Informasjonsgjenfinning i XML Dokumenter

Forord

Dette dokumentet, sammen med programkode og dokumentasjon på vedlagt CD, utgjør resultatet av hovedoppgave i faget TDT4900 Datateknikk, utført ved Instituttet for Datateknikk og Informasjonsvitenskap, Norges teknisk-naturvitenskapelige universitet, våren 2005. Veileder for hovedoppgaven har vært Trond Aalberg fra faggruppen for Informasjonsforvaltning ved instituttet.

Utgangspunket for oppgaven har vært en oppgavetekst gitt av overnevnte veileder, og baserer seg på et behov for å se nærmere på design og implementering av informasjonsgjenfinningssystem for XML dokumenter. Oppgaven har også bakgrunn i et fordypningsprosjekt utført høsten 2004, som satte fokus på state of the art innen IR i XML.

Den opprinnelige intensjonen med oppgaven var å implementere et slikt system, men da det etter hvert viste seg vanskelig å gjennomføre dette, ble oppgaven avgrenset. Den nye målsetningen ble å se på og belyse viktige aspekter ved utviklingen av et slikt system, og hva som skiller et slikt system fra tradisjonelle IR-systemer. Oppgaven kan også sees som en generell introduksjon til IR i XML.

Arbeidet har bestått av litteraturundersøkelser, testing av programvare, overordnet design, programmering og implementering (med mye prøving og feiling). Et resultat av den praktiske delen av oppgaven som kan trekkes frem er en omstrukturering av testsamlingen som benyttes i INEX.

Kent Rune Klungerbo, Trondheim

Juni, 2005

Innholdsfortegnelse

Sammendrag

Den enorme økningen av digitale dokumenter de siste 15 årene har ført til en eksplosjonsartet interesse for, og utvikling av, informasjonsgjenfinning. IR har lenge forholdt seg til dokumenter med lite eller ingen formell struktur, og har blitt dominert av de tre søkemodellene *boolsk modell*, *vektormodell* og *sannsynlighetsmodell*. Den stadig voksende mengden med digital informasjon har etter hvert ført til et behov for større formalitet, noe som har ført til utvikling og introduksjon av strukturerte dokumentformater som *SGML* og *XML*. Spesielt sistnevnte språk ser stadig mer utstrakt bruk som format for digital informasjon, noe som fører til nye utfordringer og muligheter innen IR.

Informasjonsgjenfinning i XML dokumenter setter nye krav til lagring av dokumenter, prosessering av innholdet i dokumentene, indeksering og søking etter informasjon. Et mangfold av muligheter åpner seg også, men ikke helt uten videre, inngående forskning og studier kreves for å kunne få utbytte av disse mulighetene. En viktig framdriftskraft i denne sammenhengen er *INEX*, et initiativ som eksisterer for å fremme utviklingen av informasjonsgjenfinning i XML dokumenter.

Endringene som kreves for å tilpasse seg de nye dokumentformatene ligger i detaljene. På overordnet nivå forholder ikke IR-systemer seg til om dokumentene er strukturerte eller ikke, det er i de enkelte modulene av systemet, hvor parsing, tekstbehandling, indeksering og søking foregår, at utfordringene ligger for forskere og utviklere. Samtidig presenterer XML nye muligheter for brukerne av slike systemer, gjennom nye måter å formulere spørringer på og nye muligheter for å presentere søkeresultater på.

IR i XML er et forskningsområde som er svært viktig for fremtiden innen håndtering av digital informasjon, og vil fortsette å få mye oppmerksomhet i lang tid fremover.

1 Introduksjon

Informasjonsgjenfinning fikk sitt store gjennombrudd med introduksjonen av World Wide Web på starten av 1990-tallet, og har siden den tid vært i drivende utvikling. Inntil nylig har mesteparten av informasjonen på web bestått av tekst uten noen form for formell struktur, eller med begrenset formell struktur, hvilket IR har tilpasset seg. Den senere tid har man dog opplevd økende interesse for mer strukturert informasjon, noe som setter nye og strengere krav til informasjonsgjenfinning, men som også åpner mange muligheter for bedre og mer presis håndtering av den enorme informasjonsmengden på weben, og ikke minst informasjonsgjenfinning i denne. Sentralt når det gjelder strukturert informasjon på web står *eXtensible Markup Language*, bedre kjent som XML.

Denne monumentale forandringen innen IR har vært motivasjonen for denne oppgaven, som setter fokus på IR-systemer for XML, hvordan de skal se ut og hva de skal inneholde.

1.1 Oppgaven

Den opprinnelige oppgaveteksten var som følger:

"I denne oppgaven skal studentene modellere og utvikle et informasjonsgjenfinningssystem for XML dokumenter. Systemet skal baseres på tradisjonelle IR-teknikker men også kunne håndtere og ta i bruk dokumentenes XML struktur i gjenfinning og presentasjon av informasjon.

Systemet skal være modulært slik at det er mulig å utvikle og teste ut forskjellige IR-teknikkker for XML-dokumenter. Testsamlingen som skal benyttes er INEX-samlingen av IEEE artikler, og systemet skal kunne benyttes for testresultater som kan avgies til INEX (initiativ for evaluering av XML informasjonsgjenfinning).

Oppgaven er egnet for to samarbeidende studenter, men kan også utføres av en enkelt student."

Som bakgrunn for denne oppgaven ligger også et fordypningsemne som la fokus på state of the art innen IR i XML [IRX], og som så på resultater og alternativer så langt innenfor dette feltet. Sentralt i denne fordypningen stod *INEX (Initiative for the Evaluation of XML Retrieval)*, og dette vil man også komme inn på i denne oppgaven.

Hovedfokuset i denne oppgaven har vært design og utvikling av IR-systemer for XML. Spesielt er det sett på hva som skiller et slikt system fra tradisjonelle IR-systemer, hvilke nye krav til design og implementering man får når systemet skal håndtere den strukturerte informasjonen man finner i XML-dokumenter. Ekstra fokus har man også satt på dokumentsamlingen til INEX og håndteringen av denne i IR-systemer. Under følger en avgrenset presisering av hovedoppgaven.

1.1.1 Avgrensninger

I forkant av og underveis i arbeidet med hovedoppgaven har det blitt gjort en del avgrensninger og presiseringer av oppgaven, her presenteres disse:

- Underveis i arbeidet med oppgaven ble det etter hvert klart at utviklingen av et helt system var en for omfattende oppgave, og oppgaven ble derfor redusert til et mer teoretisk studium av utviklingen av et slikt system, slik at man kunne gjøre seg erfaringer og sette fokus på sentrale problemstillinger og valg man må gjøre. Den praktiske delen av oppgaven ble heller redusert ned til å produsere så mye av systemet som tiden tillot, og la hovedfokuset ligge på de teoretiske aspektene ved utviklingen av et slikt system.

1.2 Om dette dokumentet

I neste kapittel startes det med en introduksjon til tradisjonell IR, XML og IR i strukturerte dokumenter, samt iniativet INEX, før kapittelet avsluttes med å liste opp noen designkriterier til et IR-system for XML dokumenter. Kapittel 3 omhandler valg som ble gjort og ville blitt gjort under utviklingen av et slikt system, før kapittel 4 setter fokuset på testsamlingen fra INEX og arbeidet som ble gjort med denne i hovedoppgaven. I kapittel 5 presenteres et enkelt og overordnet design av et IR system for XML dokumenter, der fokuset ligger på hvilke pakker og komponenter et slikt system består av, og hvilke grensesnitt de benytter for å kommunisere med hverandre. I kapittel 6 legges frem de konklusjoner man klarte å dra gjennom arbeidet med oppgaven, i kapittel 7 sies det litt om erfaringer som ble gjort, mens kapittel 8 til slutt retter blikket fremover og diskuterer litt rundt mulighetene for videre arbeid innenfor dette feltet.

2 Informasjonsgjenfinning i XML

I dette kapittelet vil man kort presentere bakgrunnen for hovedoppgaven, i tillegg til generell IR og IR i XML inkluderer dette også INEX (Initiative for the Evaluation of XML Retrieval). Til slutt i kapittelet vil man se på krav til design av et IR system for XML dokumenter.

2.1 Generelt om informasjonsgjenfinning

Informasjonsgjenfinning (*information retrieval – IR*) omfatter representasjon, lagring, organisering av og tilgang til informasjonselementer [BAE]. I et historisk perspektiv ble dette feltet sett på som svært snevert, men det hele eksploderte med introduksjonen av World Wide Web tidlig på 1990-tallet, sammen med en kraftig økning i mengden av digitale dokumenter også på andre områder. Etter hvert har også omfanget av feltet økt til å inkludere modellering, dokumentklassifisering og klassifisering, systemarkitektur, brukergrensesnitt, datavisualisering, filtrering og håndtering av språk.

