NTNU
Det skapende universitet

Automatisk temainndeling

Toril Ormberg Reite

Master i datateknikk
Oppgaven levert: Juli 2006
Hovedveileder: Herindrasana Ramampiaro, IDI
Biveileder(e): Jeanine Lilleng, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Målsetningen for mitt diplomarbeid blir å finne en metode som foreslår sannsynlige tema/temaer i dokument, basert på tekstlig innhold.

Først vil det bli gjennomført et litteraturstudium som har som mål å finne metoder for å identifisere tema i tekst. Det antas at disse metodene vil trenge språkressurser som ontologier thesauri o.s.v. Dette vil også bli beskrevet i litteraturstudien.

Deretter vil det bli laget en prototyp for å undersøke de mest lovende metodene i praksis.
En utredning/diskusjon basert på resultatet/funnene er en del av oppgaven.

Oppgaven gitt: 20. januar 2006
Hovedveileder: Herindrasana Ramampiaro, IDI

Sammendrag

I dagens informasjonssamfunn har man enkel tilgang til store mengder informasjon, dette fører ofte til at man finner mye mer informasjon enn man trenger og det blir vanskelig å finne det man leter etter. I min masteroppgave skal jeg prøve å finne en metode som automatisk angir tema til tekster. Ved å få angitt et tema, er det forhåpentlig enklere å se om teksten innholder relevant informasjon.

Utgangspunktet for oppgaven var en idé om at det er mulig å finne tema for en tekst ved å bruke tittelen og sammendrag (abstract) som grunnlag. Dette testes ved å dele en samling dokumenter opp i flere deler og trene en del ved å la forhåndsbestemte tema for denne delen bli satt som utgangspunkt for sammenligninger med resten av samlingen.

For å løse denne oppgaven har jeg gjennomført en litteraturstudie, tatt i bruk lovende teknologier for høsting av metadata og laget en prototyp som tester hvor godt metoden fungerer i praksis.

For å høste metadata ble Open Archives Initiative's standard for høsting av metadata valgt som løsning for høsting av data. Arc, en åpen kildekode programvare, ble brukt for selve høstingen og MySQL ble valgt som datalager for innhøstede data.

Det viste seg at metoden i de fleste tilfeller traff feil tema, sammenligner man med de fem temaene som blir foreslått er treffprosenten 11 prosent i gjennomsnitt. Dette gir for dårlig resultat til at metoden kan taes i bruk slik den er i dag. Det har underveis blitt oppdaget en mulig feilkilde i selve samlingen som er brukt for testformålet, da dokumentene i samlingen er fordelt slik at like tema ligger nært hverandre. Ved testing på et utvalg av samlingen på de 100 første dokumentene blir resultatet vesentlig bedre enn for hele samlingen. Temafordelingen i denne delen av samlingen er også mer jevnt fordelt enn resten av samlingen, dette indikerer at det er samlingen som er problemet og ikke metoden. En bedre kontroll med samlingen vil derfor antagelig gi bedre treff på riktig tema enn det prototypen min klarer, på det nåværende tidspunkt.

Jeg konkluderte med at metoden mest sannsynlig virker og at den kan brukes til å angi tema for tekstlige dokumenter. Det må imidlertid testes på blant annet om resultatet blir bedre dersom man stokker dokumentene, eller bruker en annen dokumentsamling. Dersom resultatet blir at treffprosenten nærmer seg hundre prosent er det etter min mening mulig å bruke metoden i stor skala på internett for å angi tema til dokumenter.

Forord

Denne rapporten er en del av min masteroppgave i datateknikk. Den er skrevet ved institutt for datateknikk ved NTNU. Jeg vil takke hjelpeveileder Jeanine Lilleng for gode råd og oppfølging underveis, Tore Kolltveit for korrekturlesing av rapporten, teknisk hjelp ved implementering og moralsk støtte, Ingri Ormberg Reite for å bruke ferien sin til å passe Tobias, Tobias Reite Kolltveit, du er sola mi.

Innhold

Figurliste

Tabelliste

1 Innledning

Når man skal finne informasjon om et tema, er det mange muligheter blant digitale ressurser, i form av blant annet tekstlige dokumenter, lyd, video og bilder. Internett, og søkemotorer kan på under et sekund gi tilgang til flere tusen ressurser om et tema. Problemet er å finne noe som er relevant nok i forhold til tema man søker.

I følge webstedet DomainTools er det nå over 70 millioner registrerte kommersielle domener og 2,5 milliarder registrerte landsdomener på ca 250 land. Selv om hvert domene bare skulle inneholde informasjon tilsvarende en A4 side blir det en enorm informasjonsmasse. Dagens søkemotorer blir stadig forbedret, men man får fortsatt for mange ”relevante” treff. Hvordan man begrenser søket og gir tilleggsinformasjon til søkemotoren om kontekst til tema, er en kunnskap som uerfarne brukere ofte mangler. Selv om man innehar denne kunnskapen kan man ikke alltid vite nøyaktig hva man er ute etter, da blir treffene mange, og de færreste undersøker mer enn de dokumentene som kommer opp på første side. Når i tillegg søketermen er tvetydig, fordi ord kan ha flere semantiske betydninger, vil resultatet bli at mange av treffene omhandler uinteressante tema.

Et annet problem er at treffene ofte bare viser tittelen på ressursen, da er det vanskelig å si om ressursen er relevant, uten å undersøke den. Hadde litt mer informasjon kommet frem, som for eksempel et automatisk generert tema for hvert treff, ville man hatt et bedre utgangspunkt for å undersøke ressursen eller forkaste den [1].

Internett eller World Wide Web består for det meste av et virvar av dokumenter, skrevet i det ustrukturerte og lite dynamiske språket html. Html er et lavterskelnivå språk og det finnes mange applikasjoner som gjør det enkelt for brukeren å lage websider som både ser bra ut og som inneholder mer eller mindre nyttig informasjon. Problemet med html-dokumenter er at de er lette for mennesker å forstå, men umulig for maskiner å forstå fordi de ikke er maskinlesbare.

Semantisk web er en utvidelse av internett, der man prøver å gjøre innholdet på weben semantisk forståelig både for mennesker og maskiner. Dette vil i praksis si, at datamaskinene skjønner semantisk betydning av ord og ikke bare sammenligner bokstav for bokstav. I prosjektet med semantisk web som World Wide Web Consortium (W3C) har utviklet er målet å gjøre weben i seg selv i stand til å forstå semantisk betydning av ord. For at Semantisk web skal fungere må dokumentene være skrevet på en logisk måte og en forhåndsbestemt formatering må være fulgt, eller man må ha en database som lagrer denne utvidede informasjonen for hvert dokument. Dette krever mye arbeid for webutvikleren, og en god del lagringsplass. I praksis er semantisk web ikke veldig utbredt, antagelig på grunn av all jobben med å lage dokumentene slik at all tilleggsinformasjon som kreves er med [2].

En mulig løsning for å gjøre det lettere å finne fram på nettet, uten å implementere all ektra logikk og informasjon som semantisk web krever, er kanskje å få til noe som ligger mellom

vanlig html og semantisk web. En metode som kan angi tema til ressurser ut i fra innhold i ressursen, uten noe ekstraarbeid for webutvikler, ville antagelig være en god løsning.

I min masteroppgave vil jeg derfor prøve å finne en metode som finner tema automatisk for tekstlige dokumenter uten å måtte legge til tilleggsinformasjon til dokumenter eller på annen måte gjøre et arbeid opp mot selve dokumentene.

I neste kapittel blir oppgaven introdusert, utfordringer blir klargjort og mulige løsninger presentert. Kapittel 2 beskriver de viktigste metodene og teknologiene jeg skal bruke. I kapittel 3 blir implementasjonen av prototypen for metoden beskrevet, fra valg av applikasjoner i åpen kildekode til hvordan metoden skal implementeres og teorien bak. I kapittel 4 blir resultatene fra kjøring av prototypen presentert. I kapittel 5 blir resultatene i fra kapittel 4 og arbeid som er utført diskutert og evaluert. Og til slutt i kapittel 5 blir konklusjonen av metoden gitt og videre arbeid presentert.

2 Introduksjon til oppgaven

I min masteroppgave skal jeg prøve å finne en metode som kan anslå tema til dokumenter automatisk. Temaet som skal finnes baseres på tekstlig innhold i dokumentene. Dette kapittelet presenterer forskjellige måter å finne temaet til tekster automatisk og forklare hvilke metoder og løsninger jeg valgte.

2.1 Kategorisering av dokumenter

Å kategorisere dokumenter er ikke et nytt fagfelt, dette har blitt gjort lenge før datamaskiner gjorde det mulig å lagre dokumenter digitalt. Det er først de ti siste årene at det er blitt en nødvendighet å få sortert dokumenter på tema/kategori automatisk. Som nevnt i innledningen er det utbredelsen av internett som har ført til at informasjonsmassen har økt så mye [1].

For å kategorisere dokumenter finnes det mange måter å gå frem for å gjøre dette, jeg vil nå vise hva man må tenke gjennom før man bestemmer seg for hvilke metoder og teknikker man tar i bruk for kategorisering.

Først tar jeg for meg kategorisettet. Hvordan vet man hvilke kategorier man skal sortere etter? Er kategoriene satt på forhånd? Må man finne kategoriene samtidig som man innordner dokumentene under riktig kategori?

Dersom kategoriene ikke er bestemt på forhånd, skal man da sette seg ned å lese gjennom hvert dokument og bestemme kategorien manuelt? Eller skal en maskin gjøre dette? Dersom en maskin skal gjøre det og hvor riktig blir bestemmelsen da? Skal man sortere dokumentene under kategorier etter hvert som man finner kategorier eller skal man først finne settet med kategorier og så sortere dokumentene? Dersom kategoriene er gitt, er dokumentene også sortert under dem? Kan man bruke sorterte dokumenter for å sammenligne med usorterte dokumenter?

Når man sorterer dokumenter, hvordan skal man da bruke innholdet i dokumentet for å sette kategori på et dokument, skal man bruke hele dokumentet eller skal man bare bruke deler av dokumentet, skal man bruke informasjon om dokumentet som ikke er selve dokumentet, såkalt metadata[1]?

Tabell 2-I foregriper noen av løsningene, men den viser hvordan man kan kategorisere ut i fra et helt dokument eller deler av det, og hvilke teknikker som kan brukes for kategoriseringen.

[1] Se neste kapittel for mer beskrivelse av metadata.

Min oppgave er som kjent å finne en løsning som gjør det mulig å kategorisere dokumenter uten at man må gjøre noe med dokumentene. Det er naturlig å først bestemme seg for hvor mye av et dokument man skal bruke for kategoriseringen. Det kan være at man vil ta utgangspunkt i hele dokumentet, eller deler av dokumentet, eller annen informasjon man måtte ha om dokumentene.

Når dette er bestemt vil det være naturlig å finne ut om det eksisterer et sett kategorier for dokumentene på forhånd. Det vil si at det finnes informasjon, enten i dokumentet eller utenfor dokumentet om hvilken kategori det tilhører eller hva tema det omhandler. Deretter bestemmer man seg for hvilke teknikker og metoder man velger for å gjøre kategoriseringen, det er her teknikkene i
Tabell 2-I kommer inn i bildet.

Dokument Kategorisering basert på:	**Kategorier** Oppgitt kategorisett:	Ukjent kategorisett:
Fulltekst	Maskinlæring, clustering,	Clustering
Metadata	Clustering	Clustering

Tabell 2-I Metoder for kategorisering av dokument

I min oppgave er det ikke åpenbart hva som var løsningen på alle spørsmålene jeg har stilt til nå. Men jeg har som målsetning at metoden skal være enkel å implementere og ikke kreve store ressurser.

Sebastini beskriver en metode for kategorisering som tar utgangspunkt i hele teksten, der man lar maskinen lære seg å forstå innholdet i teksten ved bruk av maskinlæring. Maskinlæring vil enkelt si at maskinene får implementert kunstig intelligens slik at de er i stand til å lære. Metoden krever en komplisert algoritme og mye arbeid for at maskinen skal forstå teksten, den er derfor ikke særlig aktuell for min oppgave [3].

Den andre måten som er beskrevet i
Tabell 2-I er clustering, eller klynging på norsk. Siden clustering er et innarbeidet begrep i litteraturen og den norske oversettelsen ikke er så god vil jeg fortsette å bruke clustering. Clustering vil i følge Baeze-Yates si å samle relevante dokumenter i en klynge, det finnes mange måter og algoritmer for hvordan man kan utføre selve clusteringen. Man kan også her ta utgangspunkt i hele teksten til dokumentet, eller man kan bruke deler av dokumentet eller bare metadataene til dokumentet, siden jeg ville prøve å få til en metode som ikke krevde store ressurser er det mest aktuelt for meg å prøve å ta utgangspunkt i deler av dokumentet eller metadataene til et dokument. [4]

Et annet spørsmål som ikke er blitt stilt ennå er hvilke dokumenter skal metoden testes på? Hvor får man tak i en samling med dokumenter som gjør det mulig å teste om metoden finner riktig tema? Først må man bestemme seg for om dokumentene skal ha et fastsatt kategorisett på forhånd. For å teste om metoden finner riktig tema, er det naturlig å finne en samling dokumenter der kategoriene er fastsatt på forhånd. Det har liten hensikt å finne dokumenter tilfeldig på nettet ved å bruke søkemotorer eller lignende tjenester for dette formålet. Digitale bibliotek[2] kan være en løsning på dette ved at man kan hente hele samlinger direkte i fra et digitalt bibliotek. Problemet er at man ofte må laste ned dokumentene i fulltekst og gjerne i formater som gjør det vanskelig å skille de forskjellige delene i et dokument fra hverandre. Er

[2] Digitale bibliotek er i følge Arms er et digitalt bibliotek et bibliotek som tilbyr ressursene sine i et maskinlesbart format. Dette gjør det mulig å søke og sortere innholdet man er ute etter. Kilde: 5. W.Y. Arms, *Automated Digital Libraries.* D-Lib Magazine, 2000. **6**(7/8).

det mulig å få samlet dokumenter uten at man må laste ned alt innholdet i dokumentet? Open Archives Initiative (OAI) har laget en standard som gjør det mulig å implementere en applikasjon som henter dokumenter i form av metadata, en slik applikasjon kalles en høster. Dette virker som en lovende metode for å samle dokumenter, som ikke krever store ressurser for nedlasting og lagring. Jeg har derfor valgt å bruke OAIs standard for å samle inn informasjonen jeg trenger om dokumentene. Valget av OAIs standard setter en del begrensninger for hvilke metoder jeg kan bruke for å finne tema automatisk, fordi det bare er metadata man får høstet. Siden metoden skal være så enkel så mulig er dette en fordel, da ressursbruken på lasting og lagring begrenser seg selv. [6]

2.2 Sammenligning av dokumenter mot tema

Valget av OAI gjør at dokumentene blir representert av sine metadata. Neste utfordring blir å finne ut hvordan man skal finne tema automatisk for dokumentene, og hvordan teste om tema som er funnet er riktig.

Når man bruker Open Archieves Initiative's standard får man dokumentene presentert i et spesielt format, kalt Dublin Core (se kapittel 3.1.4). Dette formatet presenterer hvert dokument med en unik identitet og andre elementer som tittel, beskrivelse, forfatter og dato. En må først finne ut hvordan man skal få representert dokumentet på en slik måte at de er målbare for å sammenligne med andre dokumenter og tema/kategori. Dette kan gjøres på flere måter, man kan i følge Baeza-Yates representere et dokument slik at det kan sammenlignes med andre dokumenter blant annet ved å bruke en boolsk metode, der bestemmes det bastant om et dokument er relevant eller ikke ut i fra semantisk innhold. Vektormodell, en annen metode, presenterer dokumentet som en vektor og sammenligningen med andre dokument skjer blant annet ved at man finner vinkelen mellom vektorene for dokumentene, dette gjør at man får et målbart tall for hvor nære to dokumenter ligger hverandre. Metoden er veldig enkel, og tar ikke hensyn til semantisk innhold i dokumentene. Den er i følge Baeza-Yates en meget velkjent og en klassisk metode innenfor informasjons gjenfinning fagfeltet, i tillegg passer det godt å bruke vektorer når man ikke har hele teksten i dokumentet å ta hensyn til. Av disse grunnene har jeg derfor valgt å bruke vektormetoden for å presentere tekstlige dokumenter [4].

Neste steg blir å finne ut hvor mye av metadataene som skal representere dokumentet, skal man slå sammen alle metadataene å lage en vektor av det? Det virker på meg som en lite hensiktsmessig måte å gjøre det på siden mange av metadata-feltene inneholder formater som dato og id-nummer. Dette gjør at teksten det skal lages vektorer av, består av mange ulike formateringer som ikke hører hjemme i en vanlig ordliste. Det er viktig å få representert dokumentene i klartekst uten for mange forstyrrende elementer. Jeg har derfor valgt å la metadatafeltet for ”title” og ”description” være ordene som representerer innholdet i et dokument. Dette fordi disse feltene også finnes i fulltekst utgaven av dokumentet. Vel og merke dersom ”description” er brukt til sammendraget for dokumentet. I kapittel 4.2 blir en detaljert beskrivelse av hvordan man finner vektor for dokument beskrevet.

Dokumentene er representert ved vektorer, da må også temaene være representert ved vektorer, ellers går det ikke an å sammenligne et dokument og et tema. En enkel måte å representere et tema for dokument som vektor er å ta teksten i subject-feltet til de innhøstede dokumentene å lage en vektor av dette. Men hvordan vil sammenligningen mellom dokument

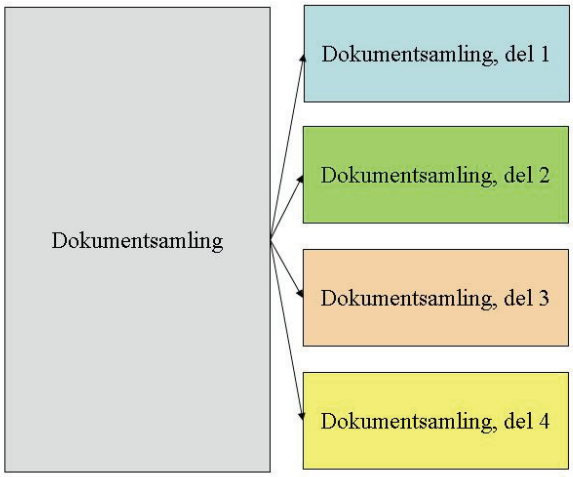
og tema bli? Det vil antagelig bli så store avvik mellom dokument og tema at ingen av temaene vil passe for noen av dokumentene, dette fordi teksten i dokumentet er vesentlig annerledes enn teksten i for temaet. Hvordan kan man da få temaet representert som en vektor på en slik måte at dokumentene blir sammenlignbare med tema?

En måte å løse dette på er å la tema-vektoren bli presentert som et gjennomsnitt av dokumenter som har dette tema. Jeg har valgt å gjøre det på denne måten fordi man da får vektorer for tema som kan sammenlignes med dokumentene. I kapittel 4.2.3 blir det beskrevet i mer detalj hvordan vektor for tema blir representert.

2.3 Testing av metoden

Dokumentene skal sammenlignes med temaene ved å bruke vektorer for å representere dem, men hvordan teste om et dokument har fått riktig tema "gjettet" i forhold til reelt tema?

For å få testet metoden er det naturlig å dele opp samlingen i flere deler. Se Figur 1 for eksempel på hvordan man kan dele opp samlingen. Ved å bruke første del av samlingen som treningssett, ved å gi temaene for denne delen av samlingen vektorer som beskrives i kapittel 4.2 kan man sammenligne resten av dokumentene i samlingen opp mot denne treningsdelen.



Figur 1 Dokumentsaling
Figuren viser hvordan dokumentsamlingen er delt inn i flere deler.

Dersom dokumentsamlingen er liten vil det være naturlig å la alle delene av samlingen være treningsdel på omgang. Dette for å kunne teste flest mulig dokumenter opp mot ulike sett av tema. I tillegg til å bestemme hvordan testsettene og treningssettene skal se ut må man finne ut hvordan man finner ut at riktig tema blir funnet. Det er gunstig at hvert dokument blir

representert slik at de tema som dokumentet faktisk innehar er tilgjengelig. Da kan man sammenligne om funnet tema stemmer med reelt tema for dokumentet. Det må bestemmes hva som skal regnes som et treff, er det kun når funnet tema stemmer fullstendig overens med de tema som dokumentet har? Eller kan man regne et treff dersom temaet er et av flere like tema? Det er flere måter å beregne et treff på og kapittel 4, viser resultatene ved testing av metoden og beskriver hvordan jeg har definert treff.

I de to neste kapitlene blir metoder i dette kapittelet forklart mer i detalj og faktisk teknologi og implementering av prototyp skildret.

3 Metoder og teknologi

For å kunne lage metoden som skal sette automatisk tema på tekster har jeg brukt en del metoder og teknologier, noen av dem er allerede nevnt i slutten på forrige kapittel. I dette kapittelet vil jeg beskrive metoder og teknologier som jeg har brukt for å løse oppgaven.

3.1 Høsting av metadata

For å få tak i data som kunne brukes for å teste metoden har jeg valgt å høste metadata. For å gi leseren et innblikk i hva metadata er har jeg valgt å først utdype hva jeg mener med metadata.

3.1.1 Metadata

Metadata blir veldig ofte omtalt som data om data, dette er en riktig beskrivelse men noe upresis. Metadata har mange ulike definisjoner ut i fra hvilke sammenheng begrepet brukes i og i følge Gilliand kan begrepet defineres til at de beskriver alle kjennetegn ved et informasjonsobjekt. Et informasjonsobjekt defineres som et digitalt element eller en gruppe av elementer som uavhengig av form kan bli adressert og manipulert som et enkelt objekt av en datamaskin [7]. Metadata gir informasjon om et informasjonsobjekt. Informasjonsobjekter har alltid følgende tre egenskaper:

- Innhold: Hva objektet inneholder er en vesentlig del av objektet.
- Kontekst: I hvilken kontekst objektet står, beskriver blant annet hva, hvem og hvordan objektet oppstod.
- Struktur: Sier noe om det formelle settet med assosiasjoner til objektet, dette gjelder både innenfor objektet i seg selv og mellom andre objekt.

