi har kryssvalidert dataene med fem deler, hvor man kan se resultatet av de respektive delene fra rad 1 til 5. Evalueringsmålet er nDCG, hvor verdien til dokumentet på plassering 10 (derfor @10) er målt. Verdien 1 er optimal score, mens 0 er dårligst score. På rad 6 ser du gjennomsnittsverdien av de fem delene. Kolonnen

lengst til høyre er prestasjonen på testsettet. Kolonnen ved siden av er prestasjonen på treningssettet. Den mest interessante nDCG-verdien er resultatet av gjennomsnittet av testsettet på linje 6. Linjen under, som begynner med Eval 113 queries, gir tall på hvordan det har gått med dokumentene fra Deichman-filen som vi ønsker å få lenger opp på trefflisten. Resultatet i figur 4.3 viser at 43 av dokumentene har flyttet seg lenger opp på trefflista, mens 32 har flyttet seg lenger ned. 38 har samme plassering på trefflista som før. Det er altså flere dokumenter som flytter seg lenger opp enn ned, og dermed kan vi si at det går riktig vei i forhold til det vi ønsker med problemstillingen. Dette tallet tar ikke i betraktning hvor mange plasseringer dokumentet har flyttet på seg. Det gjør imidlertid tallet *totaldiff*, som er en avregning av hvor mange plasseringer dokumentene har gått opp og ned. Til sammen har dokumentene flyttet seg 152 plasser oppover trefflista med en modell basert på popularitetsfeatures og LambdaMART**.**

I figur 4.4 testes popularitetsfeatures sammen med Random forest. Her ser vi at den gjennomsnittlige nDCG-scoren for testsettet er 0,6706, noe som er en liten forbedring fra LambdaMART. Derimot er det flere dokumenter som har gått nedover (40) på trefflisten enn opp (33). På tross av det har dokumentene totalt flyttet seg 95 plasser oppover trefflisten. Det kan forklares med at de dokumentene som flytter seg oppover, forbedrer plasseringen sin betydelig når de først gjør det, mens dokumentene som flytter seg nedover trefflisten forverrer plasseringen sin mer marginalt.

Kategoriske features

LambdaMART	Train	Test
1 NDCG@10	0.6709	0.6455
2 NDCG@10	0.6862	0.6292
3 NDCG@10	0.6805	0.5851
4 NDCG@10	0.6763	0.6248
5 NDCG@10	0.6784	0.5904
AVG NDCG@10	0.6784	0.6150
Eval 113 queries: improved=29, declined=43, unchanged=40, totaldiff=27		

Figur 4.5: Kategoriske features med LambdaMART

Random forest	Train	Test
1 NDCG@10	0.6575	0.6479
2 NDCG@10	0.6236	0.5211
3 NDCG@10	0.6475	0.6159
4 NDCG@10	0.6278	0.6698
5 NDCG@10	0.6357	0.5653
AVG NDCG@10	0.6384	0.6040
Eval 113 queries: improved=19, declined=30, unchanged=59, totaldiff=34		

Figur 4.6: Kategoriske features med Random forest

Figur 4.5 viser at resultatet for testsettet med LambdaMART er 0,6150. Det betyr at disse gir en dårligere nDCG-score enn popularitetsfeatures. 29 dokumenter forbedrer posisjonen sin, mens 43 dokumenter forverrer posisjonen sin på trefflista. Totalt faller dokumentene 27 plasser på trefflista.

Figur 4.6 viser at Random forest gir enda litt dårligere resultater enn LambdaMART, med en gjennomsnittlig nDCG-score på 0,6040. Den fører også til at det er flere dokumenter som forverrer seg enn dokumenter som forbedrer sin posisjon. Det viser at gruppen kategoriske features ikke bidrar til å løfte dokumentene oppover trefflisten.

Tekst-matching-features

	LambdaMART	Train	Test
0	1 NDCG@10	0.8063	0.6625
1	2 NDCG@10	0.8126	0.6167
2	3 NDCG@10	0.8089	0.6636
3	4 NDCG@10	0.7970	0.6648
4	5 NDCG@10	0.8218	0.6565
5	AVG NDCG@10	0.8093	0.6528
improved=33, declined=25, unchanged=55, totaldiff=-123			

Figur 4.6: Tekstfeatures med LambdaMART

	Random forest	Train	Test
0	1 NDCG@10	0.7662	0.7185
1	2 NDCG@10	0.7927	0.6226
2	3 NDCG@10	0.7671	0.6410
3	4 NDCG@10	0.7538	0.6757
4	5 NDCG@10	0.7806	0.6219
5	AVG NDCG@10	0.7721	0.6559

improved=27, declined=23, unchanged=62, totaldiff=-48

Figur 4.7: Tekstfeatures med Random forest

Med tanke på nDCG-scoren, så presterer Random forest (0,6559) marginalt bedre enn LambdaMART (0,6528). Det er imidlertid stor forskjell mellom totaldiff, hvor Random forrest har en verdi på -48, mens LambdaMART har -123. LambdaMART presterer dermed best i forhold til vår hovedproblemstilling. Den tekstlige featuregruppen presterer nesten like godt som popularitetsfeatures, og betraktelig bedre enn kategoriske features.