2.1.1 Å dekke et informasjonsbehov

Hovedutfordringen ved IR er å finne den informasjonen som en bruker spesifikt er ute etter. Dette må ikke forveksles med *datagjenfinning* i en database, hvor gjenfinningen består i å sammenligne en strengt formalisert spørring med innslag i databasen, og returnere de tuplene som gir eksakt treff. Denne teknikken ble tidlig forsøkt brukt også i informasjonsgjenfinning (ved å returnere de dokumentene som inneholder alle ordene fra spørringen, i IR-sammenheng blir dette kalt *boolsk metode*), men gir svært dårlige resultater med hensyn på å returnere den egentlige informasjonen som brukeren er ute etter. For informasjonsgjenfinning i en dokumentsamling blir det heller snakk om å veie *relevansen* til hvert dokument i forhold til spørringen, og denne veiningen kan utføres på mange forskjellige måter.

2.1.2 Teknikker og metoder

Først og fremst er det på sin plass å definere begrepet *relevans* i denne sammenhengen. Det er flere måter å måle relevans på, men den mest utbredte måten å gjøre det på er å bruke de to skalaene *recall* og *precision*. Recall er et mål på hvor stor andel av de dokumentene som er relevante til spørringen som faktisk ble returnert av IR-systemet, mens precision er et mål på hvor stor andel av de returnerte dokumentene som faktisk var relevante. Sjeldent klarer et IR system å være optimalt i forhold til begge disse målene, hvis precision er svært høy så kan recall bli dårlig, og vice versa. Ofte blir det nødvendig med en avveining i forhold til hvilken av de to som er viktigst i bruken av IR-systemet. Om man syns det er tungt å forholde seg til to parametere når man måler kvaliteten av et IR-system, så har det også blitt utviklet måter å kombinere disse to verdiene på, der man kan justere vektingen av precision i forhold til recall ved å endre en parameterverdi i utregningsformelen.

Når det gjelder veiing av relevans mellom dokumenter og spørring, er det tradisjonelt sett tre teknikker som har blitt brukt: Den allerede nevnte *boolske metoden*, i tillegg til den har man *vektormetoden* og også *sannsynlighetsmodell*. Boolske metoder benytter binær relevans, enten så inneholder et dokument ordene fra spørringen, eller så gjør den det ikke. Dette gir dårlige utslag for både recall og precision, og er egentlig bedre egnet for datagjenfinning og bibliografisk søking. Vektormodellen [GSA, SAL] er en svært modifisert utgave av den boolske modellen, og til dags dato den klart mest brukte IR-teknikken. Den bruker vektor-representasjoner av spørringer og dokumenter, og bruker avstanden mellom disse vektorene til å måle relevans, vanligvis måles denne avstanden ved å ta cosinus til vinkelen mellom vektorene; desto mindre vinkel mellom dem - desto større relevans mellom dokument og spørring. Vektormodellen benytter en spesiell *invertert indeks* […]. Sannsynlighetsmodellen, introdusert av S. E. Robertson og K. Sparck Jones i 1976 [ROB], forsøker, gjennom en iterativ interaksjon med brukeren, å finne den beste sannsynlige beskrivelsen av det ideelle søkeresultatet til brukerens spørring.

Andre alternativer til de tre overnevnte, klassiske modellene er *språkmodellering* og *mengdelæremodeller* samt alternative/modifiserte algebraiske modeller og sannsynlighetsmodeller.

En av de større utfordringene med IR er å formulere gode spørringer, det vil si spørringer som får systemet til å returnere den informasjonen man er ute etter. Det kan ofte være vanskelig å formulere en slik god spørring på første forsøk – ofte vet man nøyaktig hva man er ute etter av informasjon, men man vet ikke hvordan man skal forklare dette til et IR system. I denne sammenhengen har man *Relevance Feedback*, en strategi som går ut på at man justerer spørringen over flere søk. Etter hvert søk så velger brukeren ut de dokumentene som synes relevante, og basert på dette forsøker systemet å justere spørringen før den gjør et nytt søk, for slik å stadig oppnå et bedre søkeresultat.

Et viktig bidrag til å drive frem utviklingen av IR har vært *Text REtrieval Conference (TREC*) [TRE], sponset av National Institute of Standards and Technology (NIST) og U. S. Department of Defence. Initiativet ble startet i 1992, og dets mål var å fremme forskning innen IR-miljøet og legge til rette for dette ved å tilby den infrastrukturen som er nødvendig for storskala tekstgjenfinningsmetoder. Mange av dagens kommersielle søkemaskiner er basert på teknologi først utviklet i TREC.

2.2 XML - eXtensible Markup Language

Svært sentralt i fremveksten av strukturert informasjon på web står *eXtensible Markup Language (XML)* og dens forgjengere *GenCode, General Markup Language (GML)* og *Standard Generalized Markup Language (SGML)* [ABI]. SGML var den første store standarden for strukturerte dokumenter, den ble innført som ISO standard i 1986. SGML hadde en svært stor og komplisert spesifikasjon, noe som gjorde det til en tungvint standard å jobbe med, både for applikasjoner og for utviklere, og ut av dette vokste behovet for en ny standard for strukturert informasjon.

I 1996 nedsatte *World Wide Web Consortium (W3C)* en arbeidsgruppe for å se på dette problemet, og påfølgende år kunne denne gruppen presentere markupspråket XML, som utgjorde en delmengde av SGML. XML ble utstedt som en W3C Recommendation den 10. februar 1998. Det har blitt sagt om XML at det støtter 90 % av SGML's funksjonalitet, men det kun innehar 10 % av forgjengerens kompleksitet.

2.2.1 XML struktur

De fleste som bruker informasjon på web har et forhold til *Hyper Text Markup Language (HTML)*, og dermed også et forhold til den strukturen man finner i både HTML og XML. Men der HTML har som formål å presentere informasjon, har XML heller som formål å formatere informasjon. Det må også presiseres at HTML er en applikasjon av SGML, det er ikke sidestilt med SGML og XML som en egen standard for strukturerte dokumenter. Faktisk fins det en versjon av HTML, *XHTML*, som i realiteten er en omskrivning av HTML i XML, og således en applikasjon av XML.

Et XML dokument består kort forklart av et sett elementer organisert som en hierarkisk rettet, asyklisk graf. Hvert element består som regel av en start-tag og en slutt-tag, for eksempel `<land>Norge</land>`, men kan forenkles til kun en tag, eksempelvis `<land navn='Norge' />`. Hvert element kan inneholde vilkårlig mengde med tekst, og et vilkårlig antall subelementer. Start-tag kan også inneholde attributter. Dette er vist i eksempelet under:

```
<land navn='Norge'>
      <hovedstad>Oslo</hovedstad>
      <valuta>Krone</valuta>
</land>
```

Denne enkle strukturen er enkel å håndtere for både utviklere og applikasjoner, samtidig som den er svert fleksibel med tanke på å representere informasjon og data. XML blir også brukt som dataformateringsspråk i databaser og datalagre, selv om det på dette området innehar en del svakheter (hvorav de største bakdelene består i at rekkefølge på elementer *ikke* er vilkårlig, samt mangel på datatyper). Denne fleksibiliteten har bidratt til å minske gapet mellom IR-kulturen og datakulturen de siste årene.

2.2.2 DTD - Document Type Definitions

Mye av årsaken til XML's fleksibilitet, er at språket har muligheten til å definere sin egen grammatikk. Dette kan gjøres i en *Document Type Definition* deklarasjon i selve XML dokumentet, eller i et eksternt dokument som det refereres til (ofte å foretrekke hvis man ønsker å bruke samme format i mange dokumenter). I en DTD kan man definere hvilke elementer som dokumentet kan inneholde, samt hva innholdet av disse skal være. Det kan også defineres datatyper, blant annet kan man definere referanse-attributter som kan brukes til å representere koblinger mellom elementer i dokumentene.

En DTD for eksempelet over kan være følgende:

```
<!DOCTYPE  land [
      <!ELEMENT land (hovedstad, valuta)>
      <!ATTLIS land navn CDATA #REQUIRED>
      <!ELEMENT hovedstad (#PCDATA)>
      <!ELEMENT valuta (#PCDATA)>
]>
```

Et XML dokument som er i samsvar med en DTD, er et *gyldig (valid)* XML dokument, og dette er en svært interessant egenskap i sammenheng med IR; hvis man i søking kan basere seg på definisjoner av grammatikken i dokumentene, er det lettere å foreta presise søk. Ved homogene dokumentsamlinger, hvor hele samlingen har en felles DTD (som beskrevet tidligere), økes denne muligheten ytterligere.

