NTNU
Det skapende universitet

3d visualisering av konseptkart

Ole Kristian Hvaale

Master i datateknikk
Oppgaven levert: Juni 2006
Hovedveileder: Arvid Holme, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Oppgaven går ut på å visualisere, konstruere en 3d-graf, et konseptkart hvor nodene representerer konsepter og kantene forbindelser mellom disse. Konseptene er i denne samengheng læringskonsepter og kartet skal benyttes i organisering- og navigeringssamengheng.

Oppgaven gitt: 20. januar 2006
Hovedveileder: Arvid Holme, IDI

Sammendrag

I denne oppgaven er målsetningen å utvikle en metode for å organisere og visualisere en rekke med dokumenter (tradisjonelle dokumenter, læringsobjekt og kunnskapsobjekt)

I oppgaven framhever jeg at automatisk genererte konseptkart er en god måte å representere kunnskap på, jeg visualiserer derfor kunnskapen i et konseptkart. Konseptene i kartet beskrives ved hjelp av merkelapper. Relasjonene og konseptenes plassering beregnes ved hjelp av metoder fra informasjonsgjenfinning.

Dette automatisk genererte konseptkartet må visualiseres, selve visualiseringen bygger på teori fra informasjonsvisualiseringssystemer.

I beskrivelsen av hvordan kunnskap kan representeres diskuteres konsepter, forskjellige typer kunnskap, og måter å presentere kunnskap på.

Kunnskapen som kan vises som konsepter kan være tradisjonelle dokumenter eller læringsobjekter, oppgaven beskriver hva et læringsobjekt er i denne sammenheng.

I evalueringen av konseptkartet gjenomgås det prototypens egenskaper til å visualisere kunnskapen på en god og oversiktlig måte, det evalueres også hvor stor mengde med kunnskap som prototypen kan håndtere.

Jeg mener at resultatene viser at det tredimensjonale konseptkartet er et meget nyttig og pedagogisk hjelpemiddel for organisering og presentasjon av kunnskap.

Innholdsfortegnelse

1 Innledning

1.1 Organisering av kunnskap

I og utenfor skolen er det store mengder med kunnskap man tilegner seg.
Ser man på tradisjonell læring så lagres kunnskap i mentale modeller. I tillegg til de mentale modellene vil det også være en rekke dokumenter, notater og oppgaver.

Problemet man ofte har med fysisk lærestoff er at det svært sjeldent er organisert på en god måte. Siden kunnskap man ikke bruker på en stund fort blir glemt så bør kunnskapen man tilegner seg organiseres på en slik måte at det er enkelt å lagre og hente fram kunnskap.
Ved hjelp av grafiske hjelpemidler kan kunnskap presenteres på en slik måte at den blir godt organisert, oversiktlig og lettere å finne fram i.

1.2 Problemstilling

I min oppgave er målsetningen å utvikle en metode for å presentere kunnskap.

Det skal utvikles et program som framstiller en tredimensjonal graf hvor nodene representerer konsepter og kantene viser forbindelser mellom disse, med andre ord et tredimensjonalt konseptkart.

Utgangspunkter er et n-dimensjonalt vektorrom hvor vektorene representerer dokumenter (tradisjonelle dokumenter, læringsobjekter eller kunnskapsobjekter).
Ved hjelp av likhetsberegninger mellom vektorene genereres det tredimensjonale konseptkartet ved hjelp av Java 3D.

Oppgaven går ut på å visualisere disse dokumentene i et tredimensjonal graf på en oversiklig måte, det er viktig her at dokumentene viser hvordan de er forbudet innbyrdes. Ved å høyreklikke på et konsept skal det være mulig å få presentert innholdet i originaldokumentet. Ved venstreklikk på et konsept skal alle koblingene fjernes unntatt de koblingene som er knyttet til det valgte konseptet.
Det skal også legges til teknikker for å navigere og manøvrere grafen.

Det skal også være mulig for navigering i en 3D – graf hvor nodene representerer kunnskapsobjekter. Som ledd i en problemløsning kommuniserer den lærende med kunnskapsbasen og finner de kunnskapsobjektene som er aktuelle for å løse oppgave og markerer en sti som viser hvilken rekkefølge kunnskapsobjektene bør aksesseres.