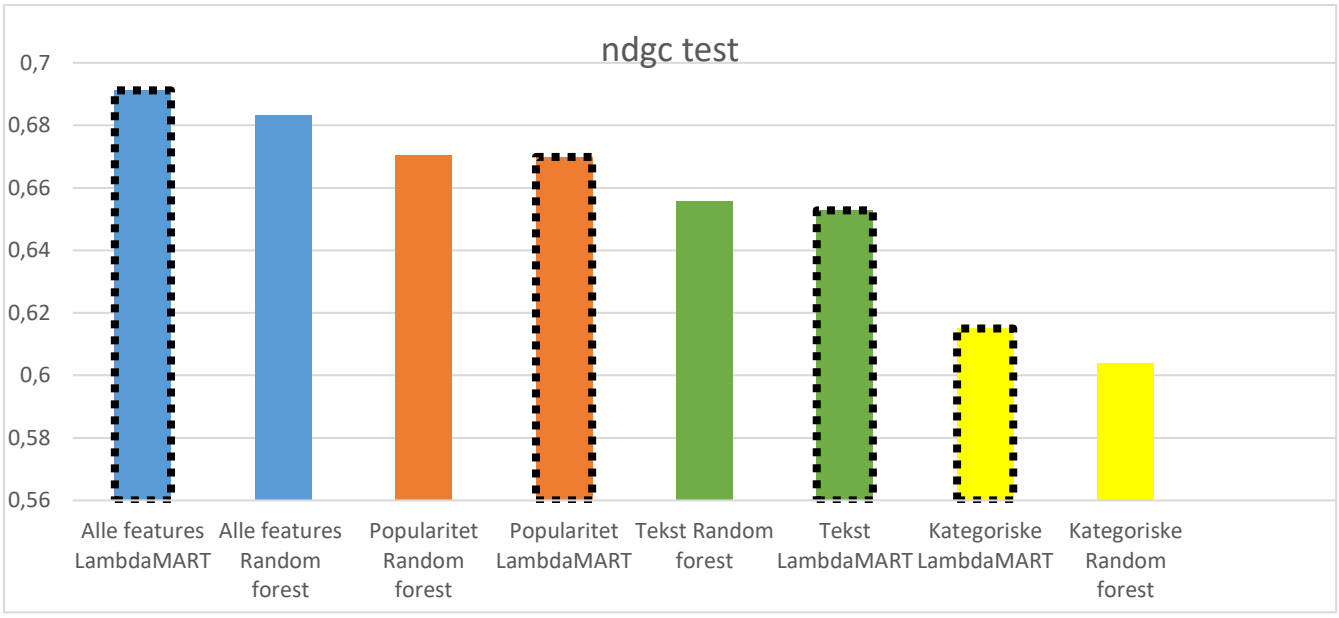
4.2.2 Resultat av alle featuregrupper samlet

LambdaMART	Train	Test
1 NDCG@10	0.8316	0.7180
2 NDCG@10	0.8301	0.7288
3 NDCG@10	0.8326	0.6824
4 NDCG@10	0.8388	0.6872
5 NDCG@10	0.8379	0.6396
AVG NDCG@10	0.8342	0.6912
Eval 113 queries:	improved=48, declined=31, unchanged=34, totaldiff=-164	

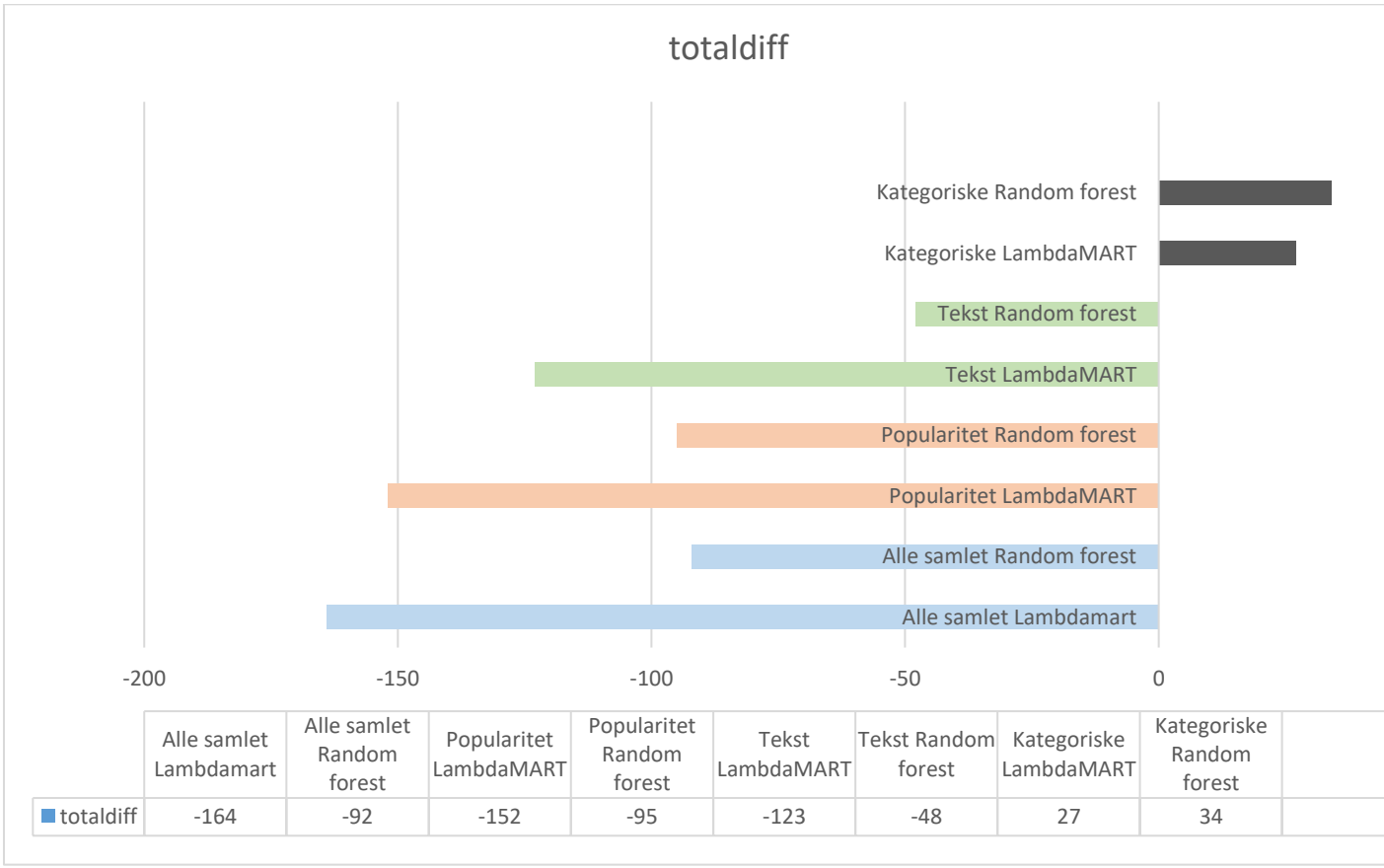
Figur 4.8: Alle featuregrupper samlet med LambdaMART

Random Forest	Train	Test
1 NDCG@10	0.8024	0.7282
2 NDCG@10	0.7911	0.6319
3 NDCG@10	0.7606	0.7257
4 NDCG@10	0.7784	0.6738
5 NDCG@10	0.8227	0.6571
AVG NDCG@10	0.7910	0.6833
Eval 113 queries:	improved=40, declined=29, unchanged=43, totaldiff=-92	

Figur 4.9: Alle featuregrupper samlet med Random forest



Figur 4.10: Oversikt over nDCG-score for alle featuregrupper og modeller, samlet og separat



	Alle samlet Lambdamart	Alle samlet Random forest	Popularitet LambdaMART	Popularitet Random forest	Tekst LambdaMART	Tekst Random forest	Kategoriske LambdaMART	Kategoriske Random forest
totaldiff	-164	-92	-152	-95	-123	-48	27	34

Figur 4.11: Oversikt over totaldiff-score for alle featuregrupper og modeller, samlet og separat

Både nDCG-scoren fra figur 4.10 og totaldiff-scoren fra figur 4.11 viser at alle features samlet, sammen med LambdaMART, gir best resultater. Det gir bedre resultater enn noen av featuregruppene alene. Begge figurene viser at popularitetsfeatures er den enkeltgruppen som presterer best. Ut fra totaldiff presterer popularitetsfeatures best sammen med LambdaMART, mens nDCG viser at Random forest fungerer bedre på popularitetsfeatures.