DTD er godt egnet som grammatikkspråk for XML som informasjonsformateringsspråk, men har en del begrensninger som definisjonsspråk for bruk av XML som datalagring. Dette har ført til utviklingen av en del alternativer, hvorav *XML Schema* [W3C] er en av de mer velkjente løsningene. Det er beskrevet i XML, og inneholder en del rekonstruksjoner og utvidelser i forhold til DTD. Det vil dog ikke få noen nærmere presentasjon i denne oppgaven, i stedet skal fokuset settes over på IR i XML og i strukturert informasjon generelt.

2.3 Informasjonsgjenfinning i strukturerte dokumenter

Tradisjonell IR har altså tilpasset og optimalisert seg for ustrukturerte dokumenter. Den stadig voksende mengden med strukturert informasjon på web har derimot ført til at man må se på IR med nye øyne, i alle fall delvis nye. Mens mange egenskaper ved de gamle IR-systemene ikke legger noen hindringer for håndtering av mer strukturert informasjon, er det også en god del endringer og tilpasninger som må til i forhold til de nye dokumentformatene. Man opplever nye krav til både dokumenthåndtering, indeksering, søking, samt nye muligheter for interaksjon mellom brukeren og IR-systemet. Her vil de viktigste forskjellene presenteres.

2.3.1 Homogene og heterogene dokumentsamlinger

Før man begynner å diskutere informasjonsgjenfinning i strukturerte dokumenter, så der det viktig å påpeke et viktig skille innenfor dette feltet: *homogene* og *heterogene* dokumentsamlinger, det vil si samlinger med og uten felles dokumentstruktur. Homogene dokumentsamlinger har en felles struktur, og er mye enklere å forholde seg til i forhold til IR. Når man kjenner strukturen i dokumentene kan man utnytte denne både i søkealgoritmen og ved å angi presise strukturelle argumenter i spørringer. Homogen dokumentstruktur finner man som regel i artikkeldatabaser (som for eksempel *INEX dokumentsamling* som presenteres senere i oppgaven), juridiske dokumentsamlinger, organisasjonsinterne dokumenter og lignende.

Å forholde seg til en verden med kun homogene dokumentsamlinger er derimot urealistisk, spesielt når det er snakk om internett. Da må man gå over til å snakke om heterogene dokumentsamlinger, det vil si samlinger av dokumenter hvor strukturen kan variere. Når man ikke har kjennskap til strukturen i dokumentene på forhånd, og heller ikke kan anta lik struktur i alle dokumentene innenfor søkemengden, blir IR fort litt mer utfordrende. Det er allikevel også her mulig å utnytte strukturen i

dokumentene for mer presis IR enn i ustrukturert informasjon, men mulighetene er litt mer begrenset enn for homogene samlinger.

2.3.2 Lagring av dokumenter

Tradisjonell IR setter ikke andre krav til dokumentlagring enn at dokumentene kan hentes ut raskt og effektivt når de skal presenteres i resultatet av en spørring. Med mer kompleks oppbygning av dokumentene, slik man finner det i strukturerte dokumenter, får man også litt strengere krav til lagring og representasjon av disse dokumentene.

Det viktigste her handler om å ivareta og representere strukturen i dokumentene på en god måte. Ved søk i strukturert informasjon kan det ofte være interessant å returnere mindre informasjonsenheter enn hele dokumenter, og således må dokumentene være lagret på en måte som gjør det enkelt å hente ut strukturelle subelementer fra disse.

En annen problemstilling her er store dokumenter. Med god formell struktur i dokumenter åpner det for muligheten til å putte veldig mye informasjon i ett og samme dokument, men dette er lite ideelt med tanke på dokumentlagring. De fleste databaser liker dårlig å håndtere store dokumenter, dermed utgjør dette lille paradokset en av de mange utfordringene ved IR i strukturerte dokumenter.

2.3.3 Indeksering

På lik linje med vanlig IR vil indekseringen i et IR-system være svært avhengig av hvilken søketeknikk som brukes i systemet, men det fins generelt sett noen forskjeller i forhold til indekser for ikke-strukturerte dokumenter. Den viktigste forskjellen ligger i muligheten for å indeksere mindre enheter enn hele dokumenter, som for eksempel <paragraph>-elementer i et XML-dokument. Dette er også ofte noe man ønsker å implementere i IR-systemer for strukturert informasjon, da det gir mer presise søkeresultater i form av at man kun returnerer de delene av et dokument som er relevante, i stedet for hele dokumentet.

Denne muligheten til å variere granularitet på indeksen åpner for flere typer løsninger:

- Full indeksering. Alle strukturelle elementer i dokumentene indekseres, både store og små. Tillater svært detaljerte søk, kun strukturen i dokumentene begrenser hvor små informasjonsenheter som kan returneres for en spørring. Fører til en del overhead under indeksering siden mange elementer indekseres flere ganger (som deler av større elementer), samt en svært stor indeks.
- Nivåbestemt indeksering. Man indekserer på elementer i et bestemt hierarkisk nivå i dokumentene, for eksempel kan man velge å indeksere på <artikkel>-elementer i en samling hvor dokumentene utgjør tidsskrifter. Et alternativ hvis dokumentstrukturen ikke er kjent på forhånd kan være å indeksere elementer over en viss størrelse, eventuelt innenfor et størrelses-intervall.
- Hierarkisk indeksering. Elementer indekseres hierarkisk, slik at et elements innslag i indeksen beregnes ut i fra subelementene, samt eventuelt eget innhold.

Det er generelt svært ønskelig å ha en god representasjon av hierarki/sti i indeksen. Dette gir flere muligheter for å utnytte strukturen i dokumentene for mer presis søking. Termer kan for eksempel vektes etter hvor langt nede i dokumenthierarkiet de befinner seg. Det aller viktigste for indeksen er dog at den er tilpasset og optimalisert søkealgoritmen(e) i systemet.

2.3.4 Søking

Den delen i IR-systemet som opplever flest nye muligheter og utfordringer ved strukturerte dokumenter, er søkemodulen. Det er kanskje viktig å påpeke at hovedvekten her ligger i *mulighetene*. Man har generelt sett to måter å søke på i strukturert informasjon; med og uten strukturelle argumenter i søkingen, i *INEX* kalles dette for hhv. *Content Only* og *Content and Structure* søk (se

kapittel 2.4.2). Mulighetene som de to søkemetodene har i seg avhenger veldig av om det er søking i homogene eller heterogene dokumentsamlinger det er snakk om. Spesielt har spørringer med strukturelle argumenter bedre forutsetninger for å presisere søket hvis strukturen i dokumentene er kjent på forhånd, og lik hos alle dokumentene innenfor søkemengden. Uten strukturelle argumenter til å begrense søket, blir det i større grad opp til systemet å finne den informasjonen som brukeren er ute etter.

Uansett betingelser så fins det et mangfold av teknikker og strategier som er prøvde og utprøvde, her er utviklingen i stadig fremdrift.

2.3.5 Brukerinteraksjon

Strukturert informasjon presenterer nye muligheter både for input og output til brukeren. Muligheten til å angi strukturelle argumenter i spørringer er allerede nevnt. Ved bruk av CAS spørringer settes det litt ekstra krav til interaksjonen mellom bruker og system; enten må brukeren ha kunnskap om strukturen i dokumentene, eventuelt kunnskap om hvordan han angir strukturelle argumenter, eller så må systemet på en god og intuitiv måte gjøre det enkelt for bruker å angi slike argumenter i søket sitt.

Strukturerte dokumenter gir også muligheten til å gi mer strukturerte presentasjoner av søkeresultatene, og således åpnes det for at brukere kan få større utbytte av de resultatene som returneres av søkemotorene.

2.4 INEX

INEX, eller *Initiative for the Evaluation of XML retrieval* [INE], er et initiativ som har for mål å tilby en felles arena for utvikling og evaluering av IR i XML. Siden 2002 har dette vært et årlig tiltak hvor utviklere melder seg på for å teste IR-systemene sine opp mot hverandre. På slutten av hvert år holdes det en workshop hvor deltakerne kan møtes for å presentere, sammenligne og diskutere resultatene sine. Resultatene hvert år publiseres i en rapport [I03]. For å på best mulig vis sammenligne resultater fra de enkelte bidragene hvert år, stiller INEX med dokumentsamling, spørringer og mål for relevans som deltakerne tester systemene sine opp mot. Disse er beskrevet lenger ned.