1.3 Rapportens oppbygging

Oppgaven er delt i to deler, den første delen tar seg av teorien, mens del to beskriver prototypen, testing, resultater og konklusjon.

Kapittel 2, 3, 4, 5, 6 og 7 er teorikapitler som underbygger valgene jeg har tatt i utviklingen av prototypen.

Kapittel 2 og 3 tar for seg læring. Kapittel 2 beskriver hva læring er og beskriver den læringsteorien jeg støtter meg på i oppgaven. Kapittel 3 forklarer kort hva nettbasert læring er, siden det er i den arenaen prototypen mest sannsynlig vil bli brukt.

I kapittel 4 blir teorien om informasjonsgjenfinning forklart i dette kapittelet beskriver jeg hvordan man kan beregne likhetsmål mellom konseptene.

Kapittel 5 forklarer hva et læringsobjekt er. Det blir her også nevnt hvordan et slik objekt kan bli bygd opp.

Kapitel 6 gir en oversikt over en rekke informasjonsvisualiseringsteknikker, og jeg beskriver her også hva en graf er og forklarer teorien rundt plasseringsalgoritmen jeg har valgt.

Kapittel 7 beskriver forskjellig type kunnskap, og måter disse representeres på. Jeg diskuterer her også konseptkart, og forskjellige måter disse kan presenteres på.

Kapittel 8 omhandler prototypen jeg har implementert.
Kapittel 9 tar for seg testkjøring av prototypen.
Kapittel 10 konkluderer prototypen.
Kapittel 11 forklarer hva som kan jobbes videre med i oppgaven.
Kapittel 12 inneholder referanser til kildene jeg har brukt i oppgaven.

2 Læring

Læring omfatter alle forandringer i menneskets personlighet som ikke direkte eller indirekte kan føre tilbake til arvebestemte faktorer.[Kjell A]

Det finnes en rekke med teorier innenfor læring. Jeg vil beskrive to av disse. Konstruktivisme og behaviorisme.
Jeg vil også komme innom en rekke læringsstrategier og noen pedagogiske metoder for læring.

2.1 Behaviorisme

Behaviorismen er en klassisk læringsteori som lenge har vært i bruk i den norske skolen. Selv om man går mer og mer bort fra denne læringsteorien har den fortsatt stor innflytelse.
I seinere tid har man tatt i bruk nye modeller for læring, og behaviorismen har blitt vektlagt mindre. Nå fokuseres det mer på aktivt arbeid og at den lærende skal se nytten av å lære.

I behaviorismen så mener man at tanker, følelser og motiver er utilgjengelig for vitenskaplig forskning. Man vektlegger derfor i behaviorismen observerbar og målbar atferd når man skal lære.

I behaviorismen legger man vekt på egenskaper som kan måles eller veies, man mener at atferd blir styrt av ytre påvirkning. I behaviorismen benyttes en modell kalt stimulus/respons modellen.

Figur 2.1: Stimulus/respons modellen

Når den lærende blir utsatt for en viss stimulus så vil man med denne prøve å frambringe en ønsket respons fra den lærende.
Kunnskap blir dannet ved at den lærende fremringer en gitt respons i forhold til en gitt stimulus.

I behaviorismen skjer læring ved betinging og assosiasjoner, den bygger på et optimistisk læringssyn som sier at alt kan læres dersom de rette stimuli foreligger.
Læring skjer på denne måten gjennom belønning og straff. Er det den rette responsen fra den lærende etter en gitt stimulus så har den lærende den riktige kunnskapen og vil bli belønnet, har derimot den lærende feil respons så har han ikke den riktige kunnskapen og vil bli straffet.
Motivasjonen ved belønning og straff er at hvis man gir riktig respons blir man belønnet.
[HIVE]

2.2 Konstruktivisme

Hva kunnskap er og hvordan den organiseres og bygges opp er sentralt i konstruktivismen. Kunnskap blir her konstruert og man vil at ethvert individ selv konstruerer kunnskapen i sin personlige ”subjektive verden”.
I konstruktivismen er virkelighetsoppfattelsen subjektiv og hver enkelts ”verden” vil her være bygd opp av subjektive strukturer, som gjennom erfaring har vist seg brukbart for å samhandle mot omgivelsene.

Jean Piaget (sett på som grunnleggeren for konstruktivismen) mener at kunnskap aldri er en representasjon av den virkelige verden. I stedet er det en samling av konseptuelle strukturer som brukes for å bygge opp en subjektiv kunnskapsbase.