4.2.3 Resultater av ulike featuregrupper kombinert

Tekst-matching-features og kategoriske features

LambdaMART	Train	Test
1 NDCG@10	0.8017	0.6712
2 NDCG@10	0.8096	0.6617
3 NDCG@10	0.8153	0.6270
4 NDCG@10	0.8076	0.6605
5 NDCG@10	0.8176	0.6551
AVG NDCG@10	0.8104	0.6551
Eval 113 queries:	improved=29, declined=30, unchanged=53, totaldiff=-121	

Figur 4.12: Tekst-matching-features og kategoriske features med LambdaMART

Random forest	Train	Test
1 NDCG@10	0.7480	0.6467
2 NDCG@10	0.7492	0.6583
3 NDCG@10	0.7693	0.7146
4 NDCG@10	0.7715	0.6531
5 NDCG@10	0.7744	0.6004
AVG NDCG@10	0.7625	0.6546
improved=28, declined=30, unchanged=53, totaldiff=-56		

Figur 4.13: Tekst-matching-features og kategoriske features med Random forest

Tekst-matching-features med popularitetsfeatures

LambdaMART	Train	Test
1 NDCG@10	0.8149	0.7005
2 NDCG@10	0.8279	0.6708
3 NDCG@10	0.8168	0.6981
4 NDCG@10	0.8157	0.6894
5 NDCG@10	0.8260	0.6285
AVG NDCG@10	0.8203	0.6775
Eval 113 queries: improved=51, declined=30, unchanged=32, totaldiff=-203		

Figur 4.14: Tekst-matching-features og popularitetsfeatures med LambdaMART

Random forest	Train	Test
1 NDCG@10	0.7939	0.7291
2 NDCG@10	0.7949	0.6356
3 NDCG@10	0.7982	0.6710
4 NDCG@10	0.7915	0.6850
5 NDCG@10	0.7763	0.6143
AVG NDCG@10	0.7910	0.6670
Eval 113 queries: improved=44, declined=24, unchanged=44, totaldiff=-126		

Figur 4.15: Tekst-Matching-features og popularitetsfeatures med Random forest

Figur 4.14 viser at tekst-matching-features kombinert med popularitetsfeatures, sammen med LambdaMART, gir best resultat.

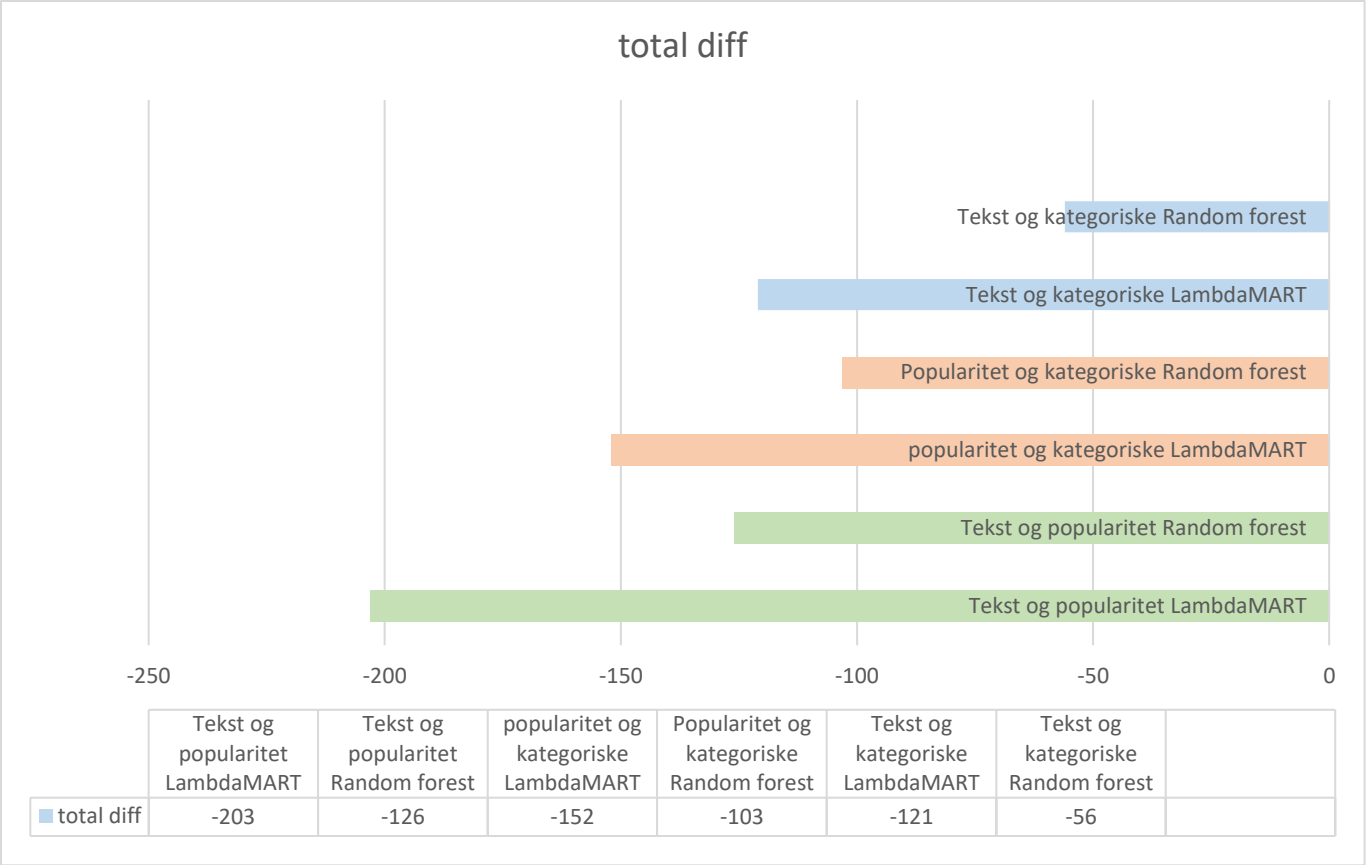
Kategoriske features og popularitetsfeatures