Informasjonsobjekter blir ofte oppbevart i digitale bibliotek, der blir de lagret digitalt. Mange av systemene for å katalogisere informasjon som blir brukt i vanlige biblioteker blir også brukt av digitale biblioteker. Derfor er det blitt utviklet mange ulike standarder og metoder for å lage metadata om informasjonsobjekter slik at de blir lett tilgjengelige. Slike standarder er for eksempel Dublin Core og Marc. I mitt arbeid blir Dublin Core brukt som standard for informasjonsobjektene og denne standarden er beskrevet i

Dublin Core kapittelet senere.

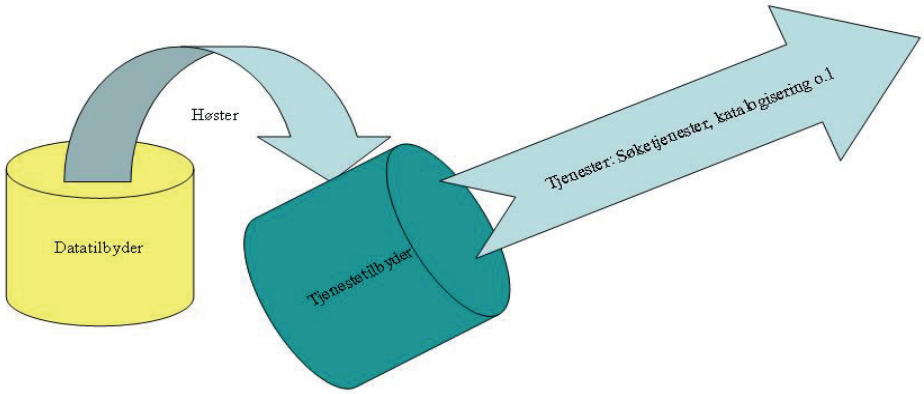
Metadata er altså data om data, og i mitt arbeid trenger jeg ikke mer informasjon om objektene eller dokumentene jeg jobber med enn det som Dublin Core (DC) kan beskrive. Dette fordi jeg vil bruke tittel, en beskrivelse i form av ”abstract” og ”subject” feltet til DC for å representere dokumentene jeg skal finne tema for. Dette gjør at man slipper å ta hensyn til hele teksten i dokumentet og man slipper å bruke komplisert maskinlæring[3].

Høsting av metadata vil si å samle inn metadata om informasjonsobjekter. Dette er en effektiv måte å samle inn store informasjonsmengder uten å måtte laste ned store filer. Høsting foregår først og fremst ved å bruke Open archieve Initiatives protocol for Metadata Harvesting, heretter omtalt som OAI-PMH. Dette er en protokoll som tilbyr et rammeverk for metadatahøsting som er plattformuavhengig og i følge Sompel et. al skal det være en lav terskel for å implementere standarden [8]. I neste avsnitt blir standarden beskrevet mer i detalj.

[3] Maskinlæring er å lære opp datamaskinen til å lære seg nye ting selv, også kalt kunstig intelligens. Se artikkel til Sebastini.

3.1.2 OAI-PMH

OAI-PMH er et sett med regler for hvordan man høster metadata. Den kan implementeres i for eksempel Java og skal som nevnt være en lavterskel standard å ta i bruk. Det er den rådende standarden innen høsting av metadata. OAI-PMH består av to aktører en datatilbyder og en tjenestetilbyder og et sett med regler som beskriver hvordan kommunikasjonen foregår mellom disse to aktørene i form av en høster. Se Figur 2, for visuell beskrivelse av OAI-PMH's deler.



Figur 2 Arkitektur til OAI-PMH

Figuren viser de viktigste komponentene til OAI-PMH. En datatilbyder har laget dataene sine i et slikt format at det er mulig for høster applikasjoner å koble seg til og høste metadataene. Høstede data blir tatt hånd om en tjenestetilbyder som består av et datalager og ofte tilbys ulike tjenester som søking, katalogisering o.l Kilde: Solvang [9]og Sompel [8], figuren er fritt tegnet etter informasjon på [10]

OAI-PMH høster kun metadata og ikke selve objektene, metadataene må da være lagret av datatilbyderen i standard format. Dette formatet er Dublin Core, se kapittel om Dublin Core [8].

Tjenestetilbyderen tilbyr ofte søking og katalogisering av ressurser. Men før den kan tilby noen tjenester må den høste metadataene fra datatilbyderen, et klientprogram som enten kan være innbygget i tjenestetilbyderen eller et eksternt program tar seg av jobben[8].

Hendelse	Funksjon
Identifiser *(Identify)*	Hente beskrivelse av datalageret, standarder og protokoller som er brukt.
List opp metadataformat *(ListMetaDataformat)*	Lister opp metadataformatene, for hele lageret dersom ikke identifier er gitt.
Lister opp sett *(ListSets)*	Viser settstrukturen til datatilbyderen.
List opp identifikatorer	Lister opp identifikatorene til alle elementene datatilbyderen tilbyr.
Hent element *(Get record)*	Høster et element, angitt med en unik identifikator. Et element blir også ofte kalt post eller metadatapost.
List opp element *(ListRecords)*	Høster metadataposter angitt av identifikator og for en bestemt periode.

Tabell 3-I Hendelser OAI-PMH

Viser hendelsene som OAI-PMH består av.

Datatilbyderen har lagret dataene på en tjener som klientprogrammet kommuniserer med under høstingen. Solvang har definert seks hendelser skjer under høstingen i Tabell 3-I vises disse hendelsene.

Når høsteren henvender seg til en datatilbyder skjer kommunikasjonen ved at høsteren sender melding om at datatilbyderen må identifisere seg, datatilbyderen sender tilbake informasjon om seg selv, så sender høsteren spørsmål om settstrukturen til datatilbyderen datatilbyder svarer ved å gi settstrukturen til tilbyderen. Så spør høsteren om listMetadataformat listIentifiers og listRecords, og datatilbyderen svarer ved å liste opp aktuelle metadataformat, identifikatorer og selve postene i XML-format [9].

Siden det bare er behov for metadata for min oppgave er OAI-PMH et bra alternativ da jeg slipper å få mye data lagret som jeg ikke har bruk for. OAI-PMH er blitt ledende standard siden den kom i 2001 og blir nå brukt av de fleste store digitale bibliotek verden over, listen over datatilbydere som er registrert på www.oai.org blir stadig lengre, og det finnes mange ferdig implementerte høstere i åpen kildekode, som er enkle å ta i bruk og som jeg vil komme tilbake senere [6].

For å kunne organisere metadata trenger man ikke bare en standard for formatet metadataene skal lagres på, man trenger også et språk som man kan kommunisere med dataene. Et slikt språk er XML, som blir beskrevet i neste underkapittel.

3.1.3 XML

Extensible Markup Language, XML, er godkjent av W3C som et markup language, eller formateringsspråk på norsk. XML beskriver data og informasjonsobjekter på mange ulike måter. Språket er likt HTML-språket ved at det inneholder tagger som beskriver noe om elementet mellom taggene. Men til forskjell fra HTML, beskriver som regel XML taggene innhold og struktur til elementene, og dette er ofte metadata. På denne måten får informasjonsobjekter lagt til metadataene i selve objektet, for en viktig egenskap ved XML er at man lagrer både selve dataene, for eksempel teksten i artikkelen, og metadataene. For å definere taggene i XML, bruker man en Document Type Definition (DTD) fil for å sette standarder for taggene, dette gjør at alle dokumentene man lager i XML, har samme definisjon for taggene. [11]

Det er mange fordeler med å ta i bruk XML, det er et språk som både er maskinlesbart og menneskelig lesbart, i Figur 3 viser et eksempel på et dokument representert i Dublin Core formatet, dette dokumentet er hentet i fra databasen hvor de høstede dokumentene ligger og en innebygd funksjon i databasen har gjort om innholdet i databasen til XML format.

Det finnes mange måter å formatere XML og siden OAI-PMH bruker Dublin Core formatet for utveksling av data, er det dette formatet som er aktuelt i mitt arbeid og jeg vil derfor beskrive innhold og struktur av dette formatet i neste kapittel.

```
<row>
       <field name="id">oai:eprints.rclis.org:1000</field>
       <field name="archive">eprints.rclis.org</field>
       <field name="datestamp">2004-02-29</field>
       <field name="sets"></field>
       <field name="title"> [The relationship between OAI-PMH and Dublin Core:
              required, recommended or other?]</field>
       <field name="creator">Lagoze, Carl</field>
       <field name="subject">L. Information technology and library
              technology.</field>
       <field name="description">Conclusions of the breakout session &amp;quot;The
              relationship between OAI-PMH and Dublin Core: required, recommended or
              other?&amp;quot;. Some issues: quality of metadata content, quality (too
              broad semantics), effort (stopping short of providing richer
              metadata).</field>
       <field name="publisher"></field>
       <field name="contributor"></field>
       <date/>
       <field name="type">Conference Paper</field>
       <field name="format">pdf
              http://eprints.rclis.org/archive/00001000/01/Group2.pdf | PPT
              http://eprints.rclis.org/archive/00001000/02/Group2.ppt</field>
       <field name="identifier">http://eprints.rclis.org/archive/00001000/</field>
       <field name="source"></field>
       <field name="language">en</field>
       <field name="relation"></field>
       <field name="coverage"></field>
       <field name="rights"></field>
       <field name="discovery">2004-01-01</field>
       <field name="groupdate">2004</field>
       <field name="cache">N</field>
       <field name="normdiscovery">2004-01-01</field>
       <discoverygroupdate/>
<field name="status"></field>
</row>
```

Figur 3 XML-Eksempel

Figur 3 viser hvordan et element som er høstet ved bruk av OAI-PMH ser ut i XML formatet. Der "Field name" gir navnet på feltet og teksten mellom taggene gir innholdet i feltet.

3.1.4 Dublin Core

Dublin Core er en standard som brukes av mange digitale bibliotek og institusjoner verden over. Den er standardisert til NISO Standard Z39.85-2001, og den brukes til å beskrive informasjonsobjekter på en standardisert måte slik at utveksling, søking og generell gjennfinning av objektene blir enklere både for mennesker og maskiner. I tabellen under vises de 15 feltene som brukes for å beskrive dokumenter som følger Dublin Core standarden [12].

Felt	Beskrivelse
Title	Tittel til informasjonsobjektet
Creator	Den som er ansvarlig for å ha laget informasjonsobjektet, kan være en person eller institusjon
Subject	Tema, for teksten uttrykt som nøkkelord
Description	Beskrivelse av innhold, gjerne i form av abstrakt (sammendrag) eller en innhodsfortegnelse.
Publisher	Den som står som utgiver av informasjonsobjektet.
Contributor	Bidragsytere, de som har hjulpet til med utgivelse av informasjonsobjektet.
Date	Dato for utgivelse
Type	Beskriver sjanger, da helst ved å bruke en kontrollert ordliste slik at alle ressursene har konsistente beskrivelser av type. Ofte observert at beskrivelse av dokumentet som for eksempel artikkel eller presentasjon.
Format	Beskriver det fysiske formatet til objektet, for eksempel bilde, tekst, lyd og lignende.
Identifier	Unik identifikator av hvert objekt.
Source	En referanse til hvor selve objektet finnes, både fysisk og digitalt.
Language	Angir språket for selve innholdet i informasjonsobjektet, eksempel: ”en” dersom engelsk.
Relation	Referanser til relaterte ressurser.
Coverage	Kan være tidsrom objektet er innenfor, eller lokasjon.
Rights	Rettighetsinformasjon for objektet.

Tabell 3-II Dublin Core Elementer

Viser de 15 elementene som Dublin core bruker for å beskrive et informasjonsobjekt. Kilde: [12]

I tillegg til vanlig Dublin Core finnes det også en kvalifisert utgave som setter strengere krav til innholdet i feltene, for mitt arbeid hadde det antagelig vært meget gunstig å kunne ta i bruk kvalifisert Dublin Core, fordi description feltet som jeg bruker for å representere et dokument ofte har veldig forskjellig innhold, og det hadde vært gunstig om det på forhånd var bestemt at Description på forhånd var satt til å kun inneholde abstract. Et annet problem jeg oppdaget med ukvalifisert Dublin Core var at Subject feltet hadde velidg forskjellig innhold. I enkelte samlinger inneholdt dette feltet bare navnet på fagfeltet, for eksempel fysikk i alle dokumentene for hele samlingen. I andre samlinger var feltet brukt for å beskrive type dokument, om det var en artikkel eller presentasjon eller lignende. Siden OAI-PMH ikke støtter for kvalifisert Dublin Core ble jeg nødt til å forholde meg til den ukvalifiserte versjonen. Dette gjorde at arbeidet med å finne en passende testsamling tok lengre tid enn antatt, og jeg fant bare en samling som var brukbar, det hadde vært ønskelig med to eller tre samlinger for å teste metoden skikkelig.

I utgangspunktet var planen å implementere høsteren som høstet metadataene jeg trenger for å teste metoden min. Men etter å ha undersøkt på OAI's nettsted fant jeg fort ut at det ville bli veldig tidsbesparende å bruke en ferdig implementert høster, og på nettstedet til OAI er det listet opp høstere som er tilgjengelig som åpen kildekode. I neste avsnitt vil jeg presentere hva jeg fant av åpen kildekode og vise hva krav jeg har satt til dem.

3.2 Åpen kildekode

Åpen kildekode er definert av Open Source Initiative til å være programvare som er gratis å laste ned, kildekoden skal være tilgjengelig for alle og man kan legge til forandringer til programvaren som kan bli brukt videre og gjelder som en del av den originale programvaren. Linux, MySQL og Apache er kjente åpne kildekodeprogrammer [13].

Open Archieve Initiative's nettsted har en liste over åpen kildekode programmer som har implementert OAI-PMH. De fleste programmer er verktøy for å gjøre om samlinger slik at de støtter OAI-PMH, dette er ikke relevant for dette prosjektet, for mitt mål med å bruke OAI-PMH er å høste ressurser i Dublin Core formatet [6].

I hovedsak var jeg ute etter å finne et program som hadde implementert en høster. Første program på listen til Open Archieves Initiative er Arc, og det blir beskrevet som et søkegrensesnitt, med en høster og lagring av data som støter OAI-PMH. Dette virket lovende, og jeg bestemte at jeg skulle teste ut dette programmet og finne ut mer om hva som er skrevet om det [6].

Solvang har i sin masteroppgave beskrevet Arc som en søketjeneste som gjør det mulig for brukeren å søke i høstede metadata som er høstet fra mange forskjellige OAI-datatilbydere. Dersom det er mulig å bruke systemet til å høste de ressursene som er nødvendig for å teste metoden, I følge installasjonsguiden til Arc skal det være mulig å manipulere databasen og finne OAI-datatilbydere i et brukervennlig og enkelt grensesnitt.

Det ble også brukt åpne kildekodeprogrammer for håndtering av databasen. MySql ble valgt til dette da dette er et velkjent og stabilt program som håndterer databasen og spørringer mot denne på en profesjonell måte.

Siden jeg ville implementere min metode i java, var det gunstig at valgte åpne kildekode program også var implementert i Java, i tilfelle noe skulle skje og jeg trengte å tilpasse systemet til mitt formål. I tillegg var det en stor fordel at de høstede dataene var lagret i en relasjons database. Disse to fordelene var så store at jeg bestemte meg for å bruke Arc til høstejobben, og planen var at metoden min kunne implementeres i ved å bruke Java Server Pages slik som søkegrensesnittet til Arc var. Dette ble etter hvert vanskelig og dette problemet og en del andre problemer vil bli beskrevet i neste kapittel, som beskriver hele implementasjonen av metoden for automatisk tema.

4 Implementering av prototyp

Oppgaven til prototypen er å teste om det er mulig å finne tema for et tekstlig dokument automatisk, basert på tittelen og en beskrivelse, helst i form av en abstract(sammendrag), til et dokument i tekstlig form. Først vil jeg kort oppsummere krav og begrensninger som ble satt i kapittel 1 og noen mer spesifiserte krav som sier litt om hvilke metoder som er brukt for å representere et dokument tallmessig, slik at det kan sammenlignes med andre dokument og tema..

- *Høsting av metadata skal skje ved å bruke Arc's høster.*
- *Data for bruk av prototypen skal lagres i relasjonsdatabasen MySQL.*
- *JDBC vil bli brukt som tilkoblingsmetode til databasen.*
- *Prototypen implementeres i Java, og resultater fra kjøring skrives til fil.*
- *Prototypen innehar ikke brukergrensesnitt.*
- *For å representere dokumenter er vektormetoden tatt i bruk.*
- *For å finne vektor som representerer tema er k-means clustering metode brukt.*

Dette er de viktigste forutsetningene som er satt før implementeringen startet og i de neste underkapitlene vil de bli gjennomgått mer i detalj. Først blir Arc's oppbygging og virkemåte beskrevet og hvordan en komplisert innstallasjonsguide og problemene dette skapte førte til at jeg kun tok i bruk høsteren til arc og ikke hele systemet. Deretter følger en detaljert skildring av hvordan dokumenter vil bli representert som vektorer og hvordan tema til tekster blir representert som vektorer. For å si noe om resultater fra kjøring og evaluere metoden som helhet, må dokumentsamlingen fra testen beskrives denne beskrivelsen finner man etter forklaring av vektorer. Siste del av kapittelet er viet selve implementasjonen av prototypen i detalj.

4.1 Arc

Arc er som nevnt i Åpen kildekode ble Arc valgt som høster for metadataene. Arc er en pioner innen bruk av OAI-PMH og i følge Sompel er Arc mye av grunnen til at OAI-PMH er blitt så populært[8].

Arc består av tre deler:

- *Høster*
- *Søkegrensesnitt*
- *Datalager*

I utgangspunktet er det bare høsteren og datalageret som var interessant for mitt arbeid, men siden webgrensesnittet inneholdt en administrasjonsside som skulle gjøre det enklere å høste og konfigurere databasen, ble det valgt å installere hele systemet.

Databasen man velger å installere i tilknytning til arc kan man i utgangspunktet velge, fritt kravet er at det er en relasjonsdatabase som har brukerkontoer som gjør at man kan manipulere databasen [14]. Arc er av utviklerne testet med Oracle database og MySQL, siden MySQL er fritt tilgjengelig falt valget på denne.
Høsteren til Arc fungerer som beskrevet i kapittelet om OAI-PMH og siden den bare er blitt brukt til å høste metadata blir den ikke beskrevet noe mer her. Det viste seg å bli vanskelig å installere hele funksjonaliteten til Arc, så det ble til slutt tatt en beslutning om å bare bruke høsteren slik den var uten å bruke det medfølgende webgrensesnittet. Dette fungerte godt og høstingen og etterfølgende parsing av dokumenter fra OAI-elementer til forståelig Dublin Core format gikk greit. Parsing av dokumentene betyr å gjøre om de innhøstede OAI-elementene til rader i en tabell, der feltene er tilsvarende med feltene i Dublin Core formatet, se
Tabell 3-II. Det er utenfor min oppgave å beskrive denne prosessen.

4.1.1 Installasjonsprosessen

Innstalleringen av Arc virket i utgangspunktet som en triviell affære, programvaren som Arc krevde for å kjøre som Apaches TomCat, MySQL, Java og JDBC var alle teknologier jeg hadde erfaring eller kjennskap til på forhånd. Men ”readme-filen” og innstallasjonsguiden både antok at man var en ekspert og den inneholdt feil ble innstalleringen en lang og omstendelig prosess som førte til at søkegrensesnittet til Arc ble forkastet og jeg stod tilbake med høsteren, som heldigvis fungerte over all forventning. For at ikke andre som ønsker å installere Arc skal oppleve de samme problemene jeg fikk skal jeg nå beskrive prosessen og gjøre rede for feilene jeg fant i installasjons- informasjonen.

Hovedproblemet var at installasjons- guiden var skrevet for ekspert brukere, som har dyp kjennskap til Apaches TomCat som webserver og til databasen MySQL og generelt hvordan manipulere tabeller i databaser. I tillegg ble det ikke enklere av at et SQL script inneholdt store feil, som gjorde installasjon av databasen svært vanskelig.

Det tok en uke å få opp MySQL databasen, dette fordi det ikke fremgikk av installasjons-guiden til Arc om man kunne bruke siste versjon av MySQL eller ikke. Det ble derfor antatt at siste versjon var bra nok. I følge MySQL’s nettsted på denne tiden (februar 2006) var siste versjon en vanlig versjon og ikke en betaversjon, noe jeg fant ut senere. Til slutt måtte databasen installeres på nytt nå med riktig versjon: 4.0 [15].

Da databasen var oppe ble neste utfordring å få lagt inn en bruker som kunne kobles opp mot arc, innstallasjonsveiledningen, dette tok et par dager å få på plass etter mye leting på MySQL’s webside. Så skulle sql-scriptet som fulgte med Arc kjøres for å opprette databasen for datalagring fra høsteren. Scriptet inneholdt flere feil, eller mangler som gjorde at det ikke ville kjøre, først antok jeg at MySQL var installert feil og det ble en ny runde på nettsiden til MySQL, med litt hjelp fra veileder Jeanine fant vi ut at det faktisk var feil i scriptet, syntaks feil og spesielle egenskaper med MySQL gjorde at scriptet ikke ville kjøre. Etter å ha gått gjennom scriptet og rettet opp i feilene fikk jeg det til å kjøre etter ca. en uke. Det viste seg imidlertid senere etter at høsteren var installert at databasen manglet flere felt i en tabell som parsede elementer skulle til, dette førte til at parsingen ikke ville starte, men på grunn av gode feilmeldinger i parseprogrammet var det enkelt å finne ut at det var flere felt i DC-tabellen til databasen som manglet [15], [14] .