INEX består også årlig av en del mindre initiativer som setter fokus på viktige aspekter ved IR i XML. I 2004 hadde de for eksempel 4 mindre initiativer:

- *Interactive Track:* Tester for å undersøke brukeres synspunkter på IR i XML. Statistikk fra disse testene ble samlet sammen og distribuert til deltakerne.
- *Relevance Feedback:* Undersøkelser for å utnytte karakteristikken til XML i relevance feedback.
- *Heterogenous collection:* Ser på mulighetene for å opprette en heterogen dokumentsamling for fremtidig bruk i INEX. Som beskrevet lenger ned er den nåværende samlingen homogen, med en felles DTD for hele samlingen.
- *Natural query language:* Ser på bruk av spørringer med naturlig språk over samlinger av XML dokumenter.

Med disse initiativene kan INEX bidra til å styre utviklingen innen feltet, og rette fokus mot temaer som kanskje ikke har fått den nødvendige oppmerksomheten så langt av forskningsmiljøene.

2.4.1 Dokumentsamling

Testsamlingen som benyttes i INEX er en samling vitenskapelige artikler donert av *Institute of Electrical and Electronics Engineers* [IEE], bedre kjent som *IEEE*. Samlingen består av 12107 artikler fra tidsskrifter utgitt av IEEE i perioden 1995 – 2002, og utgjør til sammen 500 mb. Testsamlingen er

homogen, med en felles DTD for hele samlingen, hvilket gjenspeiler hvordan INEX så langt har fokusert på søking i dokumentsamlinger hvor strukturen er kjent på forhånd. Som nevnt over jobbes det med å rette noe av fokuset over på heterogene dokumenter også, da det faktisk er dette som ofte er realiteten når man søker etter informasjon: man har ingen kjennskap til strukturen i den informasjonen man søker i, og denne kan variere. Spesielt er dette tilfelle på internett.

2.4.2 Spørringer

Hvert år har man et gitt sett med spørringer som benyttes i testingen av systemene, og disse er delt opp i to kategorier: *Content Only (CO)* og *Content and Structure (CAS)*. Spørringer av formatet CO søker, som navnet tilsier, kun på innhold, ikke på struktur. Dette tilsvarer søking slik det blir gjort i tradisjonelle IR-systemer. I spørringer av formatet CAS angir man strukturelle argumenter, og utnytter med andre ord en mulighet som ikke fins i tradisjonell IR. Man kan for eksempel angi at man kun ønsker å søke i navngitte elementer, eller i elementer av en viss størrelse. Mulighetene er mangfoldige, spesielt i homogene dokumentsamlinger hvor strukturen i dokumentene er kjent, men også for generell søking i strukturert informasjon.

2.4.3 Bedømmelse av relevans

For å evaluere søkeresultatene til deltakerne, ble dokumentene vurdert i forhold til de gitte spørringene etter følgende to skalaer: *exhaustivity* og *specificity*. Exhaustivity (e-value) forteller noe om hvor grundig et informasjonselement omhandler det man søker etter, alt i fra en liten del av det man søker informasjon om, til å dekke alt det man ønsket å finne ut. Specificity (s-value) sier noe om i hvilken grad informasjonselementet er konsentrert rundt det man søkte informasjon etter, det utgjør alt fra en liten del av informasjonselementet til å være det eneste som det omhandler. Begge disse skalaene er delt i 4 grader: not (0), marginally (1), fairly (2) og highly (3). Disse vurderingene ble foretatt av nedsatte grupper med folk med ekspertise innen de emnene som spørringene representerte.

For å evaluere søkeresultater har det blitt benyttet en rekke måleenheter, noen av dem basert på de tradisjonelle enhetene *precision* og *recall*, men også alternative evalueringsstrategier. Se [I03] for en nærmere presentasjon av disse.

2.5 Designkriterier for et IR-system for XML dokumenter

Med bakgrunn i det som har blitt presentert så langt er det nå mulig å sette opp noen overordnede designkriterier for et IR-system for XML dokumenter. Her følger en liste over krav til et slikt system:

I. Dokumenter skal lagres i en database med god støtte for håndtering av XML dokumenter. Viktigst med rask og effektiv uthenting av både hele dokumenter og av elementer.

II. Systemet må håndtere vanlig parsing av XML, samt analyse og parsing av vanlig tekst. Det skal også støtte stoppordfjerning og stemming. Siden systemet skal bruke INEX testsamling, som er på engelsk, holder det at systemet støtter tekstbehandling innenfor dette språket.

III. Indeksering skal foregå på et nivå slik at spørringer returnerer informasjonselementer av fornuftig størrelse. Dette skal være en invertert indeks tilpasset vektorsøk, og sti til indekserte termer skal representeres.

IV. Søkemotoren skal ta utgangspunkt i den klassiske vektormodellen, men skal være tilpasset søking i XML-strukturen og skal utnytte denne til å bedre presisjon i søkingen. Den skal kunne returnere elementer på forskjellige nivåer som resultat på spørringer.

V. Systemet skal støtte spørringer med strukturelle argumenter, det vil si spørringer av typen *Content and Structure*.

3 Valg og erfaringer

Et viktig fokus for hovedoppgaven har vært å se på valgene som gjøres når man utvikler et slikt system. Hvilke konsekvenser får disse valgene for sluttresultatet, samt for andre valg som gjøres videre i utviklingen av systemet. I dette skal de viktigste valgene diskuteres, både (tekniske) løsninger og programvarevalg der dette var nødvendig.

3.1 Databasesystem

Det første man la fokus på ved utviklingen av systemet, var valg av database for lagring av XML dokumenter og indeks. Databasen måtte ha et grensesnitt som støttet programmering mot Java, og ytelse var viktig. Ingen store krav til funksjonalitet, da bruk av databasen stort sett innebar lagring, enkel henting og oppdatering av data. Noen av de databaseløsningene som ble vurdert var *eXist* [XST] og *Apache Xindice* [XIN], samt *Berkeley DB* og ekstensjonen *Berkeley DB XML*.

Man kom etter hvert fram til at løsninger som eXist og Xindice ble litt svære for dette systemet, da man som nevnt over kun var ute etter enkel funksjonalitet (siden søking skulle foregå i en egen indeks). All funksjonaliteten i disse systemene gav bare uønsket overhead på ytelsen, og da så Berkeley DB mer og mer ut som et mer aktuelt alternativ. At Berkeley DB er såkalt *embedded*, det vil si at databasen opprettes og kjøres i samme navnerom som resten av applikasjonen, må også trekkes frem som en grunn til at den ble valgt til dette systemet fremfor de mange klient-tjener databasene som eksisterer (Apache Xindice er et eksempel på dette).

3.1.1 Berkeley DB

Sleepycat [SLC] tilbyr tre utgaver av databasen sin: *Berkeley DB* (BDB), ekstensjonen *Berkeley DB XML*, samt *Berkeley DB Java Edition*. De to førstnevnte er utviklet i C++, men har begge Java API'er. BDB JE er fullt og helt bygd opp i Java, men med samme funksjonalitet som i vanlig BDB. Sleepycat trekker frem følgende karakteristiske trekk ved databaseløsningen sin:

- Enkel bruk. Utviklere har muligheten til å bruke funksjonskall direkte mot databasen, i stedet for å gå via SQL og/eller lignende spørrespråk. Det er også en svært enkel database å administrere etter installasjon.
- Open Source Distribuering. Enkelt å tilpasse databasen, i tillegg til at du har en kvalitetsgaranti ved at mange tusen utviklere har gjennomgått og vurdert koden.
- Lite fotspor. Hele BDB-pakken tar svært lite plass (~375K) på de vanligste arkitekturene.

Se [BDB] for en lengre liste over tekniske momenter ved databaseløsningen.

Indekseringen som BDB benytter er også meget godt egnet til å håndtere en invertert indeks, som det var planen å implementere i dette systemet. Systemet skulle som nevnt utvikles i Java, og det ble dermed naturlig å først se på java-versjonen av Berkeley DB. Det viste seg derimot kjapt at denne hadde få fordeler fremfor standardløsningen. En rask sammenligning med BDB XML viste også at håndtering av XML dokumenter ble unødvendig tungvint i BDB JE. Som et resultat av dette ble det bestemt å bruke BDB XML videre i utviklingen.