LambdaMART	Train	Test
1 NDCG@10	0.7901	0.7057
2 NDCG@10	0.8058	0.6858
3 NDCG@10	0.7972	0.6681
4 NDCG@10	0.8058	0.6743
5 NDCG@10	0.7929	0.6594
AVG NDCG@10	0.7983	0.6786
Eval 113 queries:	improved=53, declined=31, unchanged=29, totaldiff=-152	

Figur: 4.16: Kategoriske features og popularitetsfeatures med LambdaMART

Random forest	Train	Test
1 NDCG@10	0.7723	0.6687
2 NDCG@10	0.7554	0.6160
3 NDCG@10	0.7431	0.6751
4 NDCG@10	0.7761	0.6785
5 NDCG@10	0.7774	0.6898
AVG NDCG@10	0.7649	0.6656
Eval 113 queries:	improved=46, declined=25, unchanged=41, totaldiff= -103	

Figur 4.17: Kategoriske features og popularitetsfeatures med Random forest



Figur 4.18: Oversikt over alle kombinasjoner av featuresgrupper

Figur 4.18 viser at kombinasjonen tekst-matching-features og popularitetsfeatures med LambdaMART er den beste kombinasjonen. Den forbedrer rangeringen med 203 plasser, mens kombinasjon nummer to er popularitet og kategoriske features med LambdaMART som forbedrer rangeringen med 152 plasser. Kombinasjonen av tekst-matching-features og popularitetsfeatures gir den modellen som presterer best av alle modellene. Den slår også alle grupper samlet, som ga en forbedring på 164, som vist i figur 4.11.

5 Diskusjon

5.1 Assessorene

For at rangeringsmodellen i denne oppgaven best mulig skal tjene bibliotekbrukerne, er det en fordel at datagrunnlaget best mulig representerer dem som bruker den. Timnit Gebru (2020) skriver at en av KI's store fallgruver er at den har en tendens til å produsere innhold som reflekterer hvite velstående menn fra Europa og USA, mens marginaliserte grupper ikke er godt nok representert. I vårt design har valg av assessorer mye å si for hvilket innhold som blir favorisert, i likhet med valget av dokumenter og features den trenes på. Formålet med oppgaven er å forbedre trefflisten for alle Deichmanbrukere. Ideelt sett burde utvalget av assessorer være et variert utvalg av Deichmanbrukere, altså et representativt utvalg av brukere av søketjenesten til katalogen. Det ble imidlertid vanskelig å få til av praktiske årsaker og innenfor rammene av masteroppgaven, derfor gikk vi for assessorer tilknyttet instituttet for arkiv, bibliotek og informasjonsvitenskap (ABI). Dette er en relativt homogen gruppe som ikke reflekterer mangfoldet av Deichmanbrukere. På en annen side kan man argumentere for at disse interne assessorene har kunnskap om bibliotek og søk og gjenfinning, noe som gjennomsnittsbrukeren ikke har. Likevel vil det være en fordel - og man kan argumentere for at det er mer etisk og demokratisk riktig - om data fra en representativ Deichman-bruker brukes som treningsdata. Dette krever imidlertid en del ressurser og et godt samarbeid med Deichman. Et alternativ er å gå vekk fra å skaffe menneskeskapte relevansvurderinger og heller bruke systemgenererte relevansvurderinger i form av utlånstall, klikk og lignende. På denne måten er det enklere å skaffe store mengder relevansvurderinger som i større grad representerer hele samlingen og alle brukerne. Dette er en anbefalt fremgangsmåte om man ikke har store ressurser og et godt samarbeid med Deichman til å gjøre et eksperiment med menneskeskapte relevansvurderinger (*human in the loop)* i stor skala.

Vi bør altså strebe etter å oppnå best mulig representativitet, men det er ikke en forutsetning for at resultatet skal lykkes med sine mål. Selv om ChatGPT ikke representerer hele verdens mangfold godt nok, så vil mange si at ChatGPT er en imponerende tjeneste som kan ha en stor nyttefunksjon. På samme måte er det

mulig at en rangeringsmodell, slik som vi har laget i denne oppgaven, kan forbedre søket i folkebibliotek selv om den ikke er basert på relevansvurderinger fra et representativt utvalg bibliotekbrukere. Dagens rangeringsmodell hos Deichman, som er basert på BM25, rangerer etter systembasert relevans og representerer ingen bibliotekbrukere; brukerne har ingen innvirkning på rangeringen. Dagens rangeringsmodell er ikke perfekt, så en LTR-basert rangeringsmodell, slik som vi produserer i denne oppgaven, kan likevel forbedre dagens modell til tross for sine svakheter.

5.2 Mengden treningsdata

Vi skulle gjerne hatt enda flere vurderinger, og særlig enda flere søketermer, for å bedre representere mangfoldet av søketermer og dokumenter i folkebibliotek i treningen av den nye rangeringsmodellen. 1896 dokumenter ble vurdert av i gjennomsnitt 5 personer per dokument. Det gir betydelige data som har gitt resultater, men resultatet ville vært enda bedre, og mer representativt for samlingen, om vi hadde hatt enda mer data. Modellene våre har ikke et treningsgrunnlag som dekker de over 1 million dokumentene ved Deichman, de 39 ulike formatene, dokumenter på 149 ulike språk, eller de 116 ulike sjangrene. For å gi et mer konkret eksempel: la oss si at assessorene ikke har vurdert et eneste dokument i formatet CD, og de har heller ikke vurdert et eneste dokument på språket persisk. Hvis det da finnes en CD av en persiskspråklig artist i samlingen og den nye modellen skal rangere den, da vil modellen ha dårlige forutsetninger for å rangere dette dokumentet godt. Dette har seg fordi modellen ikke har trening i å finne et mønster mellom egenskapene persisk og CD og grader av relevans. Den vil likevel rangere dokumentet basert på tekst-matching, popularitetsfeatures, og andre mulige features, men ikke basert på de andre essensielle egenskapene til dokumentet.