Konstruktivismen forandrer definisjonen av hva kunnskap er. I stedet for å si at kunnskap er representasjon av det som eksisterer, sier konstruktivismen at kunnskap er kartlegging av det som gjennom menneskets erfaring ser ut til å være mulig.

Jean Piaget er den utvilklingspsykologen som har hatt mest betydning for konstruktivismen. Han hadde som mål å finne fram til kunnskapens struktur.
Han mente at det skjer en indre tilpassing i våre mentale strukturer. Dette skjer ved at vi tolker ytre hendelser vi registrerer, og ved at vi hele tiden revurderer gamle oppfatninger som ikke lenger er holdbare.

Tar vi et barn som eksempel vil det være like morsomt å undersøke hvordan en rangle ser ut innvendig som det å riste på den. Barnet vil finne stadig nye bruksområdet for leketøyet, slik som for eksempel bite på den eller kaste den i gulvet. I følge Piaget så bygges det opp skjemaer fra disse handlingene.
Hvis for eksempel et barn vet at når det bakes så eksiterer det bolledeig. Og bolledeig smaker godt. Denne kunnskapen kan ses på som et konstruert skjema hos barnet. Når barnet lærer at bolledeig er godt så kan det deles opp i 3 deler.

1. Godkjenning av situasjonen (det bakes)
2. Assosiasjon til en spesiell aktivitet for situasjonen (putte i munnen og spise)
3. Forventning til et spesielt resultat (god smak)

Læringsprosessen i et konstruktivistisk syn kalles akkomodasjon og assimilasjon.

Assimilasjonsprosessen trer i kraft når man har nye situasjoner eller fenomener som tilpasses de gamle skjemaene barnet har fra før. Man kan for eksempel se på et barn som lurer på hvor det blir av sola etter solnedgang: Barnet vil kanskje tro at sola har lagt seg til å sove og at den vil stå opp neste morgen. Dette er barnets tolkning av solas gang over himmelen. Barnet tolker ved hjelp av et kjent skjema som barnet kjenner fra før nemlig det å legge seg og stå opp. Barnet tilpasser sine observasjoner til noe som er kjent fra før.

Ser man på akkomodasjonsprosessen, kan vi ta for oss eksemplet om barnet som trodde sola gikk å la seg. Hvis barnet ikke lenger er fornøyde med denne forklaringen så er det gamle skjema ikke tilstrekkelig. For at det skal skje læring, må de gamle skjemaene reorganiseres og utvides.

Ser man på eksemplet om sola igjen så kan en femåring kanskje oppfatte at det er jorda som går rundt og at det blir natt fordi jorda snur ryggen til sola. Akkomodasjonen vil her si å justere og forandre de kognitive strukturene slik at de kan ta inn nye sider ved omgivelsene.

Piaget hevder at kognitiv forandring og læring skjer når en hendelse avviker fra det en forventer på bakgrunn av eksisterende skjema. Dette avviket fører til en akkomodasjon som gjør at den lærende skaper ny og forbedret forståelse. [HIVE]

Siden man ser på kunnskap som en subjektiv konstruksjon hos hver enkelt fører dette til at læring handler om å bygge disse konstruksjonene selv. All læring bygger på tidligere kunnskap, som igjen er utgangspunktet for hvordan man kan tilegne seg ny kunnskap.

For at den lærende skal kunne lære så må den lærende sette ny kunnskap i forbindelse med gammel kunnskap, det må foregå en assimilasjon og akkomodasjon. Det er derfor viktig at et læringsopplegg legger til rette for aktiviteter der den lærende kan reflektere over ny kunnskap ut fra eksisterende kunnskap. Man kan si at læring må skje aktivt med den lærende i sentrum.

I konstruktivismen skiller man mellom to typer læring, opplæring og læring.
Opplæring går ut på å få noen til å handle i spesielle mønster som å sparke en ball eller utføring av en regneoppgave. Dette skaper mentale handlingsmodeller som hjelper den lærende til å oppnå mål i interaksjonen med omgivelsene.
Læring går mer ut på det å oppnå en forståelse, dette skaper måter å handle og tenke på som virker brukbare.
Andre læringsteorier har ikke slike tydelige skiller mellom disse læringstypene. Dette kan, som i behaviorismen, føre til en for kraftig fokus på opplæring som gjør det lettere for den lærende å kopiere observerbar handling, men denne typen læring vil i følge konstruktivismen ikke føre til forståelse og kunnskap.