Så var det tid for å få opp webtjenesten som fulgte med Arc, dette skulle vise seg å bli vanskelig. I installasjonsguiden stod det beskrevet at så snart man hadde installert Apache TomCat, Java og databasen kunne man legge Arc-filene under tomcat og de ville da kjøre. Det som ikke stod der og som en uerfaren bruker ikke har mulighet til å vite er at man måtte bygge opp filstrukturen og filene til arc, ved å bruke administrator og manager funksjonaliteten til TomCat, igjen fikk jeg hjelp av Jeanine til dette, og websidene vistes. Problemet var at de var ubrukelige uten å få komt inn på administratorsiden for tjenesten, dette var veldig dårlig forklart i installasjonsguiden, og etter halvannen uke ga jeg opp webtjenesten og konsentrerte meg om å få høsteren til å fungere[14].

Høsteren var heldigvis enklere å få til, alt som krevdes var at passord og bruker for databasen var satt riktig i databasepoolen for høsteren. I databasen måtte det legges til noen linjer i tabellene som identifiserte datatilbyder man ønsker å høste fra og informasjon om seg selv, dette for at datatilbyderne skal kunne gjenkjenne og verifisere de som prøver å høste. Etter ca. en uke hadde jeg fått opp høsteren. Og kunne endelig begynne å høste ressurser. Dette gikk bra, men det oppstod problemer da jeg skulle gjøre om de høstede ressursene til DC formatet, tabellen som ressursene skulle inn i manglet flere av feltene, og de la jeg til manuelt.

Til sammen tok innstalleringen nesten fem uker, dette kunne ha vært kortet ned dersom jeg hadde hatt mer erfaring på forhånd, installasjons - guiden til Arc hadde tatt hensyn til uerfarne brukere, eller henvist til eksterne ressurser for TomCat og MySql. I tillegg til at det ikke burde vært feil og mangler i SQL scriptet som opprettet databasen. Dette førte til at implementeringen og utarbeidelse av prototypen har blitt noe enklere enn planlagt. Det er valgt å bare lage et system som tester metoden og skriver resultatene til fil, istedenfor å bruke Java Server Pages for å lage webgrensesnitt der man for eksempel kan velge hvor man høster fra og hvordan resultatene skal vises.

Når Arc var på plass og ressurser var blitt høstet var det tid for å implementere metoden som finner automatisk tema, i neste underkapittel blir de mer teoretiske delene av hvordan metoden fungerer beskrevet, for å gi leseren en forståelse av

4.2 Representasjon av dokumenter

Å vise et dokument slik at man kan sammenligne det med andre dokumenter, blir ofte gjort ved å representere dokumentet som en vektor. Da kan man enkelt sammenligne to dokumenter ved å regne ut størrelsen vinkelen mellom dem. Dette underkapittelet handler om hvordan få representert tekst som vektorer, og hvordan man sammenligner to vektorer.

4.2.1 Et dokument en vektor

I faget ”Gjennfinningssystemer og verktøy2” ved Høgskolen i Oslo er det laget en enkel presentasjon til en forelesning som viser den enkleste måten å representere en tekst på en måte som gjør at man kan sammenligne to tekster[16].

Et dokument lar jeg bli representert som *title*- og *description* - feltet i Dublin Core. For å lage en vektor blir først alle radene hentet inn fra DC-tabellen i Arc databasen. *Title* og *description* blir hentet ut for hver rad og lagt til i en liste over alle ordene som samlingen

inneholder, en vocabulary, se Ordliste, det blir sjekket om ordene som blir lagt til allerede finnes i ordlisten, slik at ordlisten ikke blir for stor.

Vektoren for dokumentet blir laget ved at man tar alle ordene i *title* og *description* og sammenligner med ordlisten. Hvert ord i dokumentet blir en forekomst i vektoren, på den plasseringen ordet forekommer i ordlisten. Med vekting fordi dette gir et bedre bilde av dokumentet enn uten[16].

Eksempel:
Det finnes to dokumenter i samlingen som består av en setning hver:
Dokument 1, "*Alle barna var slemme unntatt Jill hun var snill*"
Dokument 2, "*Alle barna var syke unntatt Karoline hun hadde tatt vaksine"*

Indeks	Ordliste	Dokument 1	Dokument 2
1	alle	1	1
2	barna	1	1
3	var	2	1
4	slemme	1	0
5	unntatt	1	1
6	Jill	1	0
7	hun	1	1
8	snill	1	0
9	syke	0	1
10	Karoline	0	1
11	hadde	0	1
12	tatt	0	1
13	vaksine	0	1

Tabell 4-I Vektoreksempel

Tabellen viser hvordan man kan representere to dokumenter som vektorer med vekting.

I Tabell 4-I vises det hvordan ordlistens plassering av ordene og antall forekomster av et ord i en tekst gir elementene i en vektor. Vektorene for tekstene blir som følger:

Dokument 1 (D_1) = [1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
Dokument 2 (D_2) = [1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1]

For at en vektor skal kunne sammenlignes med andre vektorer må den normaliseres, det vil si at man lar lengden til vektoren bli lik 1. Dette gjøres ved å dividere vektoren med absoluttverdien til vektoren er gitt ved:

Definisjon 1

$$\vec{u}_{normal} = \frac{\vec{u}}{|u|}$$

Figur 4 Normal vektor – Definisjon

Kilde for alle utredninger om vektorer er Edwards & Penney: [17]

Normalvektoren til en vektor er gitt ved å dividere en vektor på absoluttverdien til vektoren der $|u|$ er absolutt verdien til vektor $\vec{u}$ [17].
Absoluttverdien til en vektor er gitt ved neste definisjon:

Definisjon 2

$$|u| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}$$

Figur 5 Absoluttverdi til vektor - Definisjon

Eksempel:

Normal vektoren til dokument D_1 er gitt ved:

$$Antall = 13$$

$$Gjennomsnitt D_1 D_2 = \left[\frac{1+1}{2}, \frac{1+1}{2}, \frac{2+1}{2}, \frac{1+0}{2}, \frac{1+1}{2}, \frac{1+0}{2}, \frac{1+1}{2}, \frac{1+0}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}\right]$$

$$Gjennomsnitt D_1 D_2 = [1, 1, 1.5, 0.5, 1, 0.5, 1, 0.5, 1, 0.5, 0.5, 0.5, 0.5]$$

Figur 6 Normal vektor - Eksempel

4.2.2 Sammenligne to vektorer

Den normaliserte formen til vektoren er viktig når man skal sammenligne to vektorer. Sett at et tema og et dokument er representert ved to vektorer. For å sammenligne de to vektorene kan man regne ut cosinus til vinkelen mellom dem. Dette gjøres ved å regne ut skalarproduktet til vektorene. Da får man et tall mellom 0 og 1. Dersom vektorene er helt like vil tallet bli 1 og den inverse til cosinus gir da at vinkelen mellom vektorene er 0. Definisjon 3 viser hvordan man regner ut skalarproduktet til to vektorer[4].

Definisjon 3

$$\vec{u} \cdot \vec{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n$$

Figur 7 Skalarprodukt – Definisjon

Skalarproduktet mellom D_1 og D_2 er gitt ved:

$$\overrightarrow{D_{1normal}} \cdot \overrightarrow{D_{2normal}} = \frac{1}{\sqrt{11}} \cdot \frac{1}{\sqrt{10}} + \frac{1}{\sqrt{11}} \cdot \frac{1}{\sqrt{10}} + \frac{2}{\sqrt{11}} \cdot \frac{1}{\sqrt{10}} + 0 + \frac{1}{\sqrt{11}} \cdot \frac{1}{\sqrt{10}} + 0 \frac{1}{\sqrt{11}} \cdot \frac{1}{\sqrt{10}} + 0 + 0 + 0 + 0 + 0$$

$$\overrightarrow{D_{1normal}} \cdot \overrightarrow{D_{2normal}} = 0,57207$$

$$\overrightarrow{D_{1normal}} \cdot \overrightarrow{D_{2normal}} \approx 0,57$$

$$\theta = \cos^{-1}\theta = \cos^{-1} 0,57 = 55,1^{\circ}$$

Figur 8 Skalarprodukt – Eksempel

Dette viser at det er 55,1 grader mellom dokument 1 og 2 , dette kan stemme bra i og med at det er i underkant av halvparten av ordene i hvert dokument som er like.

4.2.3 Vektor for Tema

Et tema blir representert av en vektor ved å ta gjennomsnittsvektoren for de dokumentene som et tema representerer se Figur 10 i kapittel 4.4, for hvordan man deler opp samlingen i treningssett og finner gjennomsnittsvektor for hvert enkelt tema. Jeg vil nå vise et eksempel på hvordan gjennomsnittsvektoren for dokument 1 og 2 kan regnes ut. Man tar antall vektorer og deler summen av elementene i vektorene. Dette tar utgangspunkt i algoritmen til MacQueen (1967), som er kalt K-means. Dette er en algoritme som sorterer objekter (tekster) ved å finne gjennomsnittet for hver gruppering man vil finne. I min metode er det bare tatt utgangspunkt i denne algoritmen og den er ikke implementert på den måten som er beskrevet i litteraturen, den skiller seg fra litteraturen ved at man finner gjennomsnittsvektoren for en gruppe dokumenter som ligger under et tema som er fastsatt på forhånd. I litteraturen brukes algoritmen til å sortere dokumenter etter hvor nært de ligger hverandre uten at dette er presisert mer [18],[19].

Eksempel:

$$Antall = 13$$

$$GjennomsnittD_1D_2 = \left[\frac{1+1}{2}, \frac{1+1}{2}, \frac{2+1}{2}, \frac{1+0}{2}, \frac{1+1}{2}, \frac{1+0}{2}, \frac{1+1}{2}, \frac{1+0}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}, \frac{0+1}{2}\right]$$

$$GjennomsnittD_1D_2 = [1,1,1.5,0.5,1,0.5,1,0.5,1,0.5,0.5,0.5,0.5]$$

Figur 9 Gjennomsnittsvektor – Eksempel

Gjennomsnittsvektor for tema blir brukt til å sammenligne opp mot vektorene for dokumenter som ligger under dette tema i relevant treningssett. Ved å ta normalisert vektor for tema og dokument finner man vinkelen mellom vektorene ved å regne ut skalarproduktet til vektorene som forklart i Sammenligne to vektorer.

Uten en ordliste som definerer hvordan vektorene skal lages kommer man ikke langt i neste underkapittel blir ordlisten og hvordan denne er laget presentert.

4.2.4 Ordlisten

Ordlisten som vektorene blir laget ut i fra, er som kjent laget av alle ordene i samlingen med dokumenter. Dette er den enkleste måten å lage en ordliste på, ved å ta alle de ulike ordene i samlingen og legge dem inn i en ordliste uten noen form for korreksjon på ordene.

Baeza-Yates et. al nevner flere metoder for å representere tekst, blant annet ved å lage en ordliste som består av termer som kan knyttes direkte til konsepter i dokumentsamlingen. Dette gjøres ved å eliminere stoppord[4] og la ordene bli "bøyd" til grunnformen av ordet. For eksempel kan alt som kan relateres til et konsept bli bøyd til grunnformen av ordet som indikerer konseptet såkalt Stemming. Et eksempel: Konseptet er katt, da blir alle ord som kan relateres til dette konseptet definert som katt. *Katten, pus, kattungen, pusekatten, pusen,* og *kattene* blir alle registrert som katt i ordlisten. For å gjøre dette mulig må man ha en gjenkjennelses mekanisme som lager vektorene slik at står det *pus* i teksten blir det forstått som katt og indeksen til katt øker med en når man skal lage en vektor [4]. Baeza-Yates et. al diskuterer også nødvendigheten og riktigheten av å bruke teknikken med stoppord og stemming, i mange sammenhenger blir resultatet misvisende når man bruker disse to teknikkene, fordi man ikke får med hele bildet av teksten. For eksempel kan det være et viktig poeng at en tekst er skrevet for barn og da er ordet *pusen* et begrep som sier noe om barns forhold til dyr, dersom *pusen* blir satt i kontekstsammenheng med katt kan man tro at teksten er en tekst om katt og ikke et eventyr for barn [4].

Ordlisten jeg har laget har ikke brukt stoppord og stemming på grunn av svakhetene nevnt i forrige avsnitt og på grunn av tiden jeg hadde til rådighet for implementeringen. En interessant utvidelse av systemet ville være å ta i bruk teknikkene helt eller delvis for å se om det hadde noen effekt på om man ville treffe flere temaer med denne metoden. Antagelig ville treffprosenten økt noe om man fjernet stoppord fra ordlisten fordi slik ordlisten er i dag er den på over 15 000 ord og da sier det seg selv at vektorene inneholder mange tomme elementer og sammenligningene blir nokså tilfeldige. Bruk av stemming er en mer komplisert teknikk og siden denne teknikken er omdiskutert vil det i første omgang være nok å bare ta vekk stoppord.

Ordlisten er som sagt meget stor, og hovedgrunnen til dette er at en del av dokumentene i samlingen ikke er på samme språk som resten av dokumentene, da blir det lagt til mange ord som svært få dokumenter inneholder og dette kan føre til at metoden ikke vil finne automatisk tema på så mange dokumenter som det kunne ha vært dersom alle dokumenter var på samme språk. Neste underkapittel tar for seg dette og alle andre sider ved valgte samling for å teste metoden.

[4] Stoppord: *Og, eller, men, for, dersom, ville, hadde* og lignende ord, ord uten semantisk betydning i seg selv blir ofte kalt stoppord og blir ofte eliminert for at sammenligningen av vektorer skal bli bedre og gi et mer sannferdig bilde 4. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. 1 ed. 1999, New York: ACM Press10:0-201-39829-X .

4.3 Samling

Jeg vil nå gi en presentasjon av valgte samling og hvilke krav jeg satte for å kunne velge denne samlingen. For å teste metoden krevdes en samling som var konsis og inneholdt metadata på en slik form at det var mulig å implementere en prototyp som kunne ta i bruk metadataene. Jeg satte derfor opp disse kravene på forhånd:

- *Dataene som skulle høstes og brukes for testing av metoden, måtte ha innhold på disse feltene:*
    - *"Title": Tittelen var viktig for denne var med på å representere et dokument sammen med*
    - *"Description": som jeg i utgangspunktet krever at skal være abstract eller sammendrag for et dokument.*
    - *"Subject": Måtte være med for å kunne sette tema for tekster automatisk.*
    - *"Language": Måtte ha samme innhold for å sikre at dokumentene som var med hadde samme språk.*
- *Dataene i feltene måtte gi inntrykk av å være godt bearbeidet og gi et visst inntrykk av standardisering for hele samlingen.*

Å finne samlinger som dekket alle disse kriteriene var ikke enkelt, men Open Archieves Initiative's nettside hadde en oversikt over alle datatilbydere som hadde registrert seg hos Open Archieves initiative. Denne siden var til litt hjelp for å finne ut om samlingene tilfredstilte kravene mine, men for å se hvordan metadataene faktisk var, ble jeg nødt til å høste samlinger å studere dem i MySQL browseren selv. Dette var en noe tidkrevende prosess og valgte samling er ikke optimal på noen måte, men den tilfredsstiller til en viss grad de kravene jeg har satt [20].

Valget falt på slutt til samlingen, E-prints in Library and Information Science (E-lis). Samlingen oppstod i 2003, og har som hovedmålsetning å være en gratis tjeneste for utveksling av dokumenter innenfor bibliotek og informasjonsvitenskap. Samlingen eller arkivet som de kaller seg selv, er drevet av The open archive for Library and Information Science [21].

Inneholder metadata i ukvalifisert Dublin Core, inneholder mer enn 3500 dokumenter, hvorav 1233 er angitt i language feltet til å være engelsk. Innholdet til samlingen er artikler, presentasjoner, upublisert og publisert. Sidne jeg bare bruker 1233 av dokumentene i samlingen er det en liten samling. Og håpet er at den skal være god nok til å kunne påvise at det er mulig å finne automatisk tema.
For å presentere samlingen mer detaljert viser Tabell 4-II en "record" eller linje i databasen etter at samlingen er høstet.

Title	Scientific and General Subject Classifications in the Digital World
Author or Creator	De Robbio, Antonella
Author or Creator	Maguolo, Dario
Author or Creator	Marini, Alberto
Subject and Keywords	IB. Content analysis (A and I, class.)
Subject and Keywords	I. Information treatment for information services
Subject and Keywords	DI. Science libraries.
Description	In the present work we discuss opportunities, problems, tools and techniques encountered when interconnecting discipline-specific subject classifications, primarily organized as search devices in bibliographic databases, with general classifications originally devised for book shelving in public libraries. We first state the fundamental distinction between topical (or subject) classifications and object classifications. Then we trace the structural limitations that have constrained subject classifications since their library origins, and the devices that were used to overcome the gap with genuine knowledge representation. After recalling some general notions on structure, dynamics and interferences of subject classifications and of the objects they refer to, we sketch a synthetic overview on discipline-specific classifications in Mathematics, Computing and Physics, on one hand, and on general classifications on the other. In this setting we present The Scientific Classifications Page, which collects groups of Web pages produced by a pool of software tools for developing hypertextual presentations of single or paired subject classifications from sequential source files, as well as facilities for gathering information from KWIC lists of classification descriptions. Further we propose a concept-oriented methodology for interconnecting subject classifications, with the concrete support of a relational analysis of the whole Mathematics Subject Classification through its evolution since 1959. Finally, we recall a very basic method for interconnection provided by coreference in bibliographic records among index elements from different systems, and point out the advantages of establishing the conditions of a more widespread application of such a method.
Date	2001-11-01
Resource Type	Journal Article (On-line/Unpaginated)
Resource Identifier	http://eprints.rclis.org/archive/00000026/
Format	html http://eprints.rclis.org/archive/00000026/01/HEPderobbio.htm
Language	en

Tabell 4-II E-LIS post

Tabellen er hentet fra http://eprints.rclis.org/perl/oai2?verb=ListRecords&metadataPrefix=oai_dc den 04.07.2006 og viser innholdet til et dokument i samlingen. Dette er et felt der title og description er på engelsk. Legg merke til at det finnes tre subject til dette dokumentet, i databasen er de avdelt med tegnet dette tegnet: |.

Noen fakta om samlingen hentet fra nettstedet til E-LIS [21]:

- Det er brukt ukvalifisert Dublin Core i samlingen.

- Subject feltene gir tema for dokumentet og alle tema er laget ut i fra et klassifiseringsskjema som sier noe om tema til dokumentet og også hvilken fysisk form det finnes på, som for eksempel harddisk cd-rom og lignende.
- Alle subject-feltene skal være på engelsk, uavhengig av språket brukt i dokumentene.

En viktige egenskaper jeg har registrert hos samlingen er at ”language” feltet i noen tilfeller angir feil språk, for eksempel at tittelen er på engelsk mens ”description” er på et annet språk. Dette kan skape problemer for hvor godt man treffer med tema, når metoden for å finne automatisk tema skal implementeres. I Tabell 4-III ser man et eksempel på et ”feilmerket” dokument.

Title	Interconnessioni tra classificazioni scientifiche e classificazioni generali nel mondo digitale
Author or Creator	De Robbio, Antonella
Author or Creator	Maguolo, Dario
Author or Creator	Marini, Alberto
Subject and Keywords	I. Information treatment for information services
Description	[Italian abstract] Il presente lavoro si riferisce ai problemi di comunicazione fra i mondi specialistici della scienza e della tecnologia e il mondo delle biblioteche, in relazione alle classificazioni disciplinari specifiche da una parte e le classificazioni generali utilizzate nelle biblioteche, in particolare la classificazione decimale Dewey (CDD) dall'altra. Dopo aver richiamato alcune nozioni generali sulla struttura, la dinamica e le mutue interferenze delle classificazioni semantiche, si espongono le linee essenziali di una sinossi diacronica delle classificazioni specifiche per le aree della matematica, dell'informatica, dell'ingegneria, della fisica e dell'astronomia. Di seguito si passa in rassegna lo sviluppo della CDD in queste aree, fino alla revisione recentemente proposta della sezione 510 della CDD (matematica, con l'esclusione dell'informatica teorica e della fisica matematica) nella prospettiva della prossima edizione (22.) della CDD. Questo articolo è, in parte, una rielaborazione in lingua italiana del lavoro dal titolo Mathematics Subject Classification and related classifications in the digital world, che appare nei Proceedings della Conferenza internazionale "Crimea2001" tenutasi a Sudak, in Ukraina, dal 9 al 17 giugno 2001 [1]. In quella occasione è stato presentato il nuovo sito La pagina delle classificazioni scientifiche [2], che raccoglie i prodotti del lavoro finora svolto.
Date	2001-07-01
Resource Type	Journal Article (On-line/Unpaginated)
Resource Identifier	http://eprints.rclis.org/archive/00000024/
Format	pdf http://eprints.rclis.org/archive/00000024/01/bibliotimecrimea1.pdf
Format	other http://eprints.rclis.org/archive/00000024/02/bibliotimecrimea1.rtf
Language	en

Tabell 4-III E-LIS Dokument i samling med feilmerket språk.

Tabellen viser et eksempel på et dokument som er merket som engelsk, men som likevel er på italiensk.Kilde: http://eprints.rclis.org/perl/oai2?verb=ListRecords&metadataPrefix=oai_dc den, 04.07.2006.

Neste underkapittel tar for seg selve implementeringen av prototypen med hvilke systemkrav og forutsetninger som ble satt før implementasjonen startet.