5.3 Popularitetsfeatures – en populistisk modell?

Resultatene viser at popularitetsfeatures er featuregruppen som presterer best. Det er ikke så rart fordi noe av det som kjennetegner dokumentene fra Deichmanfilen (som vi forsøker å flytte høyere opp) er at de er populære, og evalueringsmetoden vår (totaldiff) er å måle forflytningen av nettopp disse *populære* bøkene. Det andre evalueringsmålet vårt nDCG, måler derimot alle dokumenter på trefflisten, ikke bare

de utvalgte populære dokumentene, og da er det også popularitetsfeatures som presterer best. Dette er ikke like naturlig. Det tyder på at assessorene har gitt gode relevansvurderinger av dokumenter som scorer høyt på popularitetsfeatures, slik at algoritmen har blitt trent til å favorisere populære dokumenter.

I innledningen snakket vi om at Deichman har et ansvar for å formidle smal litteratur, det vil si litteratur som ikke er populær, eller som har spesielle formmessige eller innholdsmessige egenskaper. En slik modell som favoriserer populære dokumenter, vil samsvare dårlig med dette formålet og ha et populistisk bias. Det kan gi gode resultater, ref. wisdom of the crowds (Surowiecki, 2005), men det er viktig å reflektere etisk over om det er ønskelig at det går på bekostning av marginaliserte brukere og *smale* deler av samlingen. En løsning for å bøte på dette er å inkludere en rekke andre features som ikke angår popularitet, for eksempel kategoriske features. Om ikke det hjelper, er det også mulig å overstyre den naturlige vektingen som fremkommer av treningen, ved å manuelt vekte features som man tenker er spesielt viktige for resultatet. En annen løsning er å endre designet av hvordan man skaffer relevansvurderinger, slik at assessorene, eller oppsettet, ikke premierer popularitet i like stor grad. Vi kan eventuelt ta bort enkeltfeatures som er de mest *populistiske*. *Antall eksemplarer* er en mer balansert popularitetsfeature fordi man kan argumentere for at innkjøpshistorikken gjenspeiler bibliotekets og bibliotekarenes smak. Det er hva Behnert kaller *popularity based on authority*. *Antall utlån* og *reserveringer*, derimot, er mer rene popularitetsfeatures som kan vurderes å tas bort eller dempes. *Antall utlån* vil også akkumulere seg over tid, noe som gir en fordel til eldre verker. Det vil i tillegg føre til at modellen vil være tilbøyelig til å favorisere eldre klassikere med høyt utlån. Vi har brukt log10 av antall utlån+1 for å dempe dette biaset, men vi kunne gjort enda mer for å redusere popularitetsbias om det var ønskelig. Deichman sendte oss en liste med populære dokumenter de vil ha høyere opp på trefflisten, så det tyder også på at de tenker det er en god ide å i større grad rangere trefflisten basert på popularitet.

5.4 Hvorfor fungerte ikke kategoriske features?

Ideelt sett skulle vi testet og analysert denne kategorien ytterligere, fordi den består av flere ulike typer kategoriske features, og det kan hende at noen av dem er nyttige,

mens andre ikke er det. Det vet vi ikke basert på resultatene. Vi kunne delt opp gruppen i undergruppene *nyhetsverdi*, *målgruppe* og *litteraturtyper*, og testet og evaluert resultatet av hver undergruppe. Det ville bragt oss nærmere i diagnostiseringen. Nå kan vi bare spekulere i om måten vi behandlet disse dataene på, via feature engineering, ikke var ideell; for eksempel hvordan vi representerte de kategoriske dataene via dummyvariabler 0 og 1. Det kan også hende at vi ikke hadde nok treningsdata av kategoriene. Vi har ikke fått anledning til å undersøke dette nærmere innenfor oppgavens ramme.

5.5 Evaluering

Vi har to evalueringsmål. nDCG@10 og totaldiff. Førstnevnte evaluerer trefflista til testsettet for hvert søk i forhold til relevansvurderingene. Sistnevnte evaluerer kun om vi har lykkes i å løfte dokumentene fra Deichman-filen. Det kan argumenteres for at nDCG svarer bedre på spørsmålet om trefflisten har forbedret seg. Problemet med å evaluere testsettet er at det bare utgjør en femtedel av treningsdataene. Og som vi har diskutert tidligere, er heller ikke treningsdataene våre helt representative for Deichmansamlingen. I tillegg består nDCG av relevansvurderinger assessorer som ikke er helt representative for Deichmanbrukerne. Dette gjør det vanskelig å konkludere med at en høyere nDCG-score er bevis for at modellen på samme måte vil forbedre katalogsøket hos Deichman.

Med verdien for totaldiff evaluerer vi modellen basert på om vi får løftet treffene fra Deichman-filen. Denne Deichman-filen er en maskingenerert liste over populære dokumenter (de har fått en del klikk) som befinner seg lengre nede på lista. De fleste av eksemplene gir intuitivt mening å løfte, slik som boken *Harry Potter og de vises stein* som er på 19. plass for søket *Harry Potter*. Det er imidlertid noen eksempler som ikke er like klare. Man kan for eksempel finne boken *Golf for begynnere* av Peter Ballingall fra 1993 for søketermen *Golf*. Denne er ganske gammel og ser noe utdatert ut. Basert på ren intuisjon kan det være vanskelig å se at trefflisten blir noe særlig forbedret av at den får en høyere plassering. Ved å løfte den oppnår vi å løse oppgaven til Deichman, men slike eksempler gir grunn til å være noe skeptisk til at trefflistene vil forbedre seg hver gang disse dokumentene løftes. Andre evalueringsmetoder må imidlertid til for å svare sikkert på om den nye modellen

forbedrer trefflisten til Deichman. Men vi kan konkludere med at alle modellene som gir negativ totaldiff-score, vil løfte de populære dokumentene fra Deichman-filen. Og det *kan* også hende det ville medført en forbedring av hele samlingen, særlig for populære dokumenter, om den ble implementert, men det kan vi ikke konkludere helt sikkert over basert på resultatene.

Oppgaven foretar ikke en grundigere analyse av mulig bias i resultatet. Med bias i denne sammenhengen menes det hva som favoriseres av algoritmen. Vi vet at populære bøker favoriseres, men vi vet ikke så mye utover det. Det hadde vært mulig å skrive kode hvor vi fant ut hvor mange plasseringer dokumenter i en viss kategori i gjennomsnitt har steget eller falt på trefflista. Vi kunne for eksempel sett om sjangeren *krim* i snitt har klatret mer enn *fantasy* bøker. I så fall har modellen et *krimbias*. Vi kunne sett om dokumenter som var nye klatret på trefflista, og andre lignende tilfeller.