2.3 Læringsstrategier

Jeg vil i dette kapitlet forklare tre forskjellige læringsstrategier. Studenter tilnærmer seg læring på ulike måter som kan karakteriseres som en dypinnrettet tilnærming, overfladisk tilnærming og strategisk tilnærming til læring. I artikkelen til Torstein Rekkedal beskrives disse tre slik [Nettskolen]:

Dypinnrettet tilnærming til læring:

Intensjonen her er å få studenten til å forstå ideene selv. Studenten skal skape sin egen kunnskap gjennom å:

- Relatere ideer til tidligere kunnskap og erfaring
- Søke etter mønster og underliggende prinsipper
- Kontrollere forklaringer og bakgrunnsmateriale og trekke egne konklusjoner
- Vurdere forklaringer og argumenter grundig og kritisk

Målet er her at studenten skal bli aktivt interessert i kursinnholdet.

Overfladisk tilnærming til læring:

Intensjonen her er å tilfredsstille kurs og opplæringskrav:

Det skal reproduseres kunnskap gjennom å:

- Studere uten å reflektere over hensikt eller strategi
- Behandle kursinnholdet som urelaterte kunnskapsbiter
- Memorere fakta og fremgangsmåter rutinemessig

Problemet med denne tilnærmingen er at studenten finner det vanskelig å finne mening i nye ideer som presenteres.
Studenten får også lett følelse av stort arbeidspress og studie-"angst".

Strategisk tilnærming til læring:

Intensjonen er her å oppnå best mulig karakter.

Læring organiseres gjennom å:

- Legge stor innsats i studiearbeidet
- Finne frem til effektiv studiebetingelser og materiale
- Styre tid og innsats effektivt

Studenten er her særlig oppmerksom på vurderingskrav og kriterier og studenten tilpasser arbeidet til lærerens prioriteringer og preferanser.

En student som tilnærmer seg en dypinnrettet tilnærming ser etter meningen og det organiserende prinsipp i fremstillingen mens en student med en overfladisk tilnærming ser på innholdet som en samling fakta som skal læres og huskes.
Undervisning og evalueringsopplegg påvirker studentens læringsstrategi.

2.4 Pedagogiske metoder

Det finnes en rekke med ulike metoder innenfor konstruktivistisk tankegang. Jeg vil her ta for meg de to pedagogiske metodene problembasert læring og kollaborativ læring.

2.4.1 Problembasert læring

Problembaser læring har sin opprinnelse fra medisinsk utdanning. Oppgaver som representeres i problembasert læring er vanligvis relatert til reelle situasjoner, eller bygger på aktuelle hendelser. Noe av poenget med problembasert læring er å la studenten i større omfang få ta del i sin egen utvikling av kunnskap. Problemer kommer sjeldent ferdig definerte i det daglig liv. Det å kunne ta tak i et handlingsforløp eller en tilstandsbeskrivelse for å finne ut hva problemet består i er en vesentlig del i det å finne en løsning som kan fungere.
Man fokuserer her på problemstillingen istedenfor å fokusere på for eksempel en fastsatt tekst.

Hovedpoenget med denne typen læring er å lære opp den lærende til å resonere seg fram til løsninger ved å se sammenhenger mellom informasjonen som blir gitt.

Problembasert læring har vist seg å fungere best når man jobber i grupper. Siden man da får mulighet til å løse problemer på et nivå som ikke er mulig når man jobber alene. [Immersion]

2.4.2 Kollaborativ læring

Hovedformålet med denne typen læring er å utvikle og forbedre kritisk tenkning hos den lærende. Dette gjøres ved å rekonstruere den lærendes kunnskap og ideer gjennom dialoger, kommentarer, diskusjon og deling av informasjon. I en slik læringssammenheng vil for eksempel utveksling av ideer i små grupper øke interessen til deltakeren og øke kritisk tenking i gruppa. [Immersion]

3 Nettbasert læring

Det finnes en rekke med forskjellige definisjoner på hva nettbasert læring er. Siden nettbasert læring er et område under stadig utvikling og fornying, så spriker også ideene og forståelsen om nettbasert læring. Jeg vil prøve å beskrive hva nettbasert læring er i forhold til mangfoldet av teorier.