4.4 Implementering av prototyp

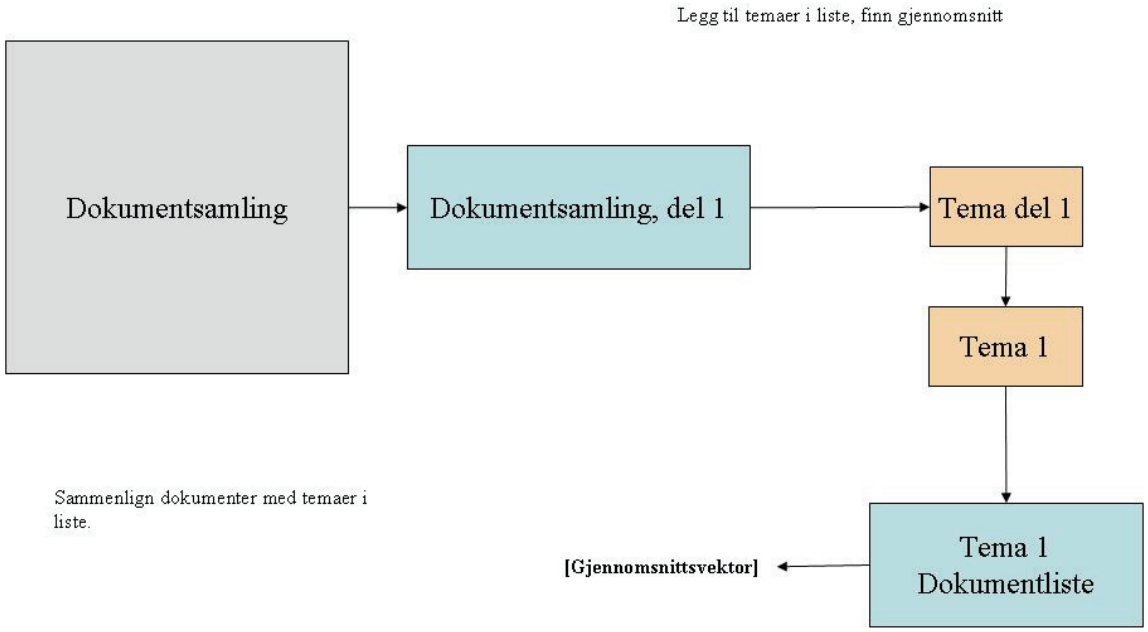
Jeg skal nå gi en beskrive hvordan jeg implementerte metoden finne automatisk tema på tekster. Prototypen er implementert i henhold til beskrevet teori og metoder i foregående

underkapittel. Dette kapittelet skal gi en innsikt i arkitektur og hvordan selve implementeringen har foregått.

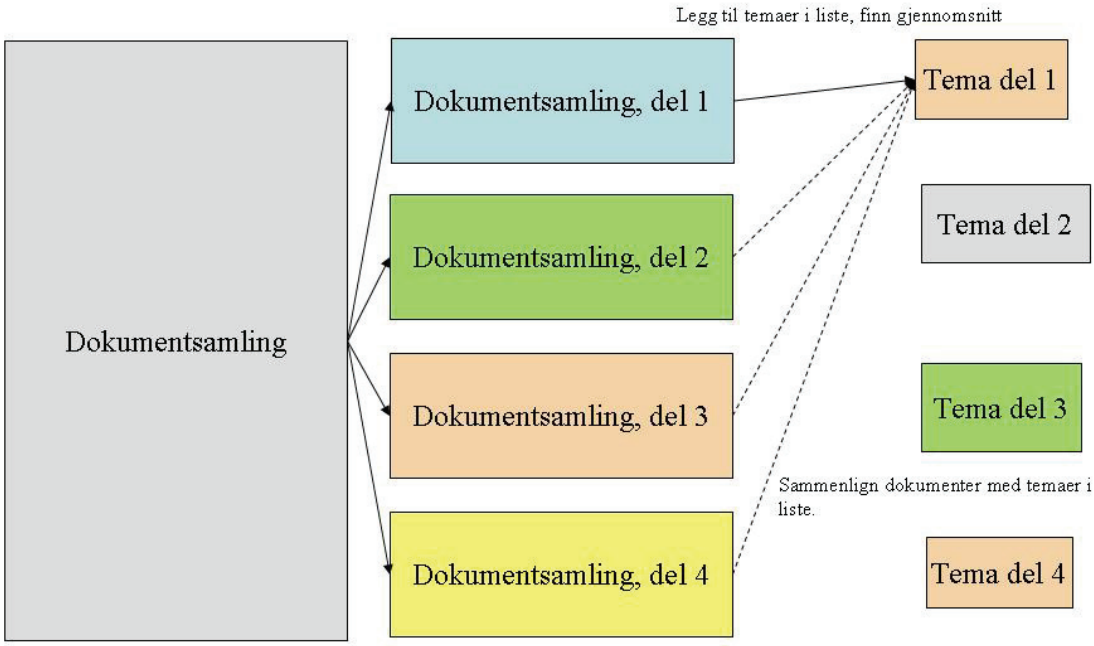
4.4.1 Overordnet arkitektur

Før jeg redegjør i detalj hva prototypen skal gjøre, blir den overordnete arkitekturen for systemet presentert. Prototypen skal finne tema ved å sammenligne dokumenter med en gruppe temaer. For å gjøre dette blir dokumentene og temaene representert som vektorer som beskrevet i kapittel 4.2. For å kunne lage vektorer er man avhengig av å ha en ordliste. Første del av systemet tar seg av denne jobben. Ved at alle dokumentene blir hentet i fra databasen, og ”title” og ”description” for hvert dokument blir hentet ut og ordene lagt i en liste som blir sendt til en fil. Se Figur 12.

Selve kjøringen



Figur 10 Oversikt over kjøring av systemet
En dokumentsamling blir delt opp i flere deler, første del som vist på figur blir gått gjennom og for hvert dokument blir tilhørende tema/subject lagt i en liste over tema for første del av samlingen. For hvert tema blir dokumentene som har dette temaet lagt i en liste. Ved å finne gjennomsnittet for alle dokumentene ved å bruke en clustering metode kalt k-means, finner man en gjennomsnittsrepresentasjon for temaet som man kan sammenligne med dokumentene i resten av samlingen.



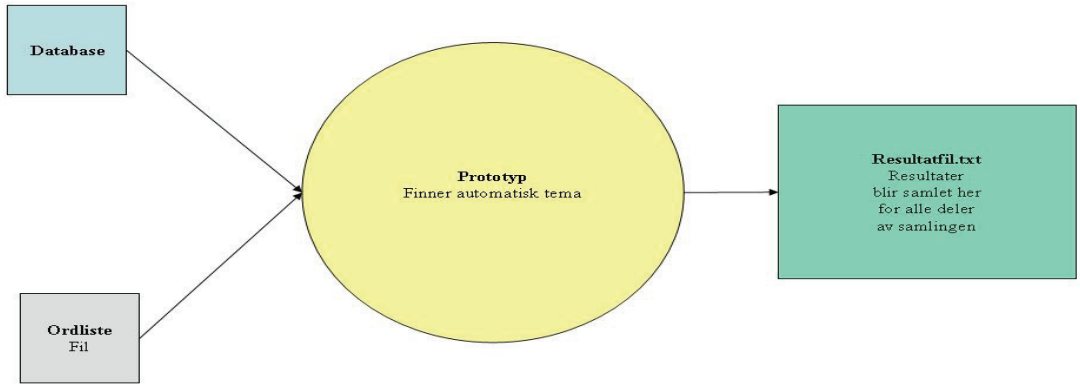
Figur 11 Oppdeling av samling

Figuren viser hvordan dokumentsamlingen blir delt opp i flere deler og at hver del lager en liste over tema som hører til denne delen og gjennomsnittet for hvert tema blir representert på en slik måte at det er mulig å sammenligne dokumenter med tema. På figuren viser stiplede linjer sammenligning mellom deler utenfor testsettet i del 1.

Figur 12 Opprette Ordliste

Viser hvordan første del av systemet henter ut alle ordene som finnes i "title" og "description" feltet i databasen og lager en liste som blir sendt til en fil.

Denne første delen blir bare gjort en gang, og har kun til hensikt å opprette ordlisten. Neste operasjon til prototypen blir å lage treningssett for temaene og teste dette mot resten av samlingen. Dette er en ganske komplisert operasjon og Figur 13 viser helt overordnet hvilke data som går inn til systemet og hvilke som går ut av systemet og skrives til fil. Figur 10 og Figur 11 viser hvordan det er tenkt at prototypen skal kjøre gjennom samlingen ved å la enkelte deler være treningsdeler mens de andre delene er testdeler.



Figur 13 Overordnet arkitetkur av Prototypen

Figuren viser hva som går inn og ut til systemet, prototypen henter inn ordlisten fra fil og dokumentene og temaene fra databasen, og skriver resultatet fra kjøring av prototypen til en tekstfil, som kan bearbeides og lage fine grafer i Microsoft Excel.

For å klarlegge hva selve prototypen skal gjøre har jeg satt opp en liste med systemkrav som sier noe om hva som skal implementeres.

4.5 Systemkrav og begrensninger

Systemkravene beskriver oppgaver og valgte løsninger for prototypen.

Først litt tekniske data om prototypen. Den er implementert ved å bruke Java versjon 1.4, grunnen til at jeg ikke bruker siste versjon av Java er fordi Arc høsteren kun kan kjøres på denne versjonen, og det viste seg i utviklingsarbeidet at jeg hadde behov for å teste ut flere samlinger underveis og da var det lite hensiktsmessig å risikere at høsteren ikke fungerte på grunn av feil versjon av Java. Ellers bruker prototypen samme database som høsteren Arc, og skriver og leser i fra fil for forskjellige operasjoner som er beskrevet under.

Prototypen skal lage normaliserte vektorer av tittel og description feltet i DC-tabellen i MySQL-database, for hvert dokument (er representert som en rad i tabellen). Dette skjer ved at tittel og description blir slått sammen til en tekststreng uten vekting. Slik at både tittel og description vektes likt når vektorene skal lages. Det kan diskuteres om dette er en bra måte å gjøre det på, kanskje man burde funnet en metode slik at tittelen fikk mer vekting enn description eller omvendt, jeg har ikke funnet noen bedre måte å gjøre dette på, så har derfor valgt å la det være slik.

Ordlisten skal lages på grunnlag av alle ordene i fra ”title” og ”descripton” -feltene i databasen, der ”language” er satt til å være engelsk. Stoppord og Stemming teknikker vil ikke bli tatt i bruk i implementering av prototypen, se kapittel Ordlisten.

Prototypen skal ikke ha noe brukergrensesnitt og resultater fra kjøring blir skrevet til en tekstfilfil (se vedlegg B) for videre behandling i Microsoft Excel.

Siden det er en prototyp som skal teste en metode og det ikke er et utviklingsprosjekt, blir det ikke laget så omfattende dokumentasjon som for tradisjonelle utviklingsprosjekt. Dokumentasjonen vil derfor bestå av

- Klassediagram, et detaljert i vedlegg X og et overordnet som viser pekere mellom klasser i neste underkapittel.
- En detaljert beskrivelse i neste underkapittel som viser hva hver klasse gjør.
- Kommentering av kode, koden blir kommentert etter beste evne, slik at andre skal kunne skjønne hva som skjer i de enkelte klasser. Dette blir gjort ved å generere Javadoc av kommentarene.

Dette er de viktigste kravene og begrensningene for prototypen, neste kapittel tar for seg hvordan selve implementering er gjort med dokumentasjon i form av klassediagram.

4.6 Implementering av prototyp som tester metoden

Jeg vil nå ta for meg alle klassene i systemet og beskrive hva de gjør og hvordan de kjører. Dette for å gi et innblikk i hvordan prototypen er implementert i forhold til beskrevet teori. Henviser til Javas API for beskrivelse av ArrayLister, HashSet og andre spesifikke java syntakser [22].

4.6.1 Vektormodell

Dette er den viktigste klassen i prototypen, her blir vektorer laget og gjennomsnittsvektorer for ”Subject” (tema) blir funnet.

Hovedoppgaven til vektormodell er å ta inn dokumenter og lage dem om til vektorer. Dette gjøres ved å sende inn et objekt av klassen Document og et objekt av klassen Samling til konstruktøren. De viktigste oppgavene til klassen er:

- Opprette en HashMap for hvert dokument, som holder orden på alle ordene i dokumentet og antallet av hvert ord.
- HashMapen for hvert dokument blir brukt når man skal lage vektor for dokumentet, da blir ordene(nøklene) i HashMapen sammenlignet med ordlisten og forekomsten til

ordene i HashMapen blir lagt til i en double[] tabell, der indeksen fra ordlisten bestemmer plassering og verdien av ordet(nøkkel) i HashMapen bestemmer størrelse på element i vektor.
- Lage normaliserte vektorer for dokumentene i form av double[] tabeller.
- Finne skalarproduktet (se kapittel 4.2.2) til to vektorer.
- Finne gjennomsnittsvektor for et Subject objekt som inneholder en liste med dokumenter.
- Resten av metodene i denne klassen er hjelpemetoder som brukes i disse oppgavene og jeg henviser leseren til kodekommentarene for mer informasjon om denne klassen.

4.6.2 Document

Denne klassen representerer dokumentene i samlingen. Her blir ”title” og ”description” -feltet i fra databasen hentet inn som strenger i konstruktøren og gjort om til ”vaskede” ord. ”Vaskede” ord vil si at tegn som komma, punktum og ropetegn er fjernet i tilknytning til ordene. Dette er gjort fordi det ble oppdaget tidlig at et ord med komma etter seg ble regnet som nytt ord, selv om det hadde mange forekomster som var helt like. De viktigste oppgavene til klassen er:

- Vaske tekst som kommer i fra databasen slik at det representerer dokumentet entydig, uten forstyrrelser i form av at ord blir representert som ulike på grunn av tegn på slutten av ordet.
- Sette en normalisert vektor for dokumentet som blir brukt ved sammenligning mot en gjennomsnittsvektor.
- Identifisere hvilke Subject (tema) som faktisk hører til dette dokumentet ut i fra Subject feltet i databasen.

4.6.3 Subject

Klassen representerer et tema, temaene er funnet ved å gå gjennom ”subject” feltet i DC-tabellen i databasen. Konstruktøren tar inn en tekststreng som er navnet på Subject. Viktigste oppgavene til Subject er:

- Holde orden på dokumentene som har dette Subject i den delen av samlingen som er definert som treningssett.
- Lage en normalisert gjennomsnittsvektor for alle dokumentene som hører til dette Subject.

4.6.4 Samling

Samling er en klasse som holder orden på listen over alle dokumenter i samlingen. Den tar ikke i mot noe i konstruktøren og inneholder en ArrayList med alle dokument objektene. De viktigste oppgavene til denne klassen er:

- Hente ordlisten og ha den tilgjengelig for de andre klassene. Etter første gjennomkjøring er ordlisten tilgjengelig fra fil så denne operasjonen går mye raskere når ordlisten først er opprettet.
- Oppretter dokumenter ved å hente tekststrenger for ”title” og ”description” i databasen.

- Legger til dokumenter i ArrayListen og dersom første gang kjører legger den til ordene i fra dokumentene i ordlisten Vocabulary.
- Metoder som gir andre klasser tilgang på ordliste og dokumenter.
- Legge dokumenter i fra deler av samlingen inn i listen til RelevantSubject slik at et treningssett blir laget.

4.6.5 Vocabulary

Vocabulary-klassen inneholder bare ordlisten for alle dokumentene i samlingen. Ordlisten er dobbelt lagret i en ArrayListe som holder orden på rekkefølgen til ordene, slik at alle vektorer har samme dimensjoner og elementene i vektorene kommer på riktig plass. I tillegg har Vocabulary et HashSet for hurtig oppslag i ordlisten. Ordlisten hentes fra fil som ArrayList og HashSet unntatt når den blir opprettet første gang, da er det dokumentenes tittel og description som oppretter ordlisten ut fra kall til database. Når ordlisten er komplett skrives den til fil for gjenbruk ved senere kjøring av metoden. Dette gjør at tiden det tar å kjøre prototypen blir kortet inn, uten at det er målt hvor mye tid man sparer.

4.6.6 RelevantSubjects

Denne klassen er i praksis treningssettet for prototypen. Den tar seg av de dokumenter som er innenfor de delene av samlingen som er definert til å være treningssett for samlingen. Klassen inneholder i likhet med Vocabulary en ArrayList og et Hashset med de relevante Subjectene fra aktuell del av samling. I tillegg inneholder klassen en ArrayList med gjennomsnittsvektorer for identifiserte Subject for denne delen av samlingen.
De viktigste oppgavene til klassen er å lage gjennomsnittsvektor for de Subjectene som er identifisert i dokumentene og sammenligne dokumenter med gjennomsnittsvektorer og returnere de fem likeste subjectene.

4.6.7 DbKobling

Klasse som oppretter forbindelse til database, brukes av DbId og SubjectList for å hente ut data av databasen og sender videre til de andre klassene.

4.6.8 DbId

Dette er en klasse som inneholder en ArrayList over alle ”id” feltene i databasen. Denne ArrayListen brukes for å gjøre databasekall av Samling og SubjectList. Ved å gå gjennom listen finner man indeksen til alle ”id”-ene i databasen og man får nøyaktig den linjen i databasen man er ute etter. Dette er gjort for å spare en del databasekall.

4.6.9 SubjectList

Inneholder en liste med alle Subjectene som er funnet i databasen. Viktigste oppgave til denne klassen er å identifisere alle ulike Subject i ”Subject”-feltet i databasen og opprette Subject-objekter for de identifiserte Subjectene. Listen brukes til å sammenligne med Subjectene som Samling-klassen identifiserer når dokumentene får lagt til de reelle subjectene.

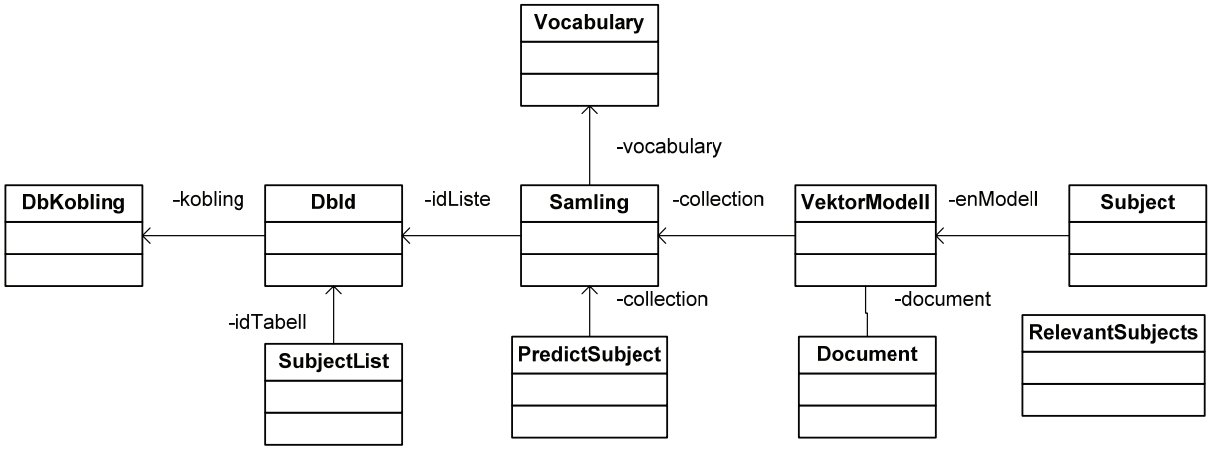
4.6.10 PredictSubjects (main)

I denne klassen blir tema for hvert dokument i samlingen ”gjettet” og sammenlignet med reelle tema for dokumentene. Dette gjøres ved at metoden i RelevantSubject som angir de fem mest relevante Subject for hvert dokument blir sammenlignet med listen i hvert dokument med de reelle Subject. En metode for hvert relevante Subject teller antall treff for dokumentene i en del av samlingene. Det blir telt på antall dokumenter som treffer på det likeste foreslåtte Subject, antall som treffer på en av de to, tre fire eller fire likeste Subjectene. Det blir ikke tatt hensyn til at et dokument gjerne har flere subject, blir det treff på et av subjectene i listen over subject regnes det som treff. Det vil altså si at det er større sannsynlighet for treff for dokumenter med flere subject, enn dokumenter med bare ett subject.

Klassen inneholder metoder for å lese og skrive fra/til fil, og resultatene av sammenligningene blir skrevet til fil.

4.6.11 Klassediagram

I Figur 14 vises et enkelt klassediagram over prototypen. I vedlegg B finnes et mer detaljert klassediagram. I diagrammet under ser man at Vektormodell er den viktigste klassen og ”motoren” i systemet, her blir alle vektorer laget og sammenligninger foregår her. RelevantSubjcets holder orden på tema i treningssett og tester hvor nærme dokumentene ligger temaene. Klasser som holder kontakt med databasen er DbKobling, DbId og SubjectList. Klasser som har med samlingen er Vocabulary, Samling og Document. PredictSubject er main klassen her blir resultatene skrevet til fil. Subject jobber opp mot vektormodell på en lignende måte som Document.



Figur 14 Enkelt klassediagram

Enkelt klassediagram, uten detaljer viser oppbygging av system.

5 Resultater av kjøring

For å teste om riktig tema er blitt funnet, ble prototypen kjørt slik at den først laget et treningssett for første del av samlingen, så ble de andre tre delene av samlingen testet mot treningssettet. Antall treff per del ble registrert ved å sammenligne reelle tema for hvert dokument, med foreslåtte tema. I utgangspunktet var planen å bare teste dokumentene mot det temaet som var likest dokumentene, men siden det var like enkelt å implementere slik at flere temaer kunne være med i et treff ble dette gjort.

Resultatene fra kjøring ble skrevet til en tekstfil, denne filen ble så bearbeidet i et Microsoft Excel regneark og diagrammer som viser treffprosenten for de forskjellige treningssettene ble laget. Det er disse resultatene med tilhørende diagrammer jeg skal presentere i dette kapittelet.

Under testing av prototypen, ble metoden også testet for de hundre første dokumentene i samlingen. Det var ikke meningen at resultatene fra denne testingen skulle være med i de endelige resultatene for kjøring av prototypen, men da resultatene fra det veldig lille utvalget av samlingen avvek mye fra kjøring på hele samlingen ble den veldig lille delen tatt med i dette kapittelet og omtalt i diskusjonen i neste kapittel. Derfor er dette kapittelet delt opp i en del for hele samlingen og en del som omhandler den veldig lille delen av samlingen.