Vi foretok ikke en analyse av svarene fra relevansvurderingene via inter-rater reliability og Cohens kappa. Det kunne vært nyttig med tanke på å evaluere relevansvurderingene grundigere, noe som ville gitt konkrete tall på hvor ulikt assessorene vurderte. Det ville kunne bedre avdekke om valget av subjektive relevansvurderinger (hvor assessorene fikk vurdere etter et selvvalgt alminnelig informasjonsbehov) var et godt valg, og om vurderingene representerte mangfoldige informasjonsbehov. Vi produserte riktignok tall over relevansvurderingene og studerte dem, som ble presentert i figur 4.2 og 4.3, og dette gjør at vi får noe av den samme innsikten som vi ville fått ved Cohens kappa. Tallene fra figurene viste at det var klare tendenser, men med en viss variasjon, og det var vi fornøyde med.

5.6 Informasjonsbehov og relevans

Som presentert i resultatdelen for relevansvurderinger, var det for søketermen «dag solstad» en klar tendens til å velge et informasjonsbehov som favoriserte romaner. Hvis dette er en tendens for alle vurderingene av alle søk, så vil modellen favorisere romaner fremfor ikke-romaner. Hvis det er tilfelle, er det uheldig når brukerne er ute etter faglitteratur. Vi vil ha en modell som fungerer på begge disse kategoriene. Tanken var at vi hadde nok assessorer per søketerm, opptil 7 personer - i

gjennomsnitt 5, som gjorde at det ble vurdert etter ulike informasjonsbehov, som for eksempel både behov 1: romaner, behov 2: biografier, slik at modellen ikke fikk et stort bias overfor det ene behovet. Da er spørsmålet om en annen måte å vurdere relevans på ville gitt et bedre resultat. For eksempel om vi gikk for subjektbasert eller systembasert relevans, hvor assessorene ikke hadde egne informasjonsbehov, men vurderte mer objektivt og så etter samsvar mellom søketerm og dokument, uten å trekke inn informasjonsbehov eller egen bibliotekarisk ekspertise om hva som er gode utlånsobjekt for brukerne. Det ville vært interessant å sammenligne resultatet av våre relevansvurderinger med vurderingene fra et slikt mer *objektivt* design. Det kan tenkes at fraværet av informasjonsbehov kunne ført til jevnere vurderinger, hvor hele trefflisten er ganske relevant, fordi trefflisten de vurderer er allerede rangert via BM25 som er basert på objektiv/systembasert relevans. Dette kunne medvirket til mindre varians (at modellen fungerer godt på nye data), men samtidig produserer en del bias og er mindre presis. Subjektiv relevans legger derimot i større grad til rette for at dokumenter på den rangerte trefflista vurderes som *ikke relevant* når et dokument ikke passer med ditt informasjonsbehov, men likevel har god *systembasert relevans*. Dette gir større variasjon i vurderingene, og derfor mer rangeringsinformasjon. Dette gir et bedre grunnlag for å flytte dokumenter oppover og nedover trefflista. Dette er et argument for å benytte subjektive vurderinger via egenskapte informasjonsbehov.

5.7 Vil resultatene for denne oppgaven være overførbare til andre norske folkebibliotek?

Treningsdataene våre er basert på dokumenter fra Deichman, og det er landets desidert største bibliotek. Det er ingen folkebibliotek i Norge som har en større samling enn Deichman. Treningsdataene vil være preget av det, og vil ikke være helt representative for resten av landets folkebibliotek. Likevel er bøkene i større eller mindre grad representert i alle bibliotek i landet, og Deichman har i stor grad alle bøkene som finnes i de forskjellige folkebibliotekene rundt om i landet, foruten lokalsamlinger og spesialsamlinger. I tillegg er utvalget vårt fra Deichmanfilen populære bøker, og populære bøker har en tendens til å være godt representert i de fleste folkebibliotek i Norge. Derfor vil modellen i teorien kunne generalisere godt basert på det. Modellen vil derimot ikke generalisere godt på samlinger som skiller

seg stort fra Deichmans samling. De som har relevansvurdert, er Osloborgere, og mange er brukere av Deichman. Hva som anses for å være relevant av en Osloborger er ikke nødvendigvis relevant for en person som bor et annet sted. Utopien er at hver biblioteksamling har en modell som er tilpasset deres samling og brukere, men dette er dessverre svært vanskelig å få til i praksis.

Det ville vært interessant å se om en modell som er basert på Deichman-data og Deichmanbrukere ville forbedret katalogsøket i alle bibliotekene i hele landet. Sannsynligvis ville den fungert bedre på folkebibliotekene i andre norske storbyer enn dem på bygdene. Kanskje ville den uansett vært en forbedring sammenlignet med dagens algoritme, selv for de små bibliotekene. Resultatene viser at metoden LTR er lovende, derfor hadde det vært interessant å utføre et lignende eksperiment basert på data fra alle folkebibliotek i landet, og basert på alle bibliotekbrukere i landet. Det ville kunne gi en algoritme som hadde fungert for alle bibliotekene. Denne modellen hadde imidlertid måtte bli noe generell, altså med høy bias men lav varians, for å generalisere godt på alle bibliotek. De største bibliotekene, som Deichman, som har ressurser til det, ville nok vært best tjent med å lage en egen modell som var bedre tilpasset sin egen samling og sine egne brukere.

5.8 Andre tjenester basert på maskinlæring og KI i norske folkebibliotek

I denne oppgaven har vi fokusert på å produsere en ny rangeringsmodell, men mulighetene er mange for å lage andre tjenester basert på maskinlæring og KI i folkebibliotek. Avslutningsvis vil vi komme med noen eksempler. Vi henviser til noe forskning, men tillater oss også å kaste ut noen ideer over muligheter som i større og mindre grad er realiserbare.