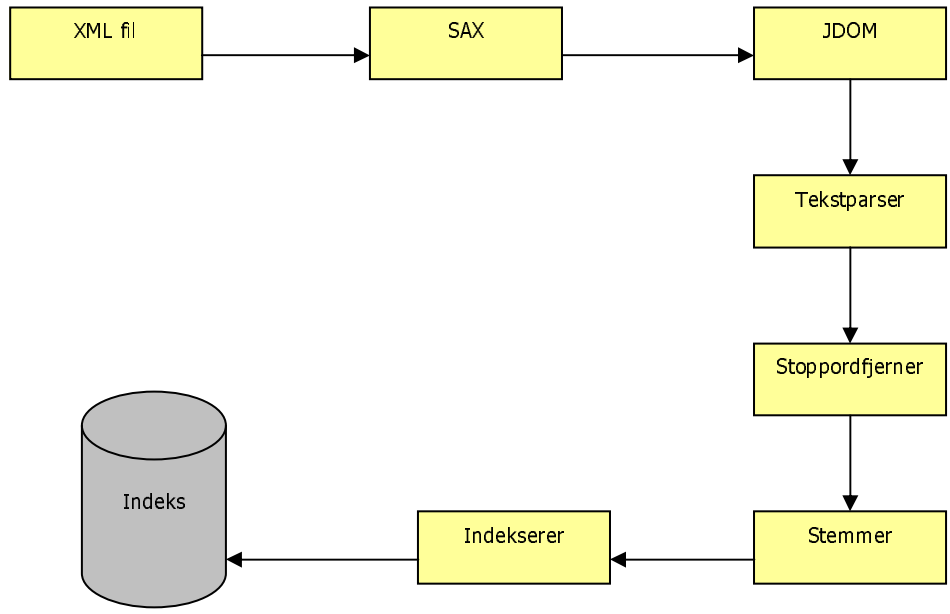
Resultatet av arbeidet med å implementere en dokumentdatabase i Berkeley XB XML foreligger i filen `DBXMLTest.java` på vedlagt CD-ROM. Her legges den dokumentene i den nye dokumentstrukturen til INEX testsamling, beskrevet i kapittel 4, inn i en enkel XML-database. Løsningen er uferdig og fungerer ikke hundre prosent, men gir et lite innblikk i hvordan man setter opp en XML database i Berkeley DB XML. I kapittel 7 foreligger noen erfaringer som ble gjort med produktet under utviklingen av systemet.

3.2 Tekstanalyse

Tekstanalyse i et IR system for XML dokumenter skiller seg lite, om ingenting, fra tradisjonelle IR systemer. Hovedforskjellen ligger i XML-taggene som må tas hensyn til, men dette byr på få utfordringer. All tekst vil normalt gå gjennom en XML-parser før de gjennomgår tradisjonell tekstanalyse- og behandling, hvilket inkluderer stoppordfjerning og stemming. Figur 1 lenger ned viser hvilke prosesser en XML fil går gjennom før den til slutt lagres i indeksen.

3.2.1 XML parsing

Parsing av XML foregår enkelt ved bruk av JDOM's [JDO] SAX-parser. SAX står for *Simple API for XML,* og er et hendelsesdrevet grensesnitt for parsing av XML data. Når innholdet i dokumentene har blitt parset av SAX representeres innholdet i Java ved hjelp av JDOM. JDOM står for *Java Document Object Modell*, og slik navnet tilsier, representerer JDOM XML dokumenter som objektmodeller, der objektene er dokumenter, elementer og attributter med referanser til hverandre. På vedlagt CD-ROM har SAX og JDOM blitt benyttet for konvertering av INEX testsamling (se kapittel 4) i filen `JDOMTest.java`.



Figur 1

3.2.2 Tekstparsing og analyse

Når dokumentinnholdet har vært gjennom SAX-parseren og er representert som dokumenttrær i JDOM, gjenstår kun standard analyse og tekstbehandling slik som i tradisjonelle IR-systemer. For dette systemet ble det planlagt å bruke den nye `Scanner`-klassen i Java for å gjennomføre disse oppgavene, og det viste seg som en god løsning under testing. Stoppordfjerning ble også utført med denne klassen, og baserer seg på en vedlagt tekstfil som definerer hvilke stoppord som skal fjernes fra dokumentene før de indekseres. En del leting på web og enkel testing konkluderte med at *Porter Stemming Algorithm* [POR] løste oppgaven med ordstemming på en tilfredsstillende måte. Bruk av `Scanner` og Porter Stemming Algorithm til å prosessere dokumenttekst presenteres i filene `Scantest.java` og `Stemmer.java` på vedlagt CD-ROM.

3.3 Indeksering

Fokuset i denne hovedoppgaven har ikke vært å lage et optimalt IR system for XML dokumenter, men å gjennomgå en utvilkling av et slikt system og belyse viktige aspekter ved et slikt system, hvilke designvalg som er viktige, og hva slags konsekvenser slike valg får for resten av systemet.

Det var derfor aldri et mål å lage en optimal indekseringsmodul, men en fungerende indekserer som utnytter XML-strukturen og legger til rette for at søkemodulen skal kunne gjøre det samme. En indeks for strukturerte dokumenter har noen viktige momenter i forhold til en indeks for tradisjonelle, ustrukturerte dokumenter:

- Granularitet. På hvilket nivå i strukturen skal dokumentene indekseres? Skal indekseringen foregå dokumentvis, skal hvert eneste element indekseres, eller skal elementer ned til et visst nivå indekseres?
- Representasjon av sti. I et XML dokument vil stien frem til elementet der et ord forekommer, være avgjørende for hvordan ordet skal vektes i et søkeresultat. Det er derfor svært ønskelig å lagre informasjon om sti i indeksen.

For ikke å komplisere utviklingen av systemet utover oppgavens omfang, ble det ikke satt så strenge krav til granulariteten, men man bestemte seg heller for å prøve seg frem. Man valgte å starte med dokumentvis indeksering (fra den nye strukturen, se kapittel 4), det vil si å indeksere på `<journal>`-elementene. Dette ble dessverre aldri implementert, se kapittel 1.2 om avgrensninger av oppgaven.

3.4 Søkemotor

De samme kriteriene gjelder her som for indeksereren, målet her er en enkel søkemotor som på ingen måte skal være en optimal løsning, men som er tilpasset søking i XML og utnytter den struktur til å øke presisjonen i søkingen. Om man kikker på resultatene fra INEX [I03] ser man fort at det fins mange alternative måter å implementere en slik søkemotor på. I denne oppgaven ønsket man å ta utgangspunkt i den allerede ustrakte bruken av vektormodellen i tradisjonell IR, ved å implementere en vektorbasert søkemotor tilpasset søking i XML.

Det fins mange gode grunner til å se på mulighetene til å overføre vektormodellen fra tradisjonell, ustrukturert IR til IR i XML og annen strukturert informasjon. Det er som kjent liten vits i å finne opp kruttet på nytt, og hvis man med noen endringer og justeringer kan fortsette å bruke en allerede eksisterende og velbrukt løsning, så har man spart seg mye ekstraarbeid.

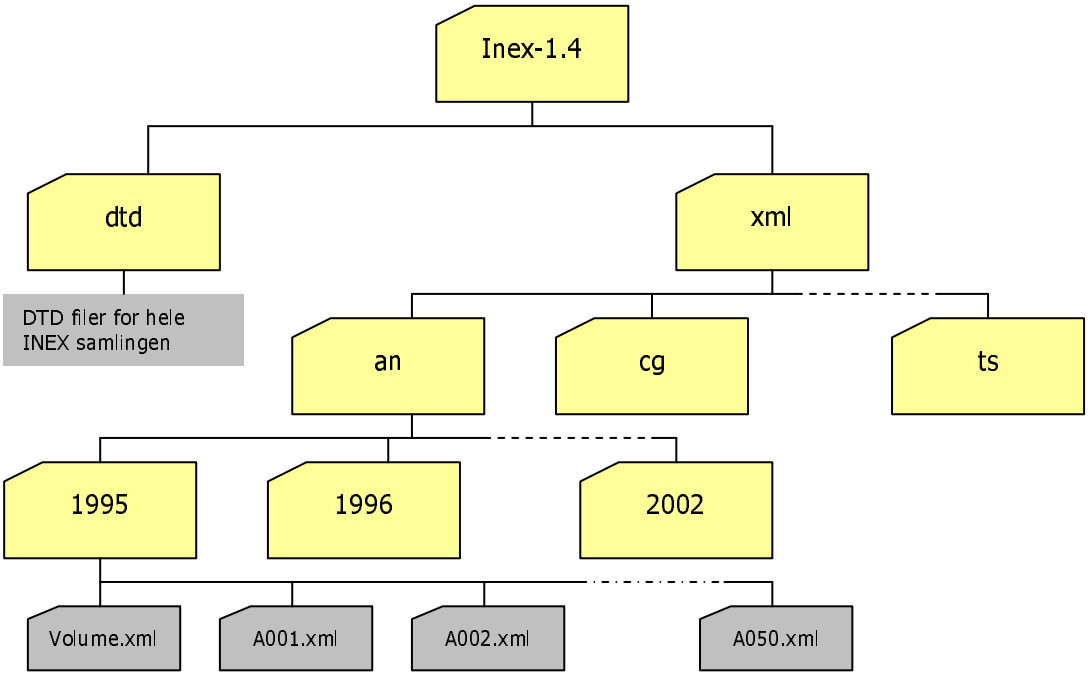
I INEX 2003 ble det presentert en løsning hvor vektormodellen hadde blitt tilpasset XML [MAS] ved å ta i beregning statistikk for elementer i dokumentene, og rangere disse fremfor å bare rangere dokumentene. Planen var å implementere en lignende løsning i dette systemet, men dessverre kom man aldri så langt i den praktiske delen av denne oppgaven, jfr. avgrensninger i kapittel 1.2.