Tabellene som presenteres i dette kapittelet er bygget opp på samme måte ved at antall tema (subject) som blir foreslått vises et mer tema for hver kolonne bortover, øverst til venstre vises hvilken del av samlingen som er treningssett og for hver rad ser man resultatene for hver del av samlingen. Tabellene er delt opp i en for reelle treff, og treff i prosent. For reelle treff vises økningen i antall treff i parentes. For prosentdelen vises også gjennomsnitt for de tre testdelene. Til slutt vises et diagram for gjennomsnittet.

5.1 Hele samlingen

Her blir resultater fra kjøring mot hele databasen med dokumenter presentert. Trenden for hele samlingen er at treffprosenten er generelt lav, selv med fem tema/subject med i treffet blir resultatet dårlig. Det merkverdige er at dette også gjelder når treningssettene blir testet på seg selv.

5.1.1 Treningssett 1

Treningssett 1 har de dårligste resultatene, treffprosenten her er dårlig for alle de tre testede delene og også når treningssettet blir testet mot seg selv.
I Tabell 5-I vises de reelle resultatene fra treningssett 1, legg merke til at del 1 har mange flere treff enn de andre delene. Dette kommer av at del en blir testet mot seg selv som er treningssett.

Treningssett 1	1	2	3	4	5	Totalt
Del 1	48	51 (+2)	54 (+3)	58 (+4)	60 (+2)	308
Del 2	19	33 (+14)	37 (+4)	49 (+12)	54 (+5)	308
Del 3	9	22 (+13)	34 (+12)	43 (+11)	52 (+9)	308
Del 4	11	21 (+10)	35 (+14)	36 (+1)	40 (+4)	308

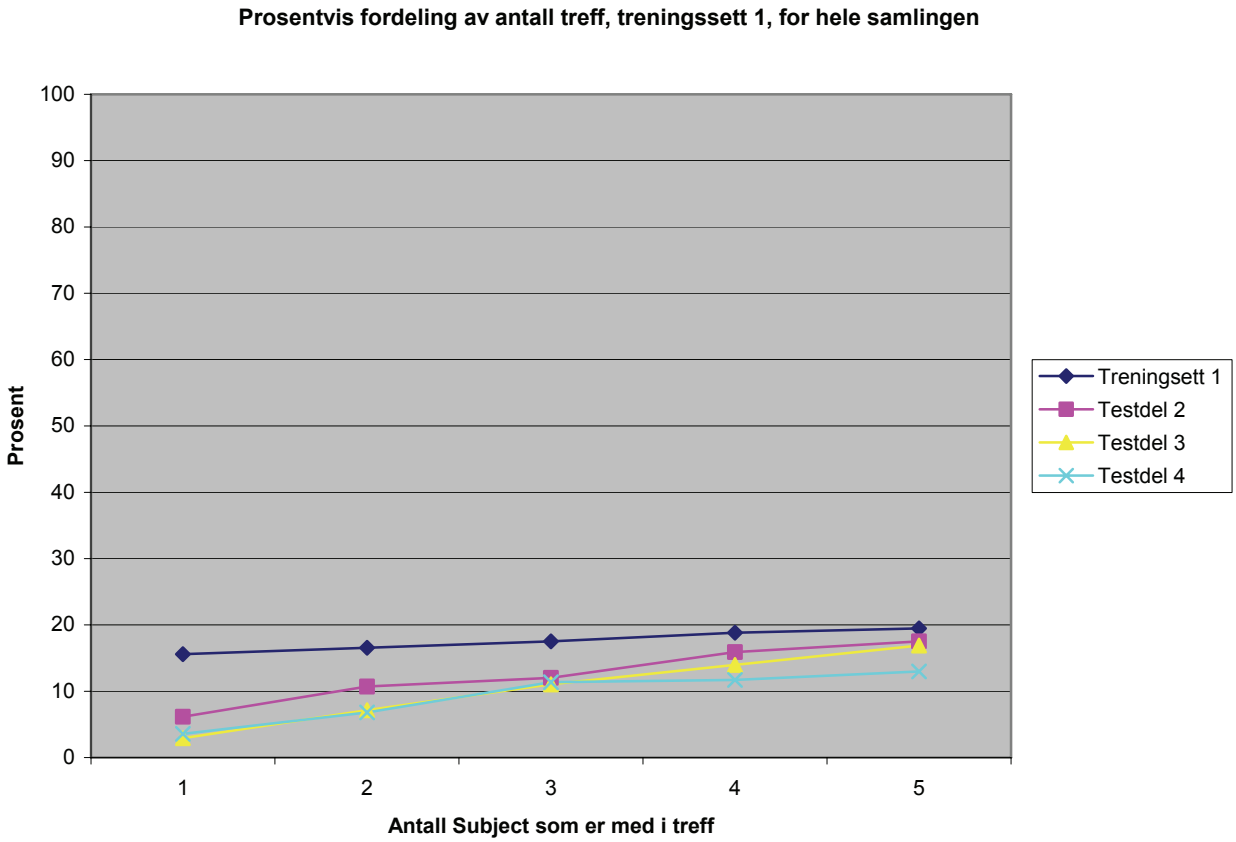
Tabell 5-I Antall reelle treff Treningssett 1

Tabell 5-I viser antall reelle treff for hver del av samlingen, og hvordan dette antallet øker dess flere subject/tema som er med i sammenligningen. Antall dokumenter i hver del er 308, antall dokumenter totalt er 1233. Tall i parentes viser økningen av treff i forhold til økningen av antall subject/tema som er med i treffet. Vi ser her at treningssettet testet mot seg selv øker mindre enn de andre delene, dette kan forklares ved at treningssettet er trenet mot subject/tema og vil derfor få flere treff på det første subject og økningen blir derfor mindre.

Treningssett 1	1	2	3	4	5
Del 1	15,6	16,6	17,5	18,8	19,8
Del 2	5,5	10,4	12,0	15,9	17,5
Del 3	3,6	7,8	11,7	14,9	17,9
Del 4	3,9	6,8	11,4	11,7	13,3
Gjennomsnitt	4,2	8,2	11,5	13,9	15,8

Tabell 5-II Prosentvis treff treningssett 1

Tabell 5-II Viser den prosentvise fordelingen av treff for treningssett 1. Denne tabellen gjenspeiler resultatene fra Tabell 5-I, ved at vi ser at treningssettet øker mindre enn de andre delene av samlingen. Gjennomsnittet er regnet ut for de tre testdelene og vi ser at snittet for testdelene når fem subject er med i treffet blir 15,8 %, dette er svært lavt.



Figur 15 **Diagram, prosentvis fordeling av antall treff, Treningssett 1**

Figur 15 viser et diagram for prosentvis fordeling av antall treff fra testsett 1, del 1er treningssett og regnes ikke som resultater, men er tatt med for å vise forskjellen mellom testdelene og treningssettet. Legg merke til at generelt både for treningsdelen og testdelen er treffprosenten meget lav.

5.1.2 Treningssett 2

Treningssett 2 har bedre resultater enn treningssett 1, testet mot seg selv, men resultatene for testdelene er litt dårligere enn for treningssett 1.

Treningssett 2	1	2	3	4	5	Totalt
Del 1	4	7 (+3)	9 (+2)	9 (0)	12 (+3)	308
Del 2	82	85 (+3)	87 (+2)	89 (+2)	91 (+2)	308
Del 3	6	13 (+7)	14 (+1)	22 (+8)	30 (+8)	308
Del 4	4	12 (+8)	14 (+2)	16 (+2)	17 (+1)	308

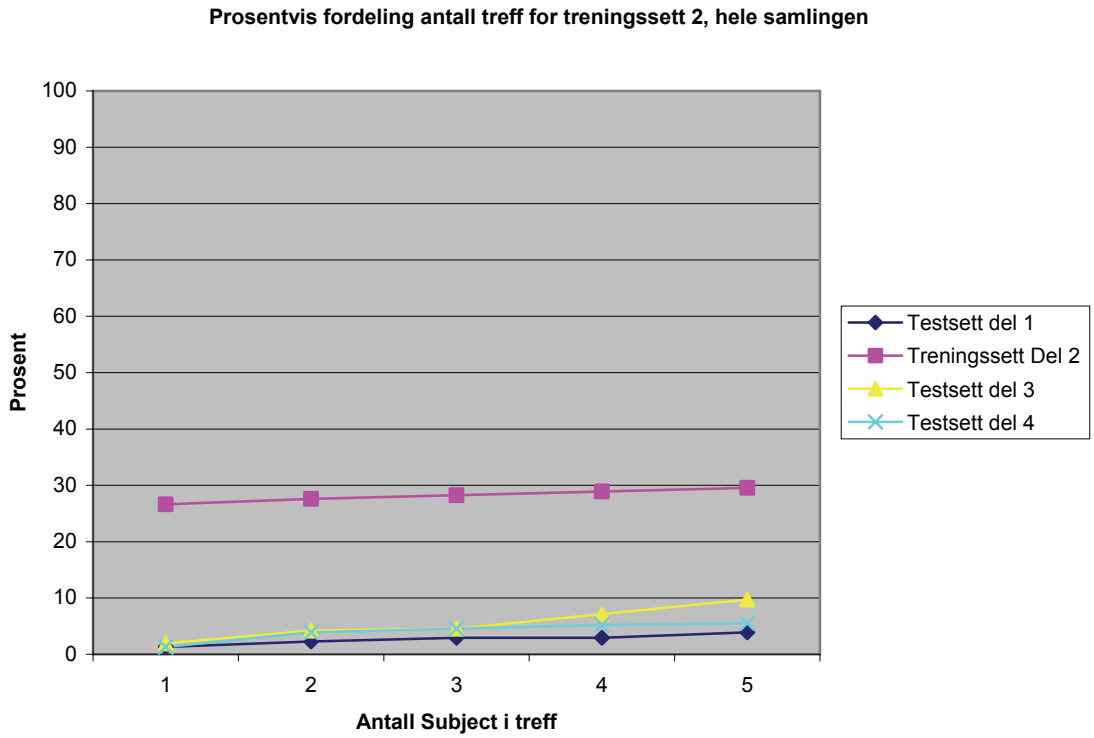
Tabell 5-III Reelle treff Treningssett 2

Tabell 5-III viser reelle tall for treningssett 2. Økningen for testdelene er en del mindre enn for treningssett 1 og det er noen færre treff. Som ved treningssett 1 er også her økningen til treningssettet testet mot seg selv lav. Treffene for testdelene begynner lavt her, men vi ser at etter tre subject øker treffene.

Treningssett 2	1	2	3	4	5
Del 1	1,3	2,3	2,9	2,9	3,9
Del 2	26,6	27,6	28,2	28,9	29,5
Del 3	1,9	4,2	4,5	7,1	9,7
Del 4	1,3	3,9	4,5	5,2	5,5
Gjennomsnitt	1,5	3,5	4,0	5,1	6,4

Tabell 5-IV Prosentvis treff Treningssett 2

Tabell 5-IV viser prosentvis fordeling for treningssett 2. Siden det var en del færre treff for dette treningssettet enn treningssett 1, ser vi at treffprosenten er lav. Gjennomsnittet er regnet ut for de tre testdelene av dette treningssettet. Treningssettet er derfor ikke med i gjennomsnittet. Et gjennomsnitt på 6,4 % når fem subject er med, er enda lavere enn for treningssett 1.



Figur 16 Diagram, prosentvis fordeling av antall treff, Treningssett 2

Figur 16 viser et diagram over prosentvis fordeling av treff for treningssett 2. Verdt å merke seg er at treningssettet ligger mye høyere enn testdelene, og treningssett 2 har litt høyere treffprosent mot seg selv enn treningssett 1. Generelt er resultatene for testdelene veldig dårlige for dette treningssettet, selv med 5 subject med i treff kommer ingen av testdelene seg over 10 prosent.

5.1.3 Treningssett 3

Treningssett 3 har de beste resultatene for hele samlingen. Både treningssettet og testdelene ligger noen få prosent over de andre delene.

Treningssett 3	1	2	3	4	5	Totalt
Del 1	1	4 (+3)	7 (+3)	15(+8)	17 (+2)	308
Del 2	17	26 (+9)	30 (+4)	37 (+7)	42 (+5)	308
Del 3	90	102 (+12)	105 (+3)	109 (+4)	114 (+5)	308
Del 4	11	17 (+6)	20 (+3)	28 (+8)	31 (+3)	308

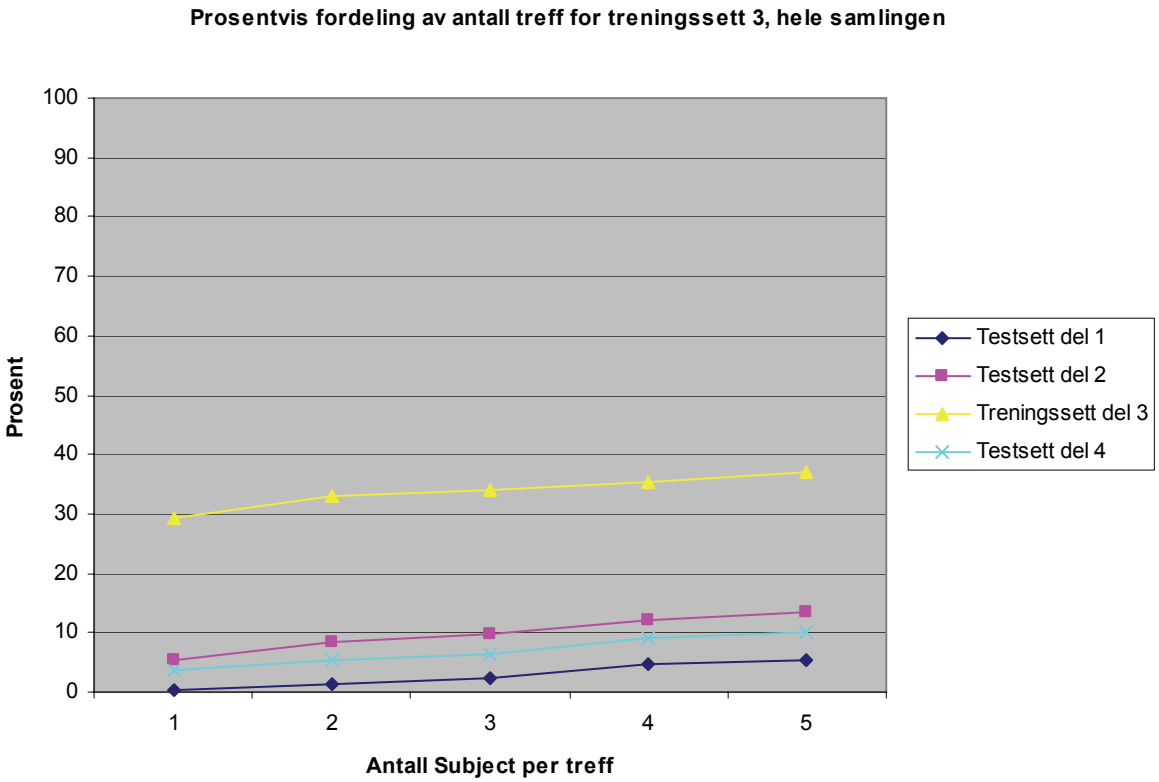
Tabell 5-V Reelle treff treningssett 3

Tabell 5-V viser antall reelle treff i treningssett 3. Her ser vi at treningssettet mot seg selv øker ganske mye fra første subject til andre subject, men økningen avtar og stopper på 114 når fem subject er med i treff. Testdelene har noen mer treff enn i treningssett 2. Og med fem subject i treffet, nærmer den beste testdelen seg 20 prosent.

Treningssett 3	1	2	3	4	5
Del 1	0,3	1,3	2,3	4,9	5,5
Del 2	5,5	8,4	9,7	12,0	13,6
Del 3	29,2	33,1	34,1	35,4	37,0
Del 4	3,6	5,5	6,5	9,1	10,1
Gjennomsnitt	3,1	5,1	6,2	8,7	9,7

Tabell 5-VI Prosentvis treff treningssett 3

Treffprosenten er fortsatt lav og Tabell 5-VI viser treffprosenten for treningssett 3. Legg merke til at gjennomsnittet for testdelene fortsatt er lavt, selv med fem subject i treffet.



Figur 17 **Diagram, prosentvis fordeling av antall treff, treningssett 3**

Figur 17 viser diagram over prosentvis fordeling for treningssett 3. Her ser en at treningssettet mot seg selv er høyere enn i de to foregående treningssett, men fortsatt er testdelene lave. Så langt har testdel 1 scoret dårligst både som testdel og som treningssett.

5.1.4 Treningssett 4

Treningssett 4 gir de nest beste resultatene, og ligger dermed litt under treningssett 3.

Testsett 4	1	2	3	4	5	Totalt
Del 1	2	8 (+6)	14 (+6)	20 (+6)	24 (+4)	308
Del 2	18	30 (+2)	34 (+4)	41 (+5)	46 (+5)	308
Del 3	13	27 (+14)	29 (+2)	37 (+8)	44 (+7)	308
Del 4	83	91 (+8)	95 (+4)	103 (+8)	103 (0)	308

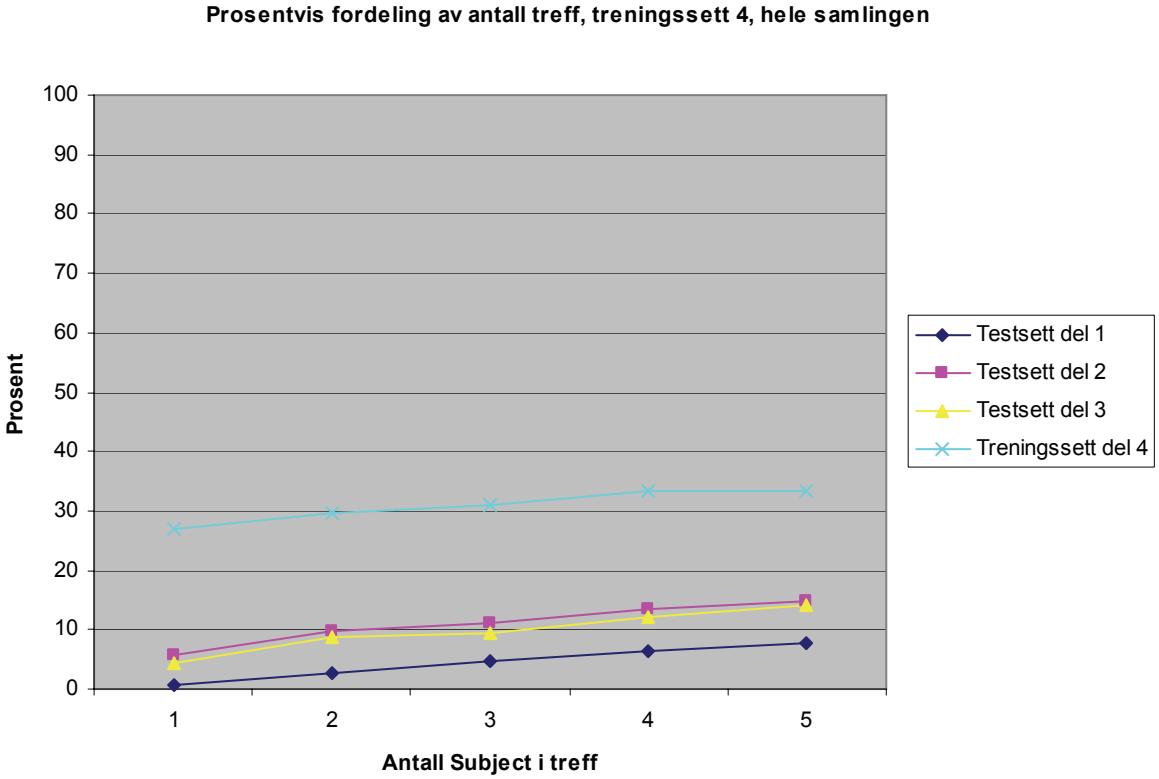
Tabell 5-VII Reelle treff treningssett 4

Tabell 5-VII viser reelle treff for treningssett 4. Dette treningssettet har ikke så mange treff når det er testet mot seg selv som treningssett 3 og testdelene ligner ganske mye på testdelene for treningssett 3. Fortsatt scorer testdel 1 lavest.

Treningssett 4	1	2	3	4	5
Del 1	0,6	2,6	4,5	6,5	7,8
Del 2	5,8	9,7	11,0	13,3	14,9
Del 3	4,2	8,8	9,4	12,0	14,3
Del 4	26,9	29,5	30,8	33,4	33,4
Gjennomsnitt	3,6	7,0	8,3	10,6	12,3

Tabell 5-VIII Prosentvis treff Treningssett 4

Tabell 5-VIII viser prosentvise treff for treningssett 4. Trenden er som nevnt for tabell 4-VII at testdelene ligger ganske jevnt med forrige treningssett. Gjennomsnittet forholdsvis høyt når det er fem subject med i treffet.