Det er mulig å bruke maskinlæring til å lage en modell som bestemmer hvilke dokumenter som bør kasseres og kjøpes inn (Rhanoui et al., 2022; Wagstaff & Liu, 2018). I innledningen ble det nevnt at det finnes forskning om hvordan man kan lage modeller som forutser fremtidig siteringsfrekvens (Abrishami & Aliakbary, 2019). Trolig kan dette konseptet brukes på utlånstall i folkebibliotek, hvor man produserer en modell som kan forutse hvor populært ett nytt dokument blir. Eldre kilder har en fordel når det kommer til utlånstall fordi tallet akkumuleres; dette problemet ved nye

dokumenter kalles *cold start problem*. Det å forutse fremtidig utlånstall kan være en løsning. Da kunne man i teorien, via maskinlæring, gitt hvert dokument et tall på antatt fremtidig utlånsfrekvens. Denne verdien kunne igjen blitt brukt som en feature i designet av en ny rangeringsmodell. Man kan også se for seg å bruke maskinlæring til å katalogisere dokumenter, hvor man trener en algoritme opp på en stor mengde korrekte katalogiserte dokumenter. Det er viktig å være presis med katalogisering, og maskinlæringen vil få en utfordring med å være helt presis, likevel kan det tenkes at en grovkatalogisering kunne hatt en nyttefunksjon. Det å predikere emneord, sjangre og dokumenttyper burde være realiserbart. Det er konseptuelt sammenlignbart med de KI-genererte emneordene i fagbibliotek som nevnt i innledningen (*ScienceDirect Topics*, 2023). Forskjellen, og utfordringen, i folkebiblioteksammenheng er at folkebibliotek-katalogene i Norge ikke tillater fulltekstsøk. Om fulltekstsøk var tilfelle, ville det gitt mye mer tekstlig data som ville gjort det enklere for algoritmene å predikere sjangre og emner av dokumenter i folkebibliotek. Fulltekstsøk vil åpne dørene for en rekke nye muligheter. Det krever økt tilgang på digitalt materiale som folkebiblioteket i dag ikke har. Nasjonalbiblioteket og bokhylla har imidlertid en stor digital samling hvor man kunne eksperimentert med nye maskinlærte tjenester. Trolig kan det åpne for semantiske søk hvor det er mulig å utforme friere søkeformuleringer som for eksempel «gi meg fem romaner med en mannlig, rik og amerikansk hovedkarakter, maks 200 sider». ChatGPT håndterer slike søk og gir deg fem romaner som mer eller mindre passer beskrivelsen. Resultatene varierer riktignok i kvalitet, hvor de kan være upresise og i tillegg er i stand til å finne opp falske dokumenter som ikke eksisterer. En ide relatert til semantisk søk er å bruke maskinlæring til å lære modeller å kategorisere ulike typer søk. For eksempel kan søkene kategoriseres som known-item-search eller exploratory search. Hvilken kategori søket kategoriseres som, kan ha konsekvenser for hvordan dokumentene i trefflisten blir rangert.

Dette er bare noen eksempler på hvordan maskinlæring kan brukes i folkebiblioteksammenheng. Den raske utviklingen av ChatGPT viser imidlertid at søketjenester forbedrer seg i et høyt tempo, og det er ikke lenger utenkelig at lignende teknologi vil kunne brukes i folkebiblioteket om noen år. Det blir spennende å se hvordan folkebiblioteket utvikler seg i fremtiden og om det benytter seg av maskinlæring for å tilby tjenester som kan komme brukerne, bibliotekarene og

samfunnet til nytte. Hvis man skal spå basert på hvordan webtjenester og tjenester knyttet til fagbibliotek utvikler seg basert på maskinlæring, er det grunn til å tro at maskinlæring og KI også vil påvirke folkebibliotekets tjenester i overskuelig fremtid.

6 Konklusjon

Forskningsspørsmål:
Hvilke typer data, omsatt til features, kan bidra til å forbedre rangeringen av dokumenter i trefflista i folkebibliotek, og hvordan bør disse kombineres?

Learning to Rank-eksperimentet i denne oppgaven ga resultater som løftet dokumentene fra Deichmanfilen lenger opp på trefflista. En modell som kombinerte popularitetsfeatures og tekst-matching-features, sammen med algoritmen LambdaMART, ga best resultater. Den modellen fikk forbedret rangeringen av dokumentene med til sammen 203 plasseringer. Modellen som presterte nest best, var alle featuregruppene samlet, med algoritmen LambdaMART. Den fikk forbedret dokumentene med 164 plasseringer. Når vi testet enkeltgruppene av features separat, ga popularitetsfeatures sammen med LambdaMART best resultat med en forbedring av 152 plasseringer. Tekst-matching-features presterte nest best med en forbedring av 123 plasseringer. Kategoriske features var den gruppen av features som presterte dårligst. Den bidro til å forverre trefflisten med 34 plasseringer med algoritmen Random forest, som var det dårligste resultatet fra alle testene. I tillegg kan vi konkludere med at algoritmen LambdaMART presterte bedre enn Random forest.

Problemstilling:
Hvorvidt kan maskinlæringsmetoden Learning to Rank (LTR) brukes til å forbedre rangeringen av dokumenter i trefflista i folkebibliotek?

Resultatet fra dette eksperimentet har vist at metoden LTR kan brukes til å forbedre rangeringen av utvalgte dokumenter i folkebibliotek. Som diskutert i diskusjonsdelen, må vi ta forbehold om at modellene som ble produsert i denne oppgaven ikke nødvendigvis vil forbedre trefflista i folkebibliotek om de faktisk skulle blitt implementert, hovedsakelig på grunn av for lite treningsdata. Likevel viser de positive resultatene fra eksperimentet at metoden LTR er lovende med tanke på å forbedre trefflista i folkebibliotek, særlig ved bruk av popularitetsfeatures kombinert med tekst-matching-features og algoritmen LambdaMART. Denne oppgaven er et forskningsbidrag om LTR og folkebibliotek, et område det er forsket lite på.

Oppgaven gir informasjon og resultater som ny forskning kan bygge videre på frem mot å produsere en maskinlært rangeringsmodell som vil forbedre trefflista i folkebibliotek.

Litteraturliste

Abrishami, A., & Aliakbary, S. (2019). Predicting citation counts based on deep neural network learning techniques. *Journal of Informetrics*, *13*(2), 485–499. https://doi.org/10.1016/j.joi.2019.02.011

Agichtein, E., Brill, E., Dumais, S., & Ragno, R. (2006). Learning user interaction models for predicting web search result preferences. *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '06*, 3. https://doi.org/10.1145/1148170.1148175

Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern information retrieval: The concepts and technology behind search* (Second edition). Addison Wesley.

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An Overview*. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

Borlund, P. (2003). The concept of relevance in IR. *Journal of the American Society for Information Science and Technology*, *54*(10), 913–925. https://doi.org/10.1002/asi.10286

Duboue, P. (2020). *The art of feature engineering: Essentials for machine learning* (First edition). Cambridge University Press.