4 Omstrukturering av INEX dokumentsamling

I oppgaveteksten stod det skrevet at INEX-samlingen [REF] av IEEE artikler skulle benyttes under utviklingen av systemet. Dette var også naturlig, da denne samlingen hadde vært i fokus allerede i fordypningsemnet [REF] foregående semester. Det ble i tillegg tidlig bestemt at man skulle utvikle IR-systemet i Java, følgelig måtte man se nærmere på hvordan format og struktur i INEX-samlingen passet med programmering i Java.

4.1 INEX dokumentsamling

INEX samlingen er allerede beskrevet i et tidligere kapittel. Mer utfyllende informasjon om den kan man også inne i [IRX] og i [WOR]. I korte trekk består samlingen av ca 12000 vitenskapelige artikler fra IEEE, formatert i XML, med felles DTD for hele samlingen. I sitt originale oppsett er fliene organisert som på Figur 2 når lagt inn i systemet.



INEX dokumentsamling slik den er strukturert i et filsystem. DTD'er for hele samlingen ligger i en egen mappe. XML dokumentene ligger organisert etter årganger for hvert enkelt tidsskrift. For hver årgang er det en volume.xml-fil med entitet-referanser til artikkel-filene a001.ml – a00X.xml, samt til DTD-filene.

Figur 2

Filsystemet som vist over førte til en del kompleksitet og utfordringer når det skulle inn i en dokumentdatabase. Det er i det hele tatt flere momenter som taler i mot å beholde denne strukturen når dokumentsamlingen skal legges inn i dokumentdatabasen. Ting som kan nevnes er:

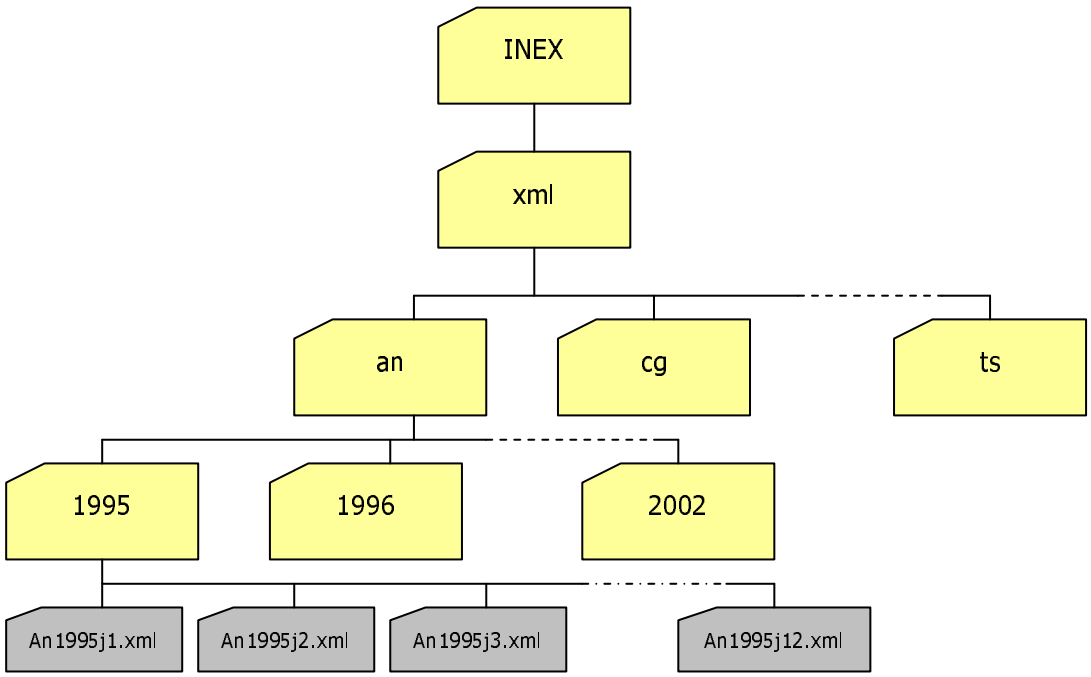
- Absolutte og relative referanser. Når `volume.xml`-filene skal parses i Java så ble det problematisk å finne igjen dtd- og artikkel-filer.
- Store filer. Når man først fikk parset og bygd `volume.xml`-filene til fullstendige xml-dokumenter i Java, så ble disse dokumentene på ~2 MB (noen større), hvilket var lite ideelt med tanke på ytelse og minnebruk i Java.

- Berkeley DB XML, og databaser generelt, er heller ikke særlig glad i store dokumenter, og vil yte langt fra optimalt hvis INEX dokumentene lagres med den opprinnelige strukturen.
- Brukere foretrekker å forholde seg til dokumenter på et lavere nivå. Volume.xml-filene vil for en bruker bli svært store å forholde seg til som en enkelt informasjonsenhet.
- Den originale INEX strukturen er generelt lite hensiktsmessig for søking.

Den originale INEX dokumentstrukturen er egentlig lite egnet for søking, og dens egentlige hensikt er uviss. For å gjøre INEX-samlingen litt bedre å jobbe med forsøkte man derfor å omstrukturere dokumentsamlingen.

4.2 Ny dokumentstruktur

I den nye dokumentstrukturen var målet å dele opp `volume.xml`-dokumentene i mindre dokumenter, og da var det i realiteten to elementer det virket fornuftig å dele opp etter: `<journal>` og `<article>`. Hver årgang består av 2 – 12 journaler (en journal utgjør et hefte, eller en enkelt utgivelse av tidsskriftet), og hver journal består av ~10 artikler. Da `<article>`-elementene i liten grad utgjør selvstendige dokumenter, ble det til slutt avgjort å dele opp dokumentene etter `<journal>`-elementene. Denne oppdelingen førte xml-filer på litt under 500 KB, og for å foreta denne oppdelingen ble det laget en liten Java-applikasjon (se filen JDOMTest.java på vedlagt CD-ROM). I den nye oppdelingen ble DTD'ene inkludert i dokumentene. Ny dokumentstruktur er vist i Figur 3.



Ny dokumentstruktur med selvstendige og fullstendige journal-dokumenter, uten referanser til andre entiteter eller til DTD-dokumenter.

Figur 3

Denne nye dokumentstrukturen ble benyttet videre i utviklingsdelen av oppgaven, og viste seg langt enklere å håndtere enn den opprinnelige strukturen.