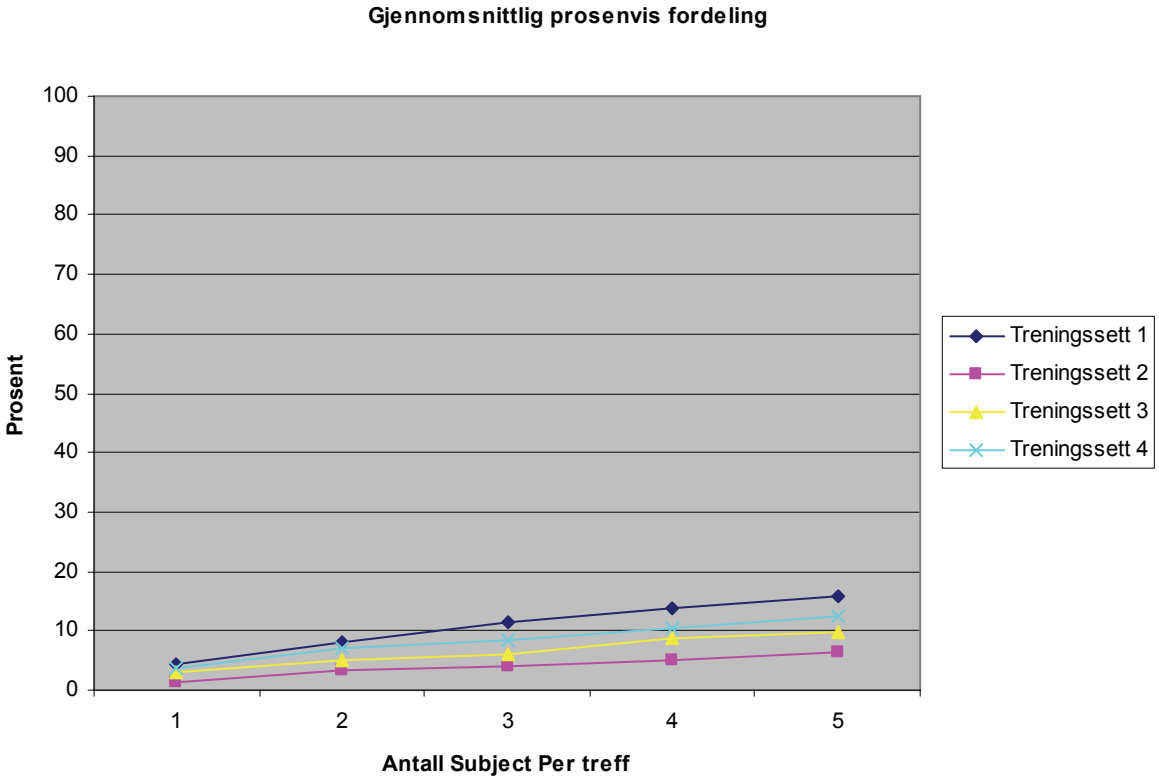


Figur 18 **Diagram, prosentvis fordeling av antall treff, Treningssett 4**

Figur 18 viser diagram over prosentvis fordeling av treff for treningssett 4. Diagrammet visualiserer det vi har lagt merke til hele veien at testdel 1 scorer lavere enn de andre testdelene. Hva dette kan skyldes blir diskutert i neste kapittel.

5.1.5 Gjennomsnittlig fordeling for treningssettene

For å få med et inntrykk av de generelle trendene i testingen av prototypen, har jeg laget et diagram som viser den gjennomsnittlige prosentvise fordelingen av treffene for alle treningssettene. Figur 19 viser at trenden er at ved flere tema/subject som kan være med i et treff blir treffprosenten større, men likevel er treffprosenten lav.



Figur 19 **Diagram, gjennomsnitt prosentvis fordeling hele samling**

5.2 Hundre første dokumenter

Som nevnt innledningsvis i dette kapittelet ble det kjørt tester på prototypen med de hundre første dokumentene i samlingen. Det viste seg at resultatene fra gjennomkjøring med disse dokumentene var mye bedre enn for hele samlingen, og for å kunne diskutere og avdekke grunnen for dette er også disse resultatene tatt med i dette kapittelet. Hele samlingen er en liten samling, vanligvis brukes det mye større samlinger for å teste lignende metoder eller teorier. Siden det som nevnt i kapittel 4.3 om valgte samling, var det vanskelig å finne en god samling som inneholdt alle felter på en konsekvent måte som jeg trengte for å teste min metode. Denne delen med 100 dokumenter kan derfor ikke brukes for å avgjøre om metoden vil virke i seg selv, men den kan avdekke trender og forhåpentlig gi noen svar på spørsmål for hvorfor metoden gir så lave treff. Denne delen av samlingen blir derfor kalt for ”veldig liten” samling, for å vise at dette er en alt for liten samling i forhold til hva som er vanlig å teste med.

Presentasjonen av resultater er gjort på samme måte som for hele samlingen. Reelle treff vises i en tabell og prosentvise treff vises i en annen der også gjennomsnittet for testdelene presenteres. For hvert treningssett vises et diagram over resultatene fra den prosentvise tabellen.

5.2.1 Treningssett 1

Treningssett 1 viser et merkverdig resultat ved at testdel 4 har høyere treffprosent fra og med det andre tema og opp til femte tema. Med 5 tema i treffet har alle testdelene unntatt del 2 større treffprosent enn treningssettet.

Treningssett 1	1	2	3	4	5	Totalt
Del 1	10	13 (+3)	13 (0)	17 (+4)	20 (+3)	25
Del 2	4	6 (+2)	10 (+4)	17 (+7)	20 (+3)	25
Del 3	8	11 (+3)	12 (+1)	19 (+7)	21 (+2)	25
Del 4	9	16 (+7)	19 (+3)	19 (0)	22 (+3)	25

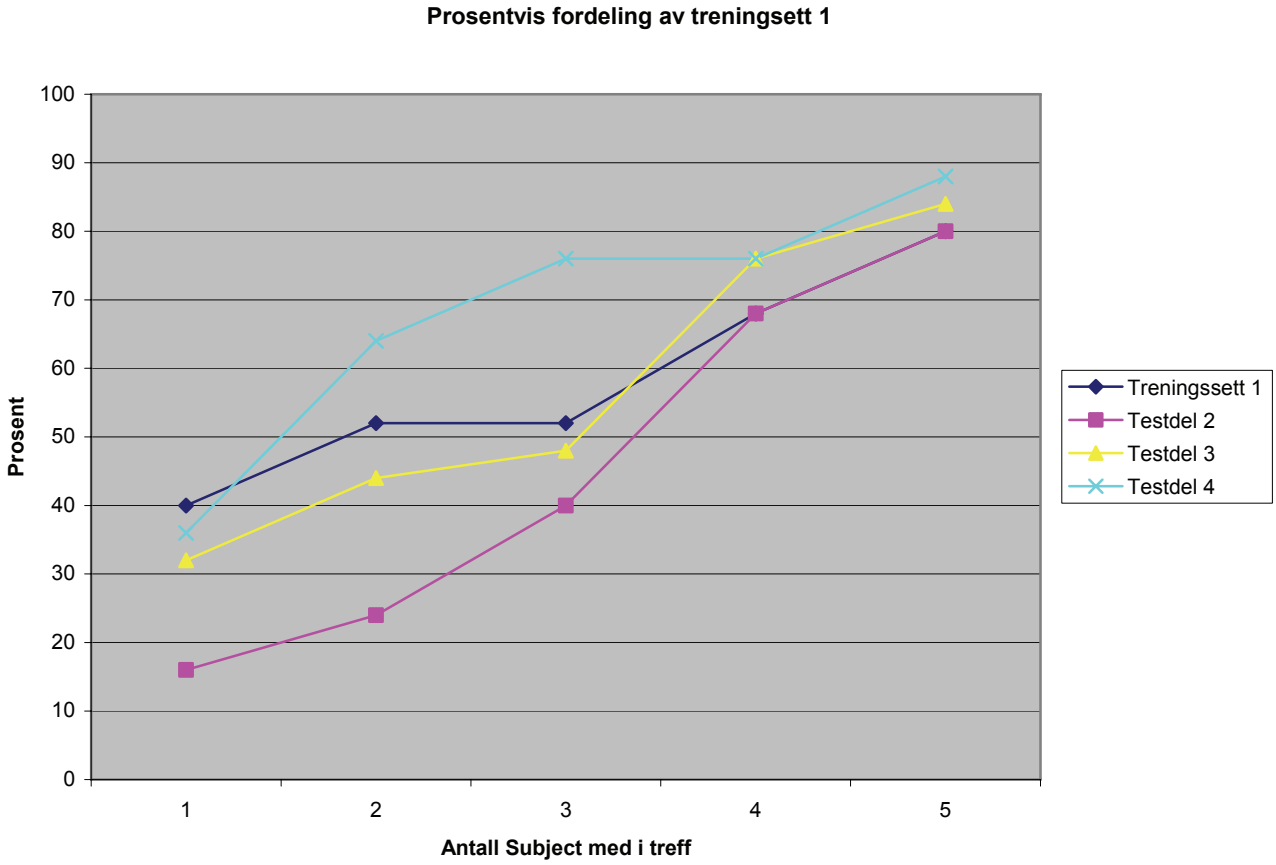
Tabell 5-IX Reelle treff for veldig liten samling, treningssett 1.

Tabell 5-IX Viser de reelle treffene for den veldig lille delen av samlingen. Her er treffene ved fem subject med i treff omtrent like for både testdelene og treningssettet. Bare treningssettet begynner med høye treff på første subject. Dette er naturlig siden denne delen er trent for subjectene.

Treningssett 1,	1	2	3	4	5
Del 1	40	52	52	68	80
Del 2	16	24	40	68	80
Del 3	32	44	48	76	84
Del 4	36	64	76	76	88
Gjennomsnitt	28	44	54,7	73,3	84

Tabell 5-X Prosentvis fordeling veldig liten samling

Tabell 5-X Viser prosentvise treff for treningssett 1. Legg merke til hvor mye høyere gjennomsnittet for testdelene i denne veldig lille samlingen er i forhold til hele samlingen. Særlig når fem subject er med i treffet blir gjennomsnittet for testdelene store.



Figur 20 Diagram, treningssett 1, veldig liten samling.

Figur 20 Viser diagram over prosentvis fordeling av treningssett 1. Legg merke til hvor mye brattere kurvene for testdelene og treningssettet stiger enn for hele samlingen. Det er ikke stor forskjell mellom treningssett og testdeler i når fem subject er med i treffet.

5.2.2 Treningssett 2

Treningssett 2 har meget gode resultater, både for treningssettet mot seg selv og for resterende testdeler. Et ganske stort avvik for testdel 4 gjør at gjennomsnittet blir trukket en del ned.

Treningssett 2	1	2	3	4	5	Totalt
Del 1	7	10 (+3)	16 (+6)	22 (+6)	22 (0)	25
Del 2	21	23 (+2)	25 (+2)	25 (0)	25 (0)	25
Del 3	9	13 (+4)	14 (+1)	19 (+5)	21 (+2)	25
Del 4	3	5 (+2)	8 (+3)	14 (+6)	14 (0)	25

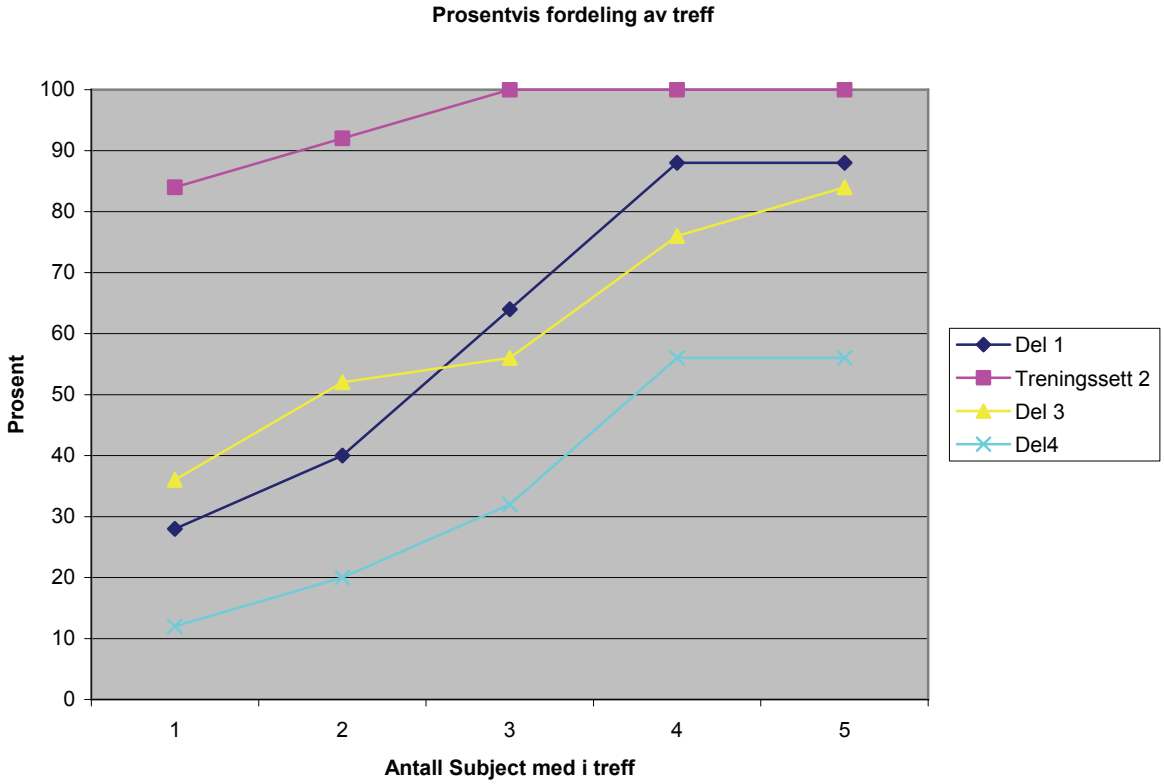
Tabell 5-XI Treningssett 2, reelle treff veldig liten samling

Tabell 5-XI Viser reelle treff for treningssett 2 veldig liten samling. Legg merke til at treningssettet scorer 25 av 25 fra tre til fem subjecter. Testdel 4 noe dårligere treff enn de andre testdelene.

Treningssett 2	1	2	3	4	5
Del 1	28	40	64	88	88
Del 2	84	92	100	100	100
Del 3	36	52	56	76	84
Del 4	12	20	32	56	56
Gjennomsnitt	25,3	37,3	50,7	73,3	76

Tabell 5-XII Treningssett 2, prosentvis fordeling, veldig liten samling

Tabell 5-XII Viser prosentvis fordeling av treff for treningssett 2. Testdel 4 trekker ned gjennomsnittet for testdelene noe, men det er fortsatt gode resultater for denne delen av samlingen og.



Figur 21 **Diagram, prosentvis fordeling av treff, veldig liten samling**

Figur 21 viser et diagram over prosentvis fordeling av treff for treningssett 2. Vi ser her at treningssettet mot seg selv scorer hundre prosent fra tre subject og ut. Testdel 1 og 4 scorer jevnt over bra, i forhold til treningssett 1. Del fire ligger litt under de to andre delene.

5.2.3 Treningssett 3

Treningssett 3 har også generelt gode resultater på linje med resultatene til treningssett 2.

Treningssett 3	1	2	3	4	5	Totalt
Del 1	4	8 (+4)	12 (+4)	14 (+2)	16 (+2)	25
Del 2	15	18 (+3)	18 (0)	18 (0)	19 (+1)	25
Del 3	16	20 (+4)	22 (+2)	22 (0)	23 (+1)	25
Del 4	12	12 (0)	16 (+4)	16 (0)	17 (+1)	25

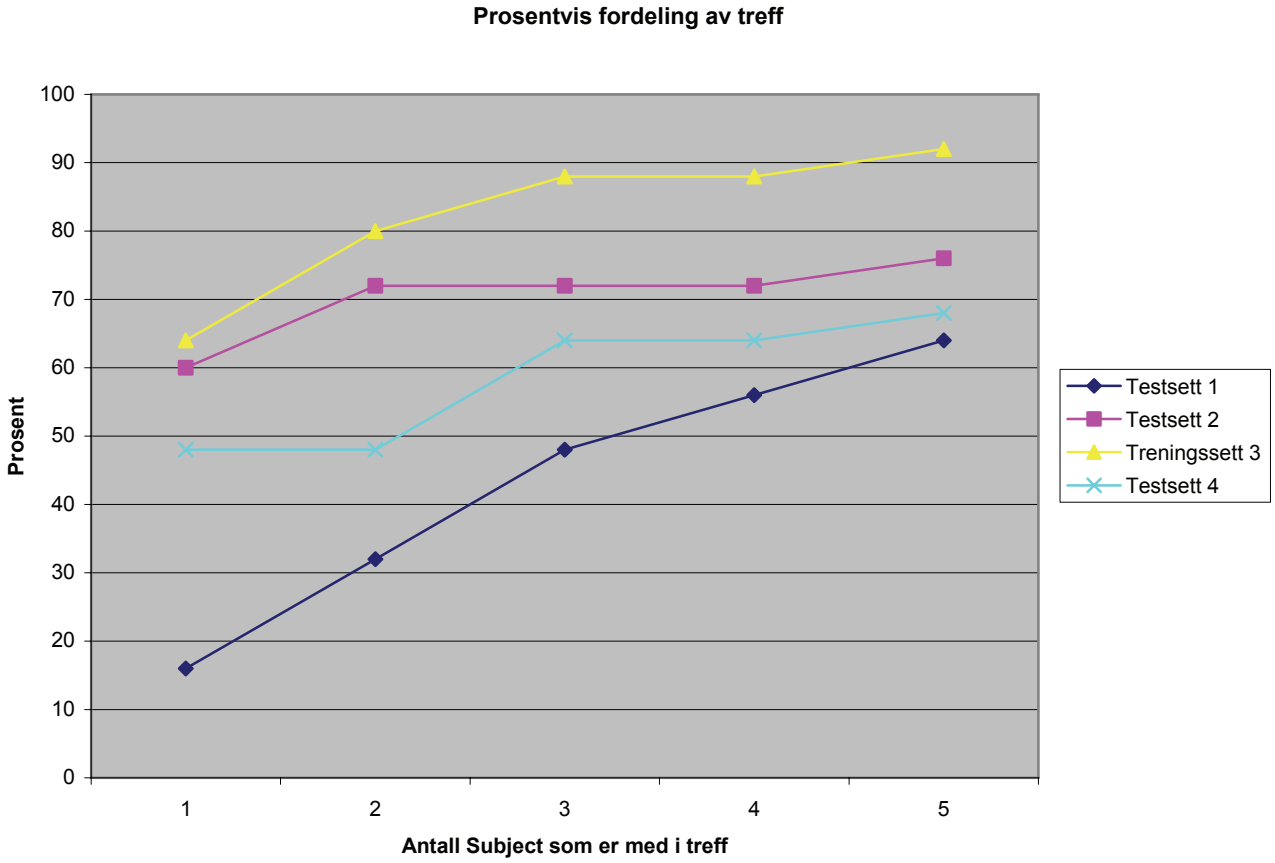
Tabell 5-XIII Treningssett 3 reelle treff, veldig liten samling

Tabell 5-XIII viser reelle treff for treningssett 3. Legg merke til testdel 2 som ikke stiger noe særlig etter andre subject som er med i treff. Generelt er treffene for testdelene ganske lave i forhold til resultatene i fra treningssett 2.

Treningssett 3	1	2	3	4	5
Del 1	16	32	48	56	64
Del 2	60	72	72	72	76
Del 3	64	80	88	88	92
Del 4	48	48	64	64	68
Gjennomsnitt	41,3	50,7	61,3	64	69,3

Tabell 5-XIV Treningssett 3 prosentvis fordeling, veldig liten samling

Tabell 5-XIV viser den prosentvise fordeling av treff for treningssett 3. Legg merke til at gjennomsnittet for testdelene er noe lavere enn for de to første treningssettene. Det er også litt merkverdig at testdel 1 begynner med så lav treffprosent som 16 på første subject. Denne delen ender opp på 64 prosent ved fem subject med i treff, som i forhold til i de andre treningssettene er noe lavt.



Figur 22 **Diagram, prosentvis fordeling treningssett 3, veldig liten samling.**

Figur 22 Viser diagram over prosentvis fordeling for treningssett 3 for denne veldig lille delen av samlingen. Her får vi visualisert at testsett 1 ligger en del under de to andre testdelene. Treningssettet testet mot seg selv scorer litt lavere enn de to foregående treningssettene.

5.2.4 Treningssett 4

Treningssett 4 har merkverdige resultater, testdelene har få treff på de fire første temaene men kurven stiger bratt fra fire til fem tema/subject. Mens treningsdelen ligger likt med testdel 1, testdel 2 og 3 ligger mye høyere når fem tema er med i treffet.

Treningssett 4	1	2	3	4	5	Totalt
Del 1	0	0 (0)	4 (+4)	4 (0)	9 (+5)	25
Del 2	0	1 (+1)	5 (+4)	6 (+1)	19 (+13)	25
Del 3	0	1 (+1)	5 (+4)	5 (0)	13 (+8)	25
Del 4	6	6 (0)	7 (+1)	8 (+1)	9 (+1)	25

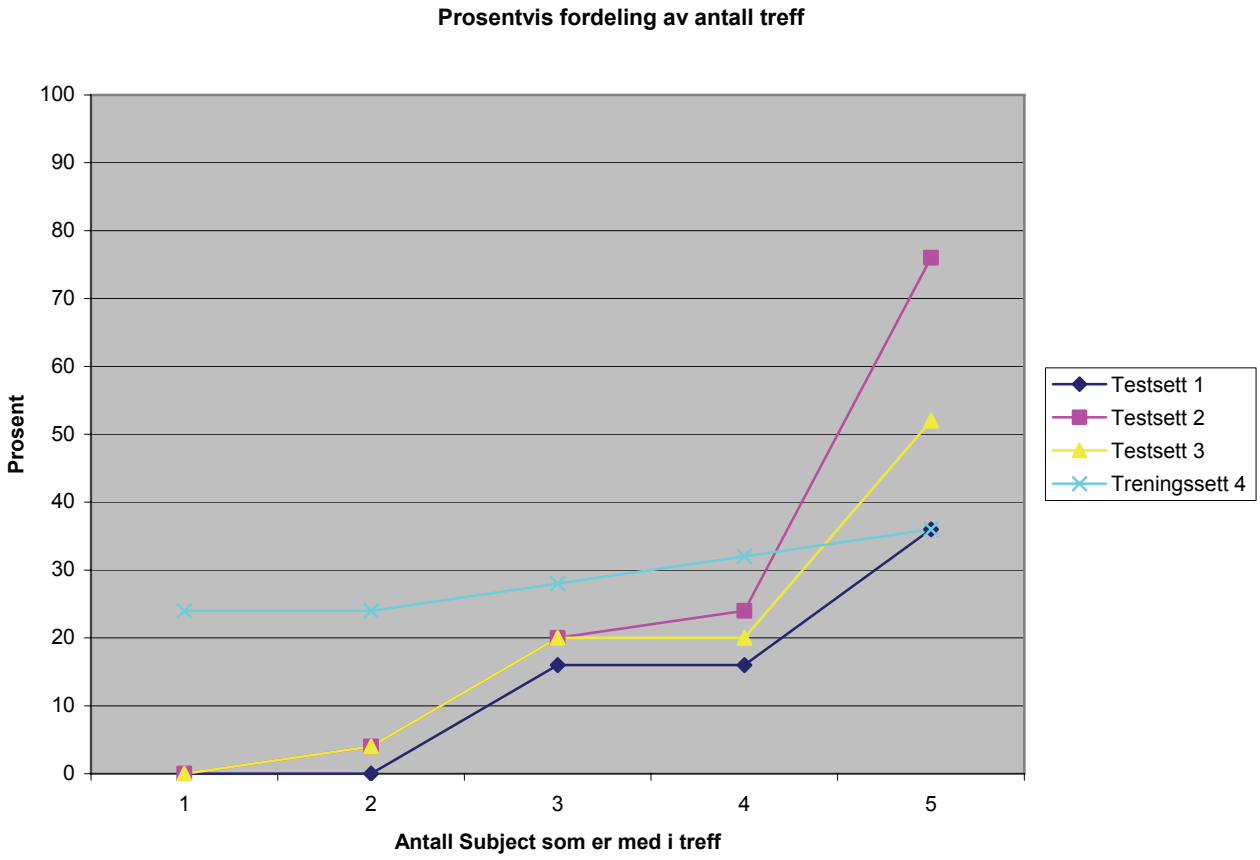
Tabell 5-XV Treningssett 4, reelle treff, veldig liten samling

Tabell 5-XV viser de reelle treff for treningssett 4. Her er flere merkverdigheter. Først ser vi at det i forhold til de andre treningssettene er få treff generelt både for treningssett mot seg selv og for testdelene. Særlig testdel 1 er veldig lav, men mer merkverdig er det at treningssett 4 mot seg selv scorer dårligere enn testdelene 2 og 3. Dette gjelder på alle antall subject. Testdel 2 og testdel 3 begynner med ingen treff ved første subject, men henter seg inn igjen ved femte subject.