3.1 Hva er nettbasert læring?

Nettbasert læring er enda et relativt ungt fagområde. Det finnes en rekke oppfatninger av hva nettbasert læring er og ikke alle disse oppfatningene er like gode. Jeg vil i dette kapitlet komme innom en rekke uheldige oppfatninger.

Kostnadsbegrensinger

Noen mener nettbasert læring er forbundet med kostnadsbegrensninger. Man mener her at det å legge ut et kurs på nett kun har en oppstartskostnad. Når kursene er på nett tror man at alt går av seg selv. De som jobber seriøst med nettbasert læring, pedagogisk og ved applikasjonsutvikling, vil mene at dette er en grov misforståelse.

Legg ut på nett

En annen misforståelse er at man tar et ikke-nettbasert opplegg og legger det ut på nett. For eksempel så kan man legge ut kapitler av en lærebok ut på nett. Dette gir ingen fordeler og er like pedagogisk og behagelig som å lese boka i papirformat. De fleste vil mene at det er mer behaglig å lese boka i papirformat en å sitte foran en dataskjerm.

Teknologi

Et problem innenfor teknologiutviklingene er at det er en oppfatning om at ny teknologi alltid er bedre en gammel teknologi uansett.

Hva er så nettbasert læring?

Noe forenklet kan man si at nettbasert læring er utdanning som helt eller delvis blir tilrettelagt gjennomført via internett.
Nettbasert læring er en form for fjernundervisning der Internet brukes til å formidle undervisning som deltakere kan følge uavhengig av tid og sted. Mange slike fjernundervisnings opplegg benytter seg av en e-læringsplattform. En slik e-læringsplattform kan her sammenlignes med et klasserom på nettet der fagstoff presenteres, lærer eller veileder underviser, gruppearbeid kan gjennomføres. Et slik system har som oftest også kommunikasjonsverktøy som gjør det mulig for lærer og kursdeltaker å kommunisere.

Noen stiller seg skeptisk til nettbasert læring siden, man i denne formen for læring mangler det tradisjonelle klasserommet. Dette kan også sees på som positivt siden det kreves nå at man tenker nytt. Man ser nå bort fra den tradisjonelle oppfatningen om at læring er å motta passiv informasjon. Det er ikke mulig å motta passiv informasjon på samme måte i et nettbasert

læringsopplegg. Ser man dette i forhold til et konstruktivistisk syn er dette positivt. Siden man i konstruktivismen mener at kunnskap ikke mottas, men må skapes.

Den lærende får i nettbasert læring en mer aktiv rolle. Den lærende må selv søke etter og bearbeide kunnskap. Dette kan gjøres alene eller i samarbeid med andre.

Nettbasert læring gir også større fleksibilitet til sted, tid og hastighet som studenten vil lære. Den lærende er her i sentrum og han kan bestemme både tempo og arbeidstid.

Hovedpoenget med nettbasert læring er at vi objektivt må se på de mulighetene som Internet har gitt oss i læringssammenheng. I denne sammenheng er målet ikke å spare utgifter, eller komme på nett, men det å kunne utnytte mulighetene som er tilgjengelig til å lage nye og bedre pedagogiske læringsopplegg. [Tore Vestues]

3.2 Datateknologi

Den mest utbredte teknologien for bruk i fjernundervisning og fleksibel læring de siste årene er internett. Det suppleres fortsatt med andre teknologiske løsninger slik som videokonferansesystem, cd-rom baserte løsninger og lignende.

For noen år siden var teknologibruken i norsk fjernundervisning mye mer variert. Man hadde tekniske løsninger som for eksempel videokonferanser, lyd/telekonferanser, cd-rom, videokassetter, lydkassetter og tv utsendelser. Internett har nå blitt den dominerende teknologien. Grunnen til dette er at teknologien er blitt mer utbredt og tilgjengelig
Tilgangen på internett finnes nå nesten over alt. Enhver student har tilgang til nett på skolen, de fleste har via jobben tigang til internett og de fleste hjem har i dag en form for internett-tilkobling.
Siden internett er så utbredt fører det til at det blir billig å ta i bruk denne teknologien i undervisning.
En annen grunn er at teknologien er blitt mer moden, man har nå fått fjernet de fleste barnesykdommene. Selve brukerkompetansen ved å benytte internett har også økt. Internett er i ferd med å bli alminneliggjort i likhet med for eksempel telefon, radio og TV. Alle er i stand til å bruke internett.