5 Design

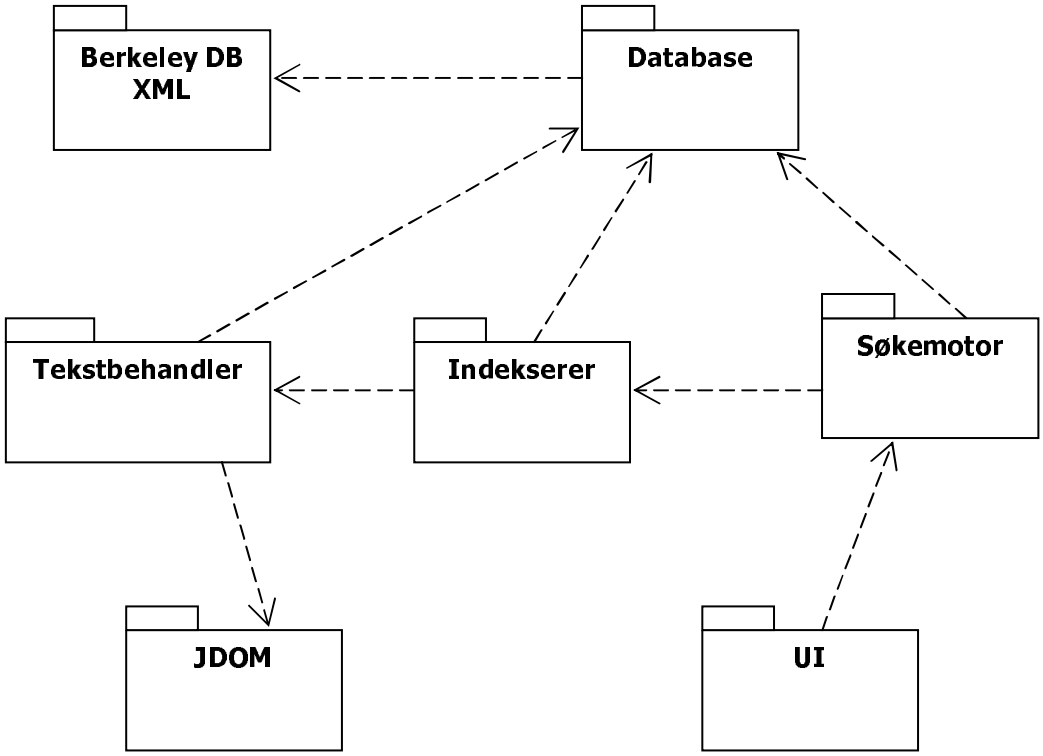
I dette kapittelet vil det presenteres et enkelt og overfladisk design av et informasjonsgjenfinningssystem for XML dokumenter. Det vil ikke inneholde mangfoldige klasse- og sekvensdiagrammer som beskriver systemet i detalj, men overordnede pakkediagram og komponentdiagrammer som viser hvilke moduler/komponenter systemet består av, samt avhengigheter og grensesnitt dem i mellom.

5.1 Generelt om systemet

Den overordnede strukturen i et slikt system trenger ikke skille seg nevneverdig fra et tradisjonelt IR-system. Selv om det er forsøkt gjort mange tilnærminger til IR-systemer for XML, har svært få av disse tilnærmingene prøvd å gjøre noe drastisk med arkitekturen i systemene. Det er heller innholdet i de enkelte modulene av systemet hvor utfordringene ligger, og dette har man også forholdt meg til i dette designet. I pakkediagrammet i kapittel 5.2 ser man et helt standard oppsett for et IR-system, hvor det er inkludert pakker for *Berkeley DB XML* og *JDOM* som benyttes til henholdsvis oppsett av databasen og til håndtering av XML. Fokuset i pakkediagrammet er å vise avhengigheter mellom de enkelte delene av systemet.

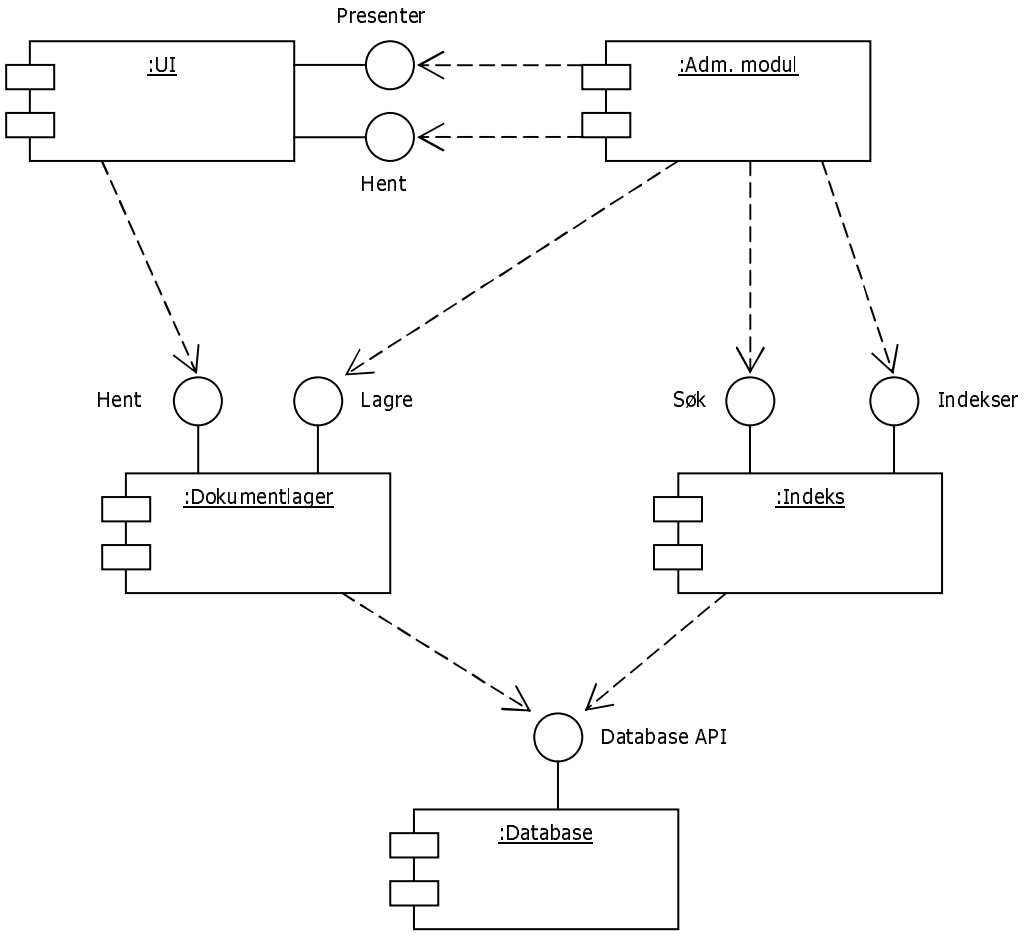
Komponentdiagrammet i kapittel 5.3 retter fokuset over på hvilke grensesnitt som komponentene tilbyr for å kommunisere med hverandre. Så lenge disse grensesnittene er konsistente, kan innholdet i hver enkelt komponent endres etter behov. Man kan for eksempel implementere ny indeksstruktur i indeksen, eller ny søkealgoritme.

5.2 Pakkediagram



Figur 4

5.3 Komponentdiagram



Figur 5

6 Konklusjoner

Hovedfokuset i denne oppgaven har ikke vært å gjøre undersøkelser som skal ende opp i slutninger og konklusjoner, men heller å belyse muligheter og valg som gjøres i utviklingen av IR-systemer for XML dokumenter, og hvilke følger og konsekvenser slike valg har. Allikevel er det mulig å si litt konklusive ting rundt noen aspekter ved prosjektet. Et av målene har jo for eksempel også vært å se på hva som skiller tradisjonelle IR-systemer fra systemer tilpasset XML dokumenter og annen strukturert informasjon.

Når det gjelder sistnevnte mål, så kan man generelt si at forskjellen er liten, men vesentlig. Overordnet struktur i et IR-system for XML dokumenter skiller seg lite eller ingenting fra tradisjonelle IR-systemer, man har de samme modulene og komponentene (se kapittel 5), og de samme avhengighetene og grensesnittene mellom disse. Forskjellen ligger i detaljene, det vil si implementeringen av de enkelte komponentene. Det er på dette nivået man faktisk håndterer og bearbeider strukturen i XML dokumentene, dermed er det her forandringene må ligge. Systemet må kunne lagre XML dokumenter, det må håndtere parsing av XML, og indeksering og søking må tilpasses XML strukturen.

Denne situasjonen kan endre seg i fremtiden, kanskje vil forskning og studier av strukturert informasjon på høyere nivåer (metadata, semantikk, ontologi) føre til nye konklusjoner, men slik situasjonen foreligger i dag innenfor IR i XML, så er det på implementasjonsnivå at forskjellene foreligger i forhold til tradisjonell IR.