Treningssett 4	1	2	3	4	5
Del 1	0	0	16	16	36
Del 2	0	4	20	24	76
Del 3	0	4	20	20	52
Del 4	24	24	28	32	36
Gjennomsnitt	8	10,7	22,7	25,3	54,7

Tabell 5-XVI Treningssett 4, prosentvis fordeling, veldig liten samling.

Tabell 5-XVI Viser den prosentvise fordelingen av treningssett 4. Her ser vi enda tydeligere hvor dårlig resultatene er generelt for dette treningssettet. Ved femte subject er treffprosenten steget noe, men det er mellom tredje og femte subject at treffprosenten begynner å øke mye.



Figur 23 **Diagram, prosentvis fordeling av treff treningssett 4 veldig liten samling.**

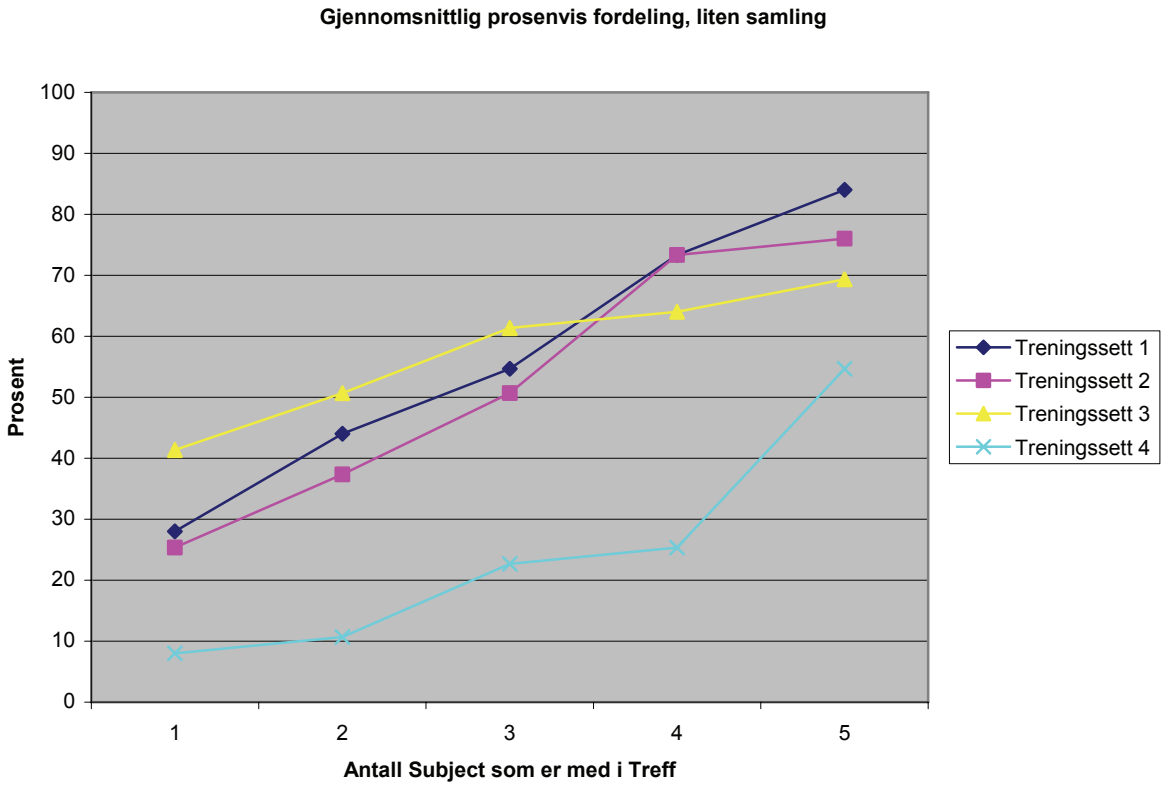
Figur 23 viser diagram over prosentvis fordeling av treff for treningssett 4. Her blir de merkverdige resultatene visualisert. Treningssettet stiger jevnt for hvert subject som blir lagt til i treffet. Testsett 2 her den største stigningen fra null til i underkant av 80 prosent ved femte subject. Testsett 3 ender opp på i overkant av 50 % ved femte subject og testsett 1 ender opp likt med treningssettet på 36 %. Det er meget merkverdig at treningssettet ligger under to av testdelene ved 5 subject.

5.2.5 Gjennomsnitt veldig liten samling

Gjennomsnittet for testdelene til treningssettene for det veldig lille utvalget av samlingen er generelt mye bedre enn for test mot hele samlingen. Mulige årsaker til dette kan være:

- *At enkelte dokumenter inneholder tekst som er på annet språk enn det som er angitt.*
- *At temaene ikke er spredd jevnt utover samlingen slik at de ulike delene har tema som overlapper til de andre delene.*
- *At ordlisten inneholder mange utenlandske ord, som konsekvens av at enkelte dokumenter ikke er merket med riktig språk. Jamfør kapittel 4.3 der samlingen som er brukt for test av metoden er beskrevet.*
- *At ordlisten inneholder mange ord som ikke gir semantisk innhold såkalte stoppord..*

Disse årsakene og måter å bruke metoden på er beskrevet og diskutert i neste kapittel.



Figur 24 Diagram, gjennomsnitt veldig liten samling

Figuren viser gjennomsnittene for den veldig lille delen av samlingen. Treningssett 4 ligger mye under de andre treningssettene, dette samsvarer godt med resultatene som treningssett 4 ga. I de tre andre treningssettene øker treffene jevnt og avviker ikke mye fra hverandre, dette samsvarer godt med det generelle inntrykket fra resultatene i tabellene til disse delene. Legg også merke til hvor mye høyere gjennomsnittet er for den veldig lille samlingen i forhold til for hele samlingen, se Figur 19 for sammenligning mellom snitt for veldig liten samling og hele samlingen.

6 Evaluering og diskusjon

Det har vært en utfordrende oppgave å lage en metode som skal finne automatisk tema for dokumenter. Og det har til tider vært mange store utfordringer, i dette kapittelet vil jeg prøve å gi en fullstendig vurdering og diskusjon rundt arbeidsprosessen og resultatene av arbeidet.

6.1 Oppsummering av arbeidet

Oppgaven var i utgangspunktet nokså ambisiøs, å finne en metode som forhåpentlig vil hjelpe til å gjøre internett mer gjennomtrengelig og oversiktlig er et ambisiøst prosjekt. Litteraturstudien bekreftet dette, det er gjort enormt mye arbeid innen området for å sortere informasjon og kategorisere dokumenter. Men felles for de fleste av slike metoder er at de enten krever mye av utvikleren av websidene, eller mye av utvikleren av systemet for kategorisering. Jamfør Semantisk web og Sebastinis system for kategorisering ved bruk av maskinlæring. [3, 23].

Målet for mitt arbeid var derfor å finne en metode som ikke tar i bruk maskinlæring og lignende teknikker, og som ikke krever mye jobb av dem som publiserer dokumenter. En metode som lar dokumentene være slik de er og som bruker metadataene til dokumentene for å finne tema. Fordelen med en slik metode er at den vil være bakover kompatibel og enkel å ta i bruk for de fleste. Ulempen er at den kan bli for enkel og ikke kunne klare å angi riktig tema. Min oppgave har derfor vært å finne en slik metode og teste om den fungerer.

For å løse oppgaven har jeg brukt åpen kildekode programvare for å høste metadataene jeg trengte, som beskrevet i kapittel 3.var det ikke helt trivielt å få programvaren til å kjøre som planlagt, men til slutt virket høsteren og det var bare å velge en passende dokumentsamling for å teste metoden. Før metoden kunne implementeres måtte jeg finne en metode for å presentere dokumenter og tema på en målbar måte, vektormetoden ble valgt på grunn av dens enkelhet og nøyaktighet. Så måtte jeg finne ut hvordan en vektor for tema skulle representeres, ved å bruke deler av en clusteringmetode kalt k-means fant man gjennomsnittsvektorer for et tema ved å ta gjennomsnittsvektorene for dokumentene som hørte til under temaene. For å få testet metoden ble samlingen delt opp i flere deler og hver del av samlingen fungerte som treningsdel, der temaene ble trenet med de gjeldende dokumentene for hvert tema. De andre delene av samlingen fungerte som testsett.

Alt i alt har arbeidet med prototypen gått greit, det har ikke vært noen store forsinkelser, på linje med installering av åpen kildekodeprogrammet Arc, og prototypen kjører og gir antall treff på tema som er nærmest dokumentet, og antall treff der de to beste temaene er med, tre beste, fire beste, helt til det femte beste temaet.

Dessverre har det vist seg at resultatene ikke er så gode som regnet med, selv etter mange testkjøringer og studering av kode, blir resultatet det samme. I resten av kapittelet skal jeg derfor redegjøre for resultatene fra testing, gi forslag til mulige årsaker og angi hvilke forbedringer som kan gjøres for å gi bedre resultater.

6.2 Resultat og mulige feilkilder

Som kapittel 4 beskriver, er ikke resultatene fra kjøringen veldig gode. Jeg vil nå oppsummere resultatene og gi noen vinklinger på hvordan metoden for automatisk tema kan bli brukt og hva som kreves av treffprosenten for at metoden kan ha troverdighet. Det er generelt vanskelig å si noe om resultatene er gode eller dårlige, da jeg ikke har funnet noen publikasjoner som sier noe om hvor gode resultater man kan forvente. Derfor er diskusjonen om resultatene er gode eller dårlige basert på mine anslag og hva som virker troverdig for at metoden skal kunne brukes.

Først vil jeg gi noen vinklinger på resultatene ut i fra hvordan man kan regne et godt resultat, middels resultat og et dårlig resultat, og hvordan resultatene kan variere ut i fra hvordan man bruker dem.

6.2.1 Hvordan kan metoden brukes

Siden ambisjonen for å finne metoden var at den kunne brukes til å sette tema til for eksempel søkeresultater på internett, eller gi et tema bare ut i fra en tittel og et sammendrag, vil man være avhengig av å kunne stole på at metoden finner riktig tema i de aller fleste tilfeller. Jeg har ikke funnet noen kilder på hvor god treffprosenten bør være for å kunne angi tema bastant for et dokument. Jeg vil derfor tippe at man må ha bortimot hundre prosent treff for å kunne sette tema. Dette kan selvfølgelig diskuteres, og det er mange faktorer som spiller inn, for eksempel har jeg ikke sagt noe om hvordan man angir tema til et søkeresultat, eller hvordan man organiserer søkene.

En måte å sette automatisk tema, er å angi sannsynligheten for at et dokument har et tema, for eksempel ved at man søker etter et tema og får opp alle dokumenter har en viss ”score” på dette temaet. Man kan for eksempel rangere dokumentene etter hvor like de er med temaet. Dersom temaet ikke finnes i treningssett eller samling, vil resultatet bli så dårlig eller tilfeldig at man antagelig ikke vil få opp noen treff. Ved å bruke denne metoden er man ikke så avhengig av at treffprosenten mellom dokument og tema blir så nært hundre som mulig. Man overlater til brukeren å vurdere hvor nært et dokument må være et tema. Brukt i en slik sammenheng kan min metode kanskje brukes som et hjelpemiddel, men siden resultatene i gjennomsnitt ligger på under 20 % for alle treningssettene, når fem tema er tatt med, se Figur 19, vil det neppe være fornuftig å ta i bruk metoden slik den er nå.

Dersom man bruker metoden slik som prototypen er laget, ved at man angir treff eller ikke på et dokument, kan man bruke metoden til å angi tema for et dokument i en søkemotor, ved at bruker skriver inn tittel eller deler av tittel og et tema eller en gruppe temaer blir angitt for dokumentet. Det var i utgangspunktet slik jeg så for meg at metoden kunne brukes, men også da må treffprosenten være mye høyere for at man skal kunne stole på den. Jeg vil tippe at man bør ha en treffprosent på mellom 70 og 100 for at det skal være aktuelt å ta metoden i bruk på denne måten.

6.2.2 Del 1, gir dårligere resultater enn resten av samlingen

Tallene fra kapittel 5.1 gir ikke stor grunn til optimisme for metoden, gjennomsnittet for alle treningssettene ligger på under 20 %, og alle fem foreslåtte tema er tatt med. Ved å studere del 1 av samlingen både som treningssett og testdel, ser man at denne delen ligger resultatene en del under de andre delene, og den er med på å trekke ned gjennomsnittet.

Hvorfor har denne delen så mye dårligere resultater enn de andre delene? Er det noe spesielt med denne delen av samlingen som gjør at resultatet blir så dårlig? Det siste spørsmålet er greit å begynne med, og for å gi svar på om det er noe spesielt med denne delen av samlingen studerte jeg samlingen slik den er i databasen, faktorer jeg studerte var om dokumentene i denne delen av samlingen hadde annen datostempling enn resten av samlingen, om det var forskjell i tema feltet (”subject”), eller om tittel, type eller description skilte seg ut. Siden datofeltet er satt til null på alle dokumentene er det umulig å si om dette spiller inn på resultatet for del 1 av samlingen. Title og description er felter som ikke skiller seg mer ut enn beskrevet i kapittel 4.3, ved at det forekommer et og annet dokument som annet språk enn de andre dokumentene. Type feltet skiller seg heller ikke noe særlig ut. Da er det bare Subject feltet igjen, ved første øyekast er feltene like, men det viser seg ved nærmere ettersyn at det er en forskjell i hvilke Subject som er med i begynnelsen av samlingen og hvilke som er med på slutten.

Dette kan forklare hvorfor resultatene for del 1 er så dårlige i forhold til resten av samlingen, fordi det da ikke vil være noen Subject som passer til dokumentene for resten av samlingen, verken som treningssett eller som testdel for de andre delene av samlingen. Dersom det er slik at de ulike delene av samlingen har Subject som ikke overlapper hverandre og som er ulike, vil dette antagelig være grunnen til det dårlige resultatet for metoden. Siden jeg ikke har hatt tid til å studere tabellen i detalj og fått dannet meg et inntrykk av hvordan fordelingen av subject faktisk er kan jeg ikke bastant slå fast at dette er grunnen for de lave treffprosenten for metoden. Se Figur 25 for et visuelt inntrykk av databasen.

6.2.3 Angivelse av språk

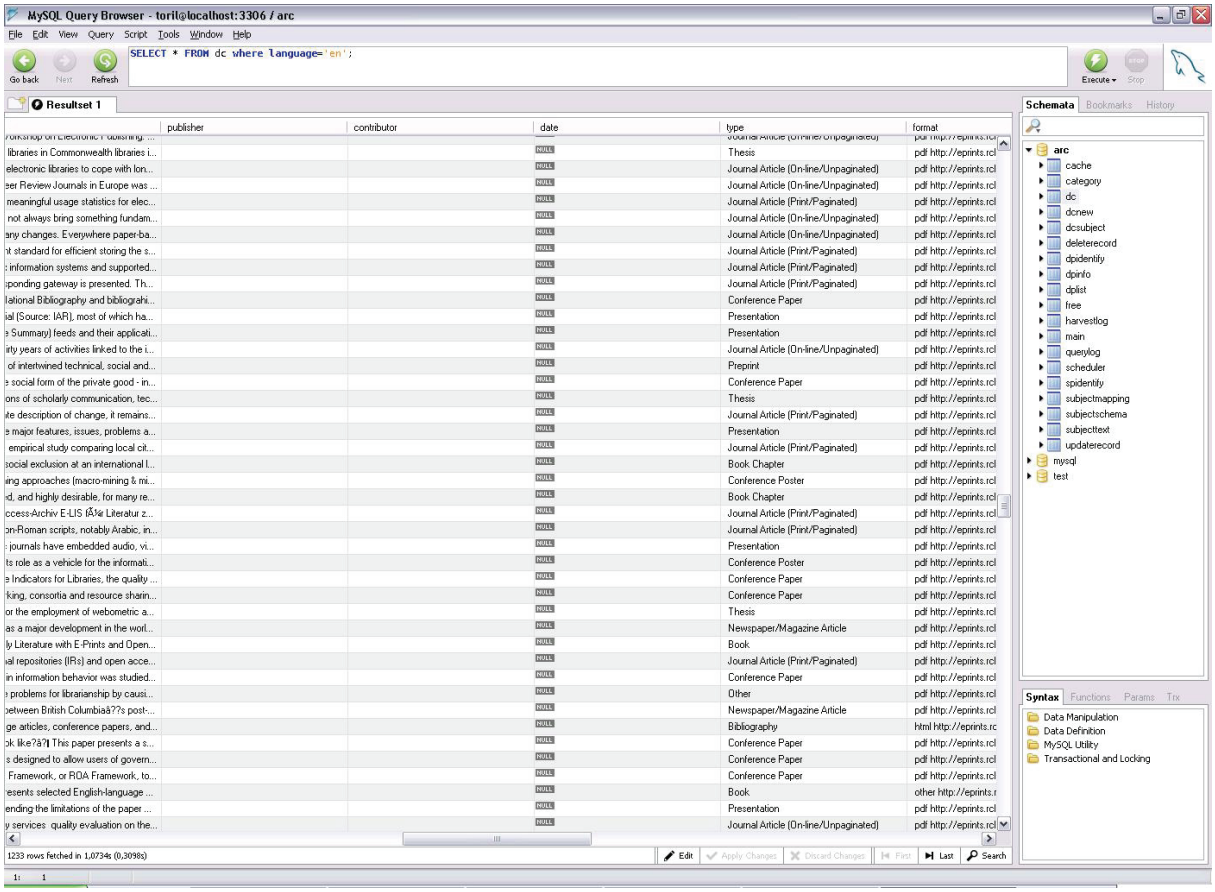
Som beskrevet i kapittel 4.3 om samlingen som er brukt for testing, er det enkelte dokument som ikke er på angitt språk i ”language” feltet. Noen av dokumentene er angitt i ”description” feltet med for eksempel [Abstract in German:] og da er sammendraget i ”description” på tysk.

Hva dette har å si for resultatet av sammenligningene er uvisst, og jeg har heller ikke klart å finne ut nøyaktig omfanget av hvor mange dokumenter det gjelder. Ut i fra det jeg kan se ved å bla i tabellen har jeg funnet antallet til å ligge et sted mellom 10 og 20 dokumenter. Dette gjør at ordlisten blir svært stor, siden ”description” feltet kan være svært stort og inneholder mye tekst, blir det mange ekstra ord i ordlisten. Effekten av dette blir at alle vektorer får mange flere elementer enn nødvendig, utregningene kan bli tregere og nøyaktigheten mindre. Men jeg kan ikke påvise at dette har noe å si for hvor godt metoden treffer riktig tema. Dette fordi det er så få dokumenter det gjelder at det ikke vil påvirke gjennomsnittsvektorene i stor grad. Dette basert på at 10 til 20 dokumenter av totalt 1233 ikke er nok til å påvirke resultatet.

En mulig løsning for problemet med feil angitt språk vil være å lage et system som tester om dokumentene har riktig språk, ved å sammenligne mot en standard ordliste i angitt språk.

6.2.4 Ulikhet mellom dokumentene

De ulike dokumentene i samlingen består blant annet av presentasjoner, publiserte artikler, upubliserte artikler, artikler fra aviser, avhandlinger, tekniske rapporter og kapitler fra bøker. Dette at dokumentene er så ulike i type fører til at "description" feltet blir brukt til ulike formål, for eksempel blir det for artikler brukt til sammendrag, mens for presentasjoner og tekniske rapporter bare inneholder stikkord. Dette gjør at lengden på "description" feltet varierer veldig, og dette har betydning for antall elementer som er med i vektorene. Hvor mye dette påvirker resultatet for metoden er jeg usikker på, men det hadde antagelig vært ideelt å hatt en samling der de fleste dokumentene var av samme type. Slik at "description" feltet bestod mest mulig av kun sammendrag.