Gebru, T. (2020). Race and Gender. I M. D. Dubber, F. Pasquale, & S. Das (Red.), *The Oxford Handbook of Ethics of AI* (s. 0). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780190067397.013.16

Gisev, N., Bell, J. S., & Chen, T. F. (2013). Interrater agreement and interrater reliability: Key concepts, approaches, and applications. *Research in Social and Administrative Pharmacy*, *9*(3), 330–338. https://doi.org/10.1016/j.sapharm.2012.04.004

Hellum, K. A. (2017). *Information Retrieval using applied Supervised Learning for Personalized E-Commerce* [Masteroppgave]. University of Stavanger.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R*. Springer US. https://doi.org/10.1007/978-1-0716-1418-1

Jacoby, J., & Matell, M. S. (1971). Three-Point Likert Scales Are Good Enough. *Journal of Marketing Research*, *8*(4), 495. https://doi.org/10.2307/3150242

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. I C. Nédellec & C. Rouveirol (Red.), *Machine Learning: ECML-98* (s. 137–142). Springer. https://doi.org/10.1007/BFb0026683

Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., & Gay, G. (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. *ACM Transactions on Information Systems*, *25*(2), 7. https://doi.org/10.1145/1229179.1229181

Kulturdepartementet. (2019). *Nasjonal bibliotekstrategi 2020-2023 - Rom for demokrati og dannselse* Nasjonal bibliotekstrategi 2020-2023 – Rom for demokrati og dannelse. Kulturdepartementet. https://www.regjeringen.no/no/dokumenter/nasjonal-bibliotekstrategi-20202023---rom-for-demokrati-og-dannelse/id2667015/

Lewandowski, D. (2009). Ranking library materials. *Library Hi Tech*, *27*(4), 584–593. https://doi.org/10.1108/07378830911007682

Lewandowski, D. (2010). Using Search Engine Technology to Improve Library Catalogs. I A. Woodsworth (Red.), *Advances in Librarianship* (Bd. 32, s. 35–54). Emerald Group Publishing Limited. https://doi.org/10.1108/S0065-2830(2010)0000032005

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. I C. Nédellec & C. Rouveirol (Red.), *Machine Learning: ECML-98* (s. 4–15). Springer. https://doi.org/10.1007/BFb0026666

Li, H. (2011). A Short Introduction to Learning to Rank. *IEICE Transactions on Information and Systems*, *E94-D*(10), 1854–1862. https://doi.org/10.1587/transinf.E94.D.1854

Iii, E. P. C. (1980). The Optimal Number of Response Alternatives for a Scale: A Review. *Journal of Marketing Research*, *17*(4), 407. https://doi.org/10.2307/3150495

Macdonald, C., Santos, R. L. T., Ounis, I., & He, B. (2013). About learning models with multiple query-dependent features. *ACM Transactions on Information Systems*, *31*(3), 1–39. https://doi.org/10.1145/2493175.2493176

McHugh M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, *22*(3), 276–282.

Marchionini, G. (2006). Exploratory search: From finding to understanding. *Communications of the ACM*, *49*(4), 41–46. https://doi.org/10.1145/1121949.1121979

Mehta, P., Bukov, M., Wang, C.-H., Day, A. G. R., Richardson, C., Fisher, C. K., & Schwab, D. J. (2019). A high-bias, low-variance introduction to Machine Learning for physicists. *Physics Reports*, *810*, 1–124. https://doi.org/10.1016/j.physrep.2019.03.001

Om oss—Deichman.no. (2023). Hentet 20. mai 2023, fra https://deichman.no/

Rangering av resultater – Slik fungerer Google Søk. (2023). Google Søk – Finn ut hvordan Google Søk fungerer. Hentet 20. mai 2023, fra https://www.google.com/intl/no/search/howsearchworks/how-search-works/ranking-results/

Google. (2023). *Google Søk – Finn ut hvordan Google Søk fungerer.* Hentet 20. mai 2023, fra https://www.google.com/intl/no/search/howsearchworks/how-search-works/ranking-results/

Rhanoui, M., Mikram, M., Yousfi, S., Kasmi, A., & Zoubeidi, N. (2022). A hybrid recommender system for patron driven library acquisition and weeding. *Journal of King Saud University. Computer and Information Sciences*, *34*(6), 2809–2819. https://doi.org/10.1016/j.jksuci.2020.10.017

Robertson, S., Zaragoza, H., & Taylor, M. (2004). Simple BM25 extension to multiple weighted fields. *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, 42–49. https://doi.org/10.1145/1031171.1031181

Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, *27*, 111–125. https://doi.org/10.1016/j.inffus.2015.06.005

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620. https://doi.org/10.1145/361219.361220

Sakai, T., Tao, S., & Zeng, Z. (2022). *Relevance Assessments for Web Search Evaluation: Should We Randomise or Prioritise the Pooled Documents? (CORRECTED VERSION)*. https://doi.org/10.48550/ARXIV.2211.00981

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Muller, K.-R. (2021). Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proceedings of the IEEE*, *109*(3), 247–278. https://doi.org/10.1109/JPROC.2021.3060483

Samimi, P., & Ravana, S. D. (2014). Creation of Reliable Relevance Judgments in Information Retrieval Systems Evaluation Experimentation through

Crowdsourcing: A Review. *The Scientific World Journal*, *2014*, 1–13. https://doi.org/10.1155/2014/135641

ScienceDirect Topics. (2023). Hentet 20. mai 2023, fra https://www.sciencedirect.com/topics

Singh, D., & Singh, B. (2022). Feature wise normalization: An effective way of normalizing data. *Pattern Recognition*, *122*, 108307. https://doi.org/10.1016/j.patcog.2021.108307

Surowiecki, J. (2005). *The wisdom of crowds* (Nachdr.). Anchor Books.

Tang, R., Shaw, W. M., & Vevea, J. L. (1999). Towards the identification of the optimal number of relevance categories. *Journal of the American Society for Information Science*, *50*(3), 254–264. https://doi.org/10.1002/(SICI)1097-4571(1999)50:3<254::AID-ASI8>3.0.CO;2-Y

Wagstaff, K. L., & Liu, G. Z. (2018). Automated Classification to Improve the Efficiency of Weeding Library Collections. *The Journal of Academic Librarianship*, *44*(2), 238–247. https://doi.org/10.1016/j.acalib.2018.02.001

Wildemuth, B. M., & O'Neill, A. L. (1995). The «Known» in Known-Item Searches: Empirical Support for User-Centered Design (Research Note). *College & Research Libraries*, *56*(3), 265–281. https://doi.org/10.5860/crl_56_03_265

Zhao, R., & Mao, K. (2018). Fuzzy Bag-of-Words Model for Document Representation. *IEEE Transactions on Fuzzy Systems*, *26*(2), 794–804. https://doi.org/10.1109/TFUZZ.2017.2690222