7 Erfaringer

Under utviklingen av systemet ble det gjort noen erfaringer med Berkeley DB som det er verdt å merke seg:

- Når man legger inn XML dokumenter i en BDB XML database, gjøres det i form av å opprette `XmlDocument` objekter som siden puttes i containere, `XmlContainer`. Når denne overføringen til containeren skjer, blir XML dokumentene *validert*. Er ikke dokumentene gyldige, blir det kastet `Exception` og dokumentet plasseres ikke i containeren, og dermed ikke i databasen. Dette var en medvirkende årsak til at INEX samlingen ble omorganisert for bruk i denne oppgaven.
- Den utgaven av BDB som man til slutt valgte å bruke er programmert i C++. Når man programmerte opp mot denne i Java, fikk man av og til følgende feilmelding under kjøring: "Unhandled exception from C++ API". Etter litt feilsøking på google fant man at en mulig årsak til dette var mangel på sletting av ubrukte objekter i C++, og en del problemer løste seg ved å kalle `delete()` på objekter man ikke lenger hadde bruk for. Denne upresise feilmeldingen var uansett vanskelig å forholde seg til.

8 Fremtidig arbeid

Som nevnt flere ganger gjennom dette dokumentet, har ikke hensikten med hovedoppgaven vært å produsere et ideelt IR system for XML dokumenter. Målet har vært å se på utviklingen av et slikt system, komme seg gjennom så mye som mulig av denne utviklingsprosessen, for å gjøre seg erfaringer og rette fokus mot hva som er viktig å se på videre. I dette kapittelet legges fokuset på dette aspektet.

8.1 Indeksering

Indeksering er absolutt et område som opplever nye muligheter gjennom strukturerte dokumenter. Som nevnt tidligere i oppgaven er indeksen nært knyttet opp mot søkemotoren i IR-systemet, og det er vanskelig å se på indeksering uten å samtidig ta i betraktning søking og søkealgoritmer. Skal man allikevel forsøke, så er det mest interessant å se på hvordan man best mulig kan representere dokumentstrukturen i indeksen og utnytte denne for å få en mest mulig rask og effektiv indeks.

XML dokumenter inneholder en mengde større og mindre elementer som kan indekseres, og heri ligger det en utfordring i å finne ut hva av dette innholdet som skal indekseres. I kapittel 2.3.3 nevnes noen strategier for indeksering av strukturerte dokumenter, men mulighetene er flere, og dette er absolutt noe å se nærmere på i fremtidig arbeid med å forbedre IR i XML.

8.2 Søketeknikker

I tradisjonell IR fins det etter hvert et stort mangfold av søkealgoritmer og teknikker, men vektormodellen har hatt klart størst suksess, og få andre modeller har kunnet måle seg i forhold til denne. IR i XML er såpass nytt at man har til gode å se en modell som skiller seg klart ut her på samme måte. Ved å se på resultatene fra de enkelte INEX-årgangene [INE] kan man se et stort spekter av metoder og teknikker som har blitt prøvd. Både forsøk på å overføre vellykkede klassiske IR-modeller til bruk i XML (som for eksempel modifiserte utgaver av nevnte vektormodell), samt nye og uprøvde innfallsvinkler på IR-problemet som de nye dokumentformatene åpner for.

Noe av motivasjonen for å prøve å overføre eksisterende IR-teknikker til bruk for IR i XML handler om gjenbruk: mye tid og penger kan bli spart hvis man finner måter å utvide eksisterende IR-systemer på som gjør at de også støtter søking i XML og annen strukturert informasjon. [MAS] er allerede nevnt tidligere, og er en måte å utvide vektormodellen på til å støtte søking i XML.

Det er nok av metoder å teknikker å se nærmere på og jobbe videre med, og fremtiden vil vise om en teknikk vokser frem som den mest brukte, eller om vi får et kompromiss med mange forskjellige løsninger som blir brukt i søkemotorer for XML dokumenter. Uansett hva utfallet blir så vil nok INEX ha en viktig finger med i spillet.

8.3 Brukergrensesnitt

Informasjonsgjenfinning i XML dokumenter og i strukturert informasjon generelt gir også nye muligheter innen interaksjon med brukerne av systemene i forhold til tradisjonell IR. Dette er også noe som INEX [INE] har valgt å rette litt av fokuset på gjennom sitt *Interactive Track* som ble startet opp i 2004. Begge aspektene ved brukerinteraksjonen, både spørringer og presentasjon av tilhørende søkeresultater, er interessante å se videre på.

Når det gjelder spørringer, så er det de av formatet *Content and Structure* som er mest interessant å se videre på. I denne formen for spørringer har brukeren muligheten til å angi strukturelle argumenter i spørringen, og på det viset avgrense og presisere søket. Det fins flere måter å angi slike argumenter på, man kan velge å kun søke i navngitte strukturelle elementer, eller man kan velge å kun søke i

elementer som ligger innenfor et gitt størrelsesintervall. Mulighetene er mange. En spennende utfordring på dette området kan være hvordan man skal tilby søking med strukturelle argumenter for den ufaglærte, hverdagslige bruker av IR-systemer. Mange brukere av IR-systemer tar seg ikke tid til, eller har ikke kompetanse til, å sette seg inn i kompliserte spørringsformuleringer, og kunne hatt stort utbytte av grensesnitt som enkelt og intuitivt tilbød begrensning av søk gjennom bruk av strukturelle argumenter.

Når det kommer til presentering av søkeresultater er det også en del nye muligheter i XML. Som eksempel kan trekkes frem det systemet som ble brukt for å teste brukerinteraksjon i nevnte Interactive Track i fjorårets utgave av INEX. Her ble søkeresultatene presentert slik at når man klikket på linkene til de informasjonselementene som ble returnert som søketreff, så fikk man opp innholdet i dette elementet, samt et dokument-tre til venstre der man kjapt kunne klikke seg fram til andre deler i samme dokument, samtidig som det gav en kjapp oversikt over dokumentstrukturen. Dette er en mulighet å utnytte XML-strukturen på, og dette er et interessant aspekt å studere med tanke på å gi brukerne bedre utnytte av den informasjonen som søkemotorene returnerer.

8.4 Ontologi

Forskning og studier innen ontologi og *Semantisk Web* [SWB] for informasjonsbehandling er et svært spennende aspekt ved strukturert informasjonsbehandling. Med Semantisk Web søker man å dele informasjon og data på tvers av applikasjons-, bedrifts- og samfunnsgrenser ved å gi informasjonen veldefinert mening og semantikk, og på dette viset skape bedre samarbeid både mennesker og datamaskiner imellom. Denne utviklingen er nært knyttet opp mot utviklingen innen IR i XML og annen strukturert informasjon, og utgjør et svært spennende potensial for fremtidig håndtering av digital informasjon.

Referanser

- [BAE] Ricardo Baeza-Yates, Berthier Ribeiro-Neto – *Modern Information Retrieval* (Addison Wesley) 1999
- [ABI] Serge Abiteboul, Peter Bunemann, Dan Suciu - *Data On The Web – From Relations to Semistructured Data and XML* (Morgan Kaufman Publishers) 2003
- [ROB] S. E. Robertson, K. Sparck Jones – *Relevance weighting of search terms* (Journal of the American Society for Information Sciences, 27(3):129-146) 1976
- [SAL] G. Salton, M. E. Lesk – *Computer evaluation of indexing and text processing* (Journal of the ACM, 15(1):8-36) Januar 1968
- [GSA] G. Salton – *The SMART Retrieval System – Experiments in Automatic Document Processing* (Prentice Hall Inc., Englewood Cliffs, NJ) 1971
- [TRE] *Text Retrieval Conference* - http://trec.nist.gov/
- [INE] *Initiative for the Evaluation of XML retrieval* - inex.is.informatik.uni-duisburg.de/
- [I03] *INEX 2003 Workshop Proceedings*
- [IEE] *Institute of Electrical and Electronics Engineers* – www.ieee.org
- [W3C] *World Wide Web Consortium* – www.w3c.org
- [POR] *Porter Stemming Algorithm* - http://www.tartarus.org/~martin/PorterStemmer/
- [MAS] Yosi Mass, Matan Mandelbrod – *Retrieving the most relevant XML Components*
- [XIN] *Apache Xindice* - xml.apache.org/xindice/
- [XST] *eXist* - exist.sourceforge.net/
- [SLC] *Sleepycat Software* - www.sleepycat.com
- [BDB] *Berkeley DB Technical Features* - www.sleepycat.com/products/featurelist.shtml
- [XQR] *XQuery* - www.w3.org/XML/Query
- [IRX] Kent Rune Klungerbo - *Informasjonsgjenfinning i XML dokumenter*
- [JDO] *JDOM* – www.jdom.org
- [ITR] *INEX Interactive Track* - http://inex.is.informatik.uni-duisburg.de:2004/tracks/int/
- [SWB *W3C Semantic Web* - http://www.w3.org/2001/sw/