Figur 25 Database printscreen

6.2.5 Oppsummering av resultater

Jeg har definert resultatene som dårlige på grunn av at det ikke er mulig å bruke metoden verken til å angi tema for et dokument, eller for å angi i hvor stor grad et dokument har et tema. Dette var skuffende resultater, men det er mye som tyder på at det er fordelingen av temaene slik samlingen er lagret i databasen, og problemet med dokumenter med annet språk enn angitt som er grunnen.

I arbeidet med å finne grunner for de dårlige resultatene ble det gjort en del tester av metoden ved å bruke en veldig liten del av samlingen, lage en egen ordliste for denne delen av samlingen og dele opp samlingen og kjøre den på samme måte som for hele samlingen.

Resultatene var oppsiktsvekkende og oppmuntrende, i kapittel 5.2 blir resultatene presentert, kort oppsummert ser en at metoden treffer på 80-90 % på dokumentene når man har med alle fem angitte tema. Dersom metoden hadde hatt denne treffprosenten for hele samlingen ville jeg anbefalt metoden, både for å sette tema til dokumenter slik jeg gjør og angi dokumenter med et mulig tema.

Resultatene for samlingen når bare 100 dokumenter er med er jevnt over mye bedre enn for hele samlingen, unntatt for treningssett 1 og 4. Her er avviker resultatene stort i fra de andre delene av samlingen. Blant annet har treningssettet dårligere resultater når fem tema er med enn testsett 2 og 3. Se Figur 23 og Figur 20 i kapittel 5 for visuell framstilling av resultater for disse treningssett. De to andre treningssettene har resultater mer som forventet ved at treningssettene testet mot seg selv har mye større treffprosent enn testdelene.

Det positive med resultatene er at treffprosenten er så mye høyere enn for hele samlingen. Dette kan tyde på at det er egenskaper ved samlingen og ikke prototypen, eller metoden jeg har funnet, som gjør at treffprosenten blir lav for hele samlingen. I de 100 første dokumentene er fordelingen av tema mye bedre enn for hele samlingen, fordi dokumentene i de forskjellige delene innehar i mye større grad de samme temaene, det er ikke slik at siste delen har helt ulike temaer som første delen av samlingen. Hele samlingen har som kjent en ganske stor variasjon i tema, slik at siste delen ikke har mange felles tema med første del. Se tidligere dette kapittel.

I tillegg til å ha bedre fordeling på temaer innehar ikke denne delen, etter det jeg kan se, noen dokumenter med språk som ikke er angitt. Derfor blir ordlisten mer konsekvent for denne delen av samlingen og vektordimensjonen blir mye mindre. Om dette har noe å si for treffprosenten er jeg som kjent usikker på, men at det ikke teller negativt inn er jeg ganske sikker på.

Kort oppsummert vil jeg si at det antagelig er egenskaper ved samlingen som gjør at treffprosenten blir så lav og ikke metoden i seg selv. I neste avsnitt vil jeg derfor foreslå noen endringer i hvordan man bruker samlingen, og en beskrivelse hvordan en ideell samling burde vært, samt noen endringer i metoden som kanskje vil gjøre resultatene bedre.

6.2.6 Mulige forbedringer

Siden systemet som er blitt laget er en prototyp av metoden og denne kan inneha feil og mangler vil jeg nå foreslå forbedringer som kan gjøre at metoden får en høyere treffprosent på riktig valgt tema.

I de foregående avsnitt er det avdekket egenskaper med valgte samling, E-LIS, som gjør at jeg mistenker denne for å være hovedgrunnen for de dårlige treffprosentene.
Et enkelt tiltak for å løse problemet med at temaene ikke er jevnt fordelt over hele samlingen er å stokke om på dokumentene slik at dokumenter med samme tema ikke lenger ligger samlet. Erfaringen fra testing med 100 dokumenter i samlingen tilsier at man da antagelig vil få mye bedre resultater. Kort oppsummert må man gjøre noe for å få bedre kontroll over samlingen.

Siden samlingen inneholder en del dokumenter som ikke har riktig angivelse av språk, vil det være naturlig å gjøre noe med ordlisten. Her er det flere alternativer, man kan bruke en engelsk standardisert ordbok til formålet. Fordelen ved å gjøre det på denne måten er at den er

bearbeidet og man er sikker på at et ord bare opptrer en gang. Ulempen er at man ikke kan vite om alle faguttrykk er med i ordlisten. En annen måte er å ikke la stoppord bli lagt til i ordlisten. Dette er antagelig det tiltaket som vil ha best effekt på treffprosent og ytelsen til programmet, fordi vektordimensjonene blir mye mindre og utregningene på vektorene vil da gå raskere. Treffprosenten vil antagelig bli høyere når det bare er ord med semantisk mening som står igjen.

Som nevnt i kapittel 4.2.4 er ordlisten ikke basert på stemming form. I følge Baeza-Yates er det flere i den senere tid som tviler på om dette vil gjøre resultatene bedre fordi man går glipp av kontekster i språket da grunnformen av et ord ofte ikke sier noe om konteksten til ordet. I tillegg til at ordlisten må være på stemming form må man også gjøre om teksten i dokumentene til stemming form før man lager vektorer. Dette bryter med prinsippet mitt om at man ikke skal endre noe av innholdet i dokumentene for å implementere metoden. For å testformål kan man implementere stemming og sjekke om dette øker treffprosenten. Dersom det viser seg at stemming er avgjørende for om metoden skal treffe så godt som jeg foreslår i kapittel 6.2.1, blir metoden slik jeg ser det for tungvindt å ta i bruk til at den vil få gjennomslag på internett.[4]

Det kan være gunstig å teste flere samlinger, da først vil man virkelig se hva metoden er god for. I tillegg kan det være gunstig å forbedre slik at antall aksesser til database blir så lave som mulig og optimalisere utregninger med vektorer. Dette har ikke noe å si for hvor godt metoden vil fungere, men kjøring av prototypen vil gå raskere og det vil være nødvendig når samlingene blir mye større enn E-LIS.

I neste kapittel vil konklusjonen oppsummere de viktigste delene av dette kapittelet og videre arbeid vil angi hva jeg anbefaler som videre arbeid for å videreutvikle metoden.

7 Konklusjon og videre arbeid

7.1 Konklusjon

Målet med arbeidet mitt var å finne en metode som kan angi tema for et tekstlig dokument automatisk. I tillegg var kravet for metoden at den skulle være så enkel i bruk at man ikke trenger å gjøre noen forandringer i dokumentet, og selve implementeringen skulle være enkel og ikke ta i bruk avanserte teknikker som maskinlæring.

I korte trekk baserer metoden seg på at man skal finne tema for et dokument, basert på tittelen og sammendraget (abstract) for dokumentet. Ved å sammenligne med de tema som er gitt for dokumentene på forhånd, kan man finne ut om metoden finner riktig tema.

Prototypen, som implementerte metoden, ga svært få treff på riktig tema. I gjennomsnitt under 20 prosent når de fem nærmeste tema var tatt med. I testfasen av prototypen ble det kjørt tester med et utvalg på de 100 første dokumentene i samlingen, siden resultatene for disse testene var mye bedre enn for hele samlingen ble det studert grundig hva som skilte den veldig lille delen av samlingen fra resten samlingen. Ulikheten i resultatet indikerte at det var noe med egenskapen til samlingen som gir lav treffprosent.

Det er særlig to faktorer som spiller inn. Det faktum at samlingen inneholder dokumenter som avviker fra angitt språk og at temaene for dokumentene ikke er jevnt fordelt på hele samlingen.

Siden ordlisten inneholder noen få dokumenter som ikke samsvarer med angitt språk, fører det til at ordlisten blir veldig stor og som igjen fører til at vektorene får flere dimensjoner enn nødvendig. Det er vanskelig å si eksakt hvor mye dette hadde å si for resultatet. I den utvalgte delen av samlingen var ingen av tekstene av annet språk enn det som var angitt, og resultatene var mye bedre for denne delen av samlingen. Det antas at det var snakk om 10 til 20 dokumenter som ikke hadde riktig språk, og av en samling på 1233 dokumenter blir dette en veldig lav andel av dokumentene, så jeg vil konkludere med at dette ikke har hatt stor betydning for den dårlige treffprosenten.

I den veldig lille delen av samlingen var de fleste temaer representert i alle treningssettene, dette gjorde at de fleste sammenligninger ville det reelle tema for et dokument være en del av de tema det ble gjettet på. Siste treningsdel avvek noe fra resten av treningsdelene og temaene her var en stor del av dem ikke med i de resterende delene, dette førte til at treffprosenten for denne delen ble veldig lav i forhold til resten. Dette indikerer at det er fordelingen av temaene som gjør at treffprosenten blir så lav for hele samlingen. Ved bedre kontroll på samlingen vil antagelig treffprosenten til prototypen bli bedre enn nå.

På grunnlag av de gode resultatene for den veldig lille samlingen, der temaene var mer jevnt fordelt vil jeg konkludere med at metoden vil kunne fungere dersom man har bedre kontroll over samlingen. Denne kontrollen innebærer at tema blir fordelt jevnt over hele samlingen og at man sjekker at dokumentene har riktig språk i forhold til det angitte språket. Metoden vil slik den er i dag, ikke kunne implementeres i stor skala i tilknytning til søkemotorer eller som tjeneste på internett.

Det finnes, slik jeg ser det, stort potensial i metoden til at den kan bli bedre og kanskje brukes i stor skala på internett. I videre arbeid er en prioritert rekkefølge gitt for hvordan man kan utbedre metoden slik at den forhåpentlig vil fungere for internett.

7.2 Videre arbeid

I kapittel **Feil! Fant ikke referansekilden.** er mulige forbedringer for metoden foreslått og diskutert. Jeg vil derfor angi videre arbeid med metoden i den rekkefølgen jeg prioriterer er viktigst for at metoden skal gi flest mulig treff på riktig tema.

- Få kontroll over samlingen. Den mest sannsynlige feilen er som kjent i fra diskusjon og konklusjon at samlingen slik den ligger i databasen fordeler tema ujevnt.
- Fjerne stoppord fra ordlisten.
- Sjekke at angitt språk for hvert dokument samsvarer med det reelle språket.
- Innføre stemming.

For hvert punkt man gjennomfører på listen bør man teste om det har blitt forbedringer. Dersom metoden gir så gode resultater som jeg ønsker i diskusjonskapittelet (nærmere 100 prosent) kan man prøve å implementere metoden i forbindelse med en søkemotor og teste metoden opp mot reelle brukere. Brukerundersøkelser kan da teste om metoden fungerer i stor skala på internett

8 Litteraturliste

1. *DomainTools*. 2006 [cited 28.06.06]; Web page]. Available from: http://www.domaintools.com/internet-statistics/.
2. S. Decker, et al., *The Semantic Web: The Roles of XML and RDF*. IEEE Internet Computing, 2000. **4**(5): p. 63-73.
3. F. Sebastiani, *Machine learning in automated text categorization.* ACM Comput. Surv., 2002. **34**(1): p. 1-47.
4. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. 1 ed. 1999, New York: ACM Press10:0-201-39829-X
5. W.Y. Arms, *Automated Digital Libraries.* D-Lib Magazine, 2000. **6**(7/8).
6. *Open Archieves Initiative*. 2006 [cited 30.06.06]; Available from: http://www.openarchives.org/.
7. T. Gill, A.J. Gilliland, and M.S. Woodley. *Setting The Stage*. Introduction To Metadata, Pathways To Digital Information [Web Page] 2006 [cited 2006 27.02.2006]; 2.1:[Available from: http://www.getty.edu/research/conducting_research/standards/intrometadata/index.html.
8. H.V.d. Sompel, et al., *Resource Harvesting within the OAI-PMH Framework.* D-Lib Magazine, 2004. **Volume 10 Number 12**.
9. E.H. Solvang, *Enhetlig tilgang til heterogene metadatabaser, Interoperabilitet v.h.a. OAI-PMH*. 2004, IDI, NTNU: Trondheim. p. 1-188.
10. D.L. Federation. *OAI for Beginners - the Open Archives Forum online tutorial*. [cited; Available from: http://www.oaforum.org/tutorial/.
11. W3C. *Extensible Markup Language (XML)*. [Web page] 2006 [cited 2006 12.07.06]; Available from: http://www.w3.org/XML/.
12. *The Dublin Core Metadata Initiative* 2006 [cited 27.06.2006]; Available from: http://dublincore.org/.
13. O.S. Initiative. *Open Source Initiative*. 2006 [cited 01.07.2006]; Available from: http://www.opensource.org/.
14. B. Vargheese. *Arc Software Installation* 2002 [cited 2006 01.07.2006]; Available from: http://sourceforge.net/docman/display_doc.php?docid=13495&group_id=61532.
15. MySQL. *MySQL*. [Web page] 2006 [cited 10.02.06]; Available from: http://www.mysql.com/products/database/.
16. J. Anjer, *Vektorer*. 2006, Høgskolen I Oslo. p. 32.
17. H.C. Edwards and D.E. Penney, *Calculus with analytic Geometry*. 5. ed. 1998: Prentice Hall0-13-736331-1
18. J.B. MacQueen, *Some Methods for classification and Analysis of Multivariate Observations.* Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, 1967. **1**: p. 281-297.
19. J. Hartigan and M. Wong, *Algorithm AS 136: A K-Means Clustering Algorithm.* Applied Statistics, 1979. **28**(1): p. 100-108.
20. OAI. *Registered Data Providers*. [Web Page] 2006 [cited 04.07.06]; Available from: http://www.openarchives.org/Register/BrowseSites.

21. *eprints.rclis.org*. 2006 [cited 28.06.06]; Available from: www.eprints.rclis.org.
22. S. Microsystems. *JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification* [Web page] 2003 [cited 2006 05.07.2006]; Available from: http://java.sun.com/j2se/1.4.2/docs/api/.
23. T. Berners-Lee, J. Hendler, and O. Lassila, *The Semantic Web*, in *Scientific American*. 2001.

9 Vedlegg A

Flere resultater:
Råfil:

```
*  a  *  b  *  c  *  d  *  e  *** Relevant del 1
*  48  *  51  *  54  *  58  *  61  *** 1
*  17 * 32 * 37 * 49 * 54 *** 2
*  11 * 24 * 36 * 46 * 55 *** 3
*  12 * 21 * 35 * 36 * 41 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 2
* 3 * 5 * 6 * 10 * 11 *** 1
* 82 * 84 * 88 * 91 * 92 *** 2
* 4 * 13 * 14 * 25 * 33 *** 3
* 4 * 10 * 14 * 16 * 17 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 3
* 3 * 7 * 12 * 17 * 20 *** 1
* 17 * 26 * 31 * 35 * 42 *** 2
* 91 * 99 * 103 * 112 * 117 *** 3
* 15 * 21 * 24 * 29 * 31 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 4
* 3 * 7 * 12 * 17 * 20 *** 1
* 17 * 26 * 31 * 35 * 42 *** 2
* 91 * 99 * 103 * 112 * 117 *** 3
* 15 * 21 * 24 * 29 * 31 *** 4
```

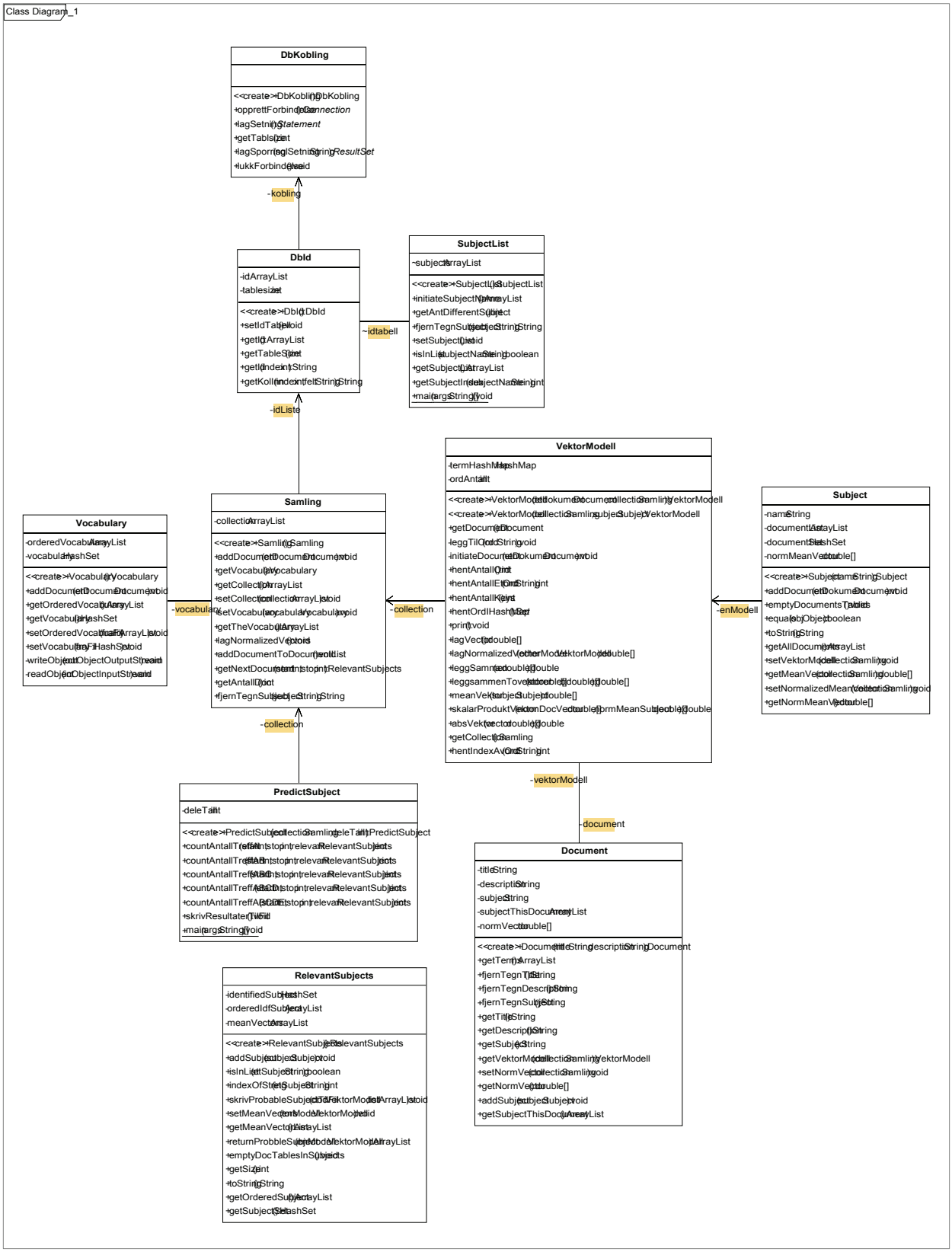
Forklaring av oppsettet over. Bokstavene a til e forklarer antall tema som dokumentene treffer på. Det første foreslåtte temaet for a, 2 første for b, 3 første for c og så videre. Det samme gjelder for resultater av hundre første dokumenter under:

```
*  a  *  b  *  c  *  d  *  e  *** Relevant del 1
*  10  *  13  *  13  *  17  *  20  *** 1
*  4 * 6 * 10 * 17 * 20 *** 2
*  8 * 11 * 12 * 19 * 21 *** 3
*  9 * 16 * 19 * 19 * 22 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 2
* 7 * 10 * 16 * 22 * 22 *** 1
* 21 * 23 * 25 * 25 * 25 *** 2
* 9 * 13 * 14 * 19 * 21 *** 3
* 3 * 5 * 8 * 14 * 14 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 3
* 4 * 8 * 12 * 14 * 16 *** 1
* 15 * 18 * 18 * 18 * 19 *** 2
* 16 * 20 * 22 * 22 * 23 *** 3
* 12 * 12 * 16 * 16 * 17 *** 4

*  a  *  b  *  c  *  d  *  e  *** Relevant del 4
```

```
* 4 * 8 * 12 * 14 * 16 *** 1
* 15 * 18 * 18 * 18 * 19 *** 2
* 16 * 20 * 22 * 22 * 23 *** 3
```
* 12 * 12 * 16 * 16 * 17 *** 4

Se elektroniske vedlegg for selve tekstfil og excel-ark som viser grafer og resultater i sin helhelt.

Vedlegg B


Class Diagram_1
DbKobling
-kobling
DbId
-idListe
~idtabell
SubjectList
Vocabulary
-vocabulary
Samling
-collection
VektorModell
-vektorModell
-enModell
Subject
PredictSubject
-collection
-document
Document
RelevantSubjects

Klassediagrammet er bare med som dokumentasjon på utført arbeid, da det var svært vanskelig å få figuren over klassediagrammet inn på en A4 side slik at det er enkelt å lese hva som står der.
Henviser til elektronisk vedlegg for et bedre bilde av klassediagrammet.

Vedlegg C

Installasjon av Arc og prototyp – Tips og råd

For å kunne kjøre mitt system må man kunne høste ressurser, siden jeg har brukt Arc som høster gir jeg her litt råd og hjelp for å få denne tjenesten oppe å gå.

For å teste mitt system kreves det at Java sdk versjon 1.4 er installert og MySQL database med tilhørende JDBC driver installert. Man bør opprette en database i MySQL som har disse egenskapene:

Database navn: arc
Brukernavn: toril
Passord: (ikke passord tomt felt)

Dersom man ikke vil bruke mitt brukernavn og passord på databasen må man gjøre om i klassen DbKobling.java slik at databasenavn, brukernavn og passord passer med valgte for database.

For å kjøre mitt system trenger man bare tabellen DC, denne ligger i XML format i elektronisk vedlegg, sammen med kildekode for systemet. I MySQL er det mulig å importere dette for å legge inn samlingen.

Dersom man ikke ønsker å bruke denne samlingen må man høste en ny, det finnes flere tilgjengelige høstere. Jeg vil forklare de største problemene jeg hadde med arc og hva som må til for at den skal fungere.

Database scriptet som følger med Arc vil ikke fungere slik de har laget det, jeg har laget et nytt med alle forbedringer det finnes i elektronisk vedlegg.

For å bruke høsteren til Arc er dette alt som må gjøres annerledes enn installasjonsguiden til arc.

Lykke til!