Selve internettteknologien egner seg til fjernundervisning og fleksibel læring. Siden den gir stor mulighet for kommunikasjon, samhandling og fleksibilitet i forhold til når man vil lære. Den egner seg også meget godt til støtte for utarbeiding/formidling av tekster og samarbeid om tekster. [Gunnar Myklebost]

4 Informasjonsgjenfinning

Med dagens bruk av datasystemer øker mengden med informasjon som lagres på digital form. Med denne økningen av informasjon blir det vanskeligere å finne fram til den informasjonen man virkelig er på jakt etter. I tillegg blir det lagt ut nye dokumenter på web hver eneste dag. Disse dokumentene kan inneholde alt fra riktig faktainformasjon til informasjon som er uriktig eller falsk informasjon. Arbeid innenfor fagfeltet informasjonsgjenfinning går på å lette prosessen med å finne fram til relevant informasjon for brukerne av et informasjonsgjenfinningssystem. [INFO221]

4.1 Klassiske informasjonssystemer

Informasjonsgjenfinningssystemer brukes til å presentere, lagre og organisere informasjonsenheter. Et slikt system gir aksess til informasjonsenheter ut fra brukerens informasjonsbehov.
Brukerens informasjonsbehov blir formulert ved en spørring som prosesseres av et IR system som raskest mulig henter fram dokumentene eller informasjonsobjektene som best svarer til spørringen ved å rangere dem i forhold til likhet. [WIKIPEDIA][INFO221]

Et informasjonssystem består som oftest av en dokumentsamling.
Hvert dokument i samlingen er beskrevet ved et sett av representative nøkkelord kalt indekstermer. Indekstermene brukes for å indeksere dokumentene, dette gjøres med en semantikk som hjelper oss å huske dokumentenes hovedtemaer.

IR systemet må ved hjelp av statistikk fra informasjonsobjektene(dokumenter) i en samling rangere dem etter hvor relevant de er i forhold til brukerens forespørsel, et godt IR system må finne og returnere alle dokumenter som er relevante i forhold til brukerens forespørsel og returnere færrest mulig ikke-relevante dokumenter. [INFO221][Holme]

Informasjonsgjenfinning var muligens kun relevant for bibliotekarer og informasjonseksperter fram til rundt 1990. Men på 1990 tallet kom World Wide Web som gav en universell kilde for å dele kunnskap og kultur.

Denne nye muligheten for å spre kunnskap og deling av informasjon tiltrakk seg en stor mengde med uerfarne brukere. Informasjonen var vanskelig å finne fordi det manglet en veldefinert underliggende datamodell. Dette gjorde at brukerterskelen ble for stor for de nye brukerne. [INFO221]

4.2 Modeller

World Wide Web sitt fremskritt i deling av informasjon ga en fornyet interesse for IR og flere modeller ble tatt i bruk.
Noen av de mest brukte modellene innenfor informasjonsgjenfinning er boolsk modell og vektormodellen. [INFO221]

4.2.1 Boolsk modell

I en boolsk modell søker man etter dokumenter ved å oppgi søketermer som identifiserer dokumentene. Selve søket bygges opp ved boolske operatorer som AND, OR og NOT. Ulempen med denne modellen er at man bruker et sett med binære utvelgingskriterier. Enten så matcher søkekriteriene eller så matcher de ikke. Det blir med denne modellen derfor ingen virkelig rangering.
Det er i tillegg vanskelig for uerfarne brukere å anvende en slik modell siden de må formulere spørring ved hjelp av boolske utrykk. Dette er vanskelig siden det ikke ligner mye på naturlig språk. Boolsk modell returnerer også ofte for få eller for mange dokumenter etter et gitt søk. [INFO221][Holme]

4.2.2 Vektor modell

Dokumenter blir her representert som vektorer i et *n*-dimensjonalt vektorrom. Her er *n* antall unike indekstermer som finnes i dokumentsamlingen. Selve dokumentvektoren består av et sett med *n* termvekter. En termvekt angir hvor viktig termen er for dokumentet.
Spørringen som blir utført ut fra brukerens informasjonsbehov blir også framstilt som en vektor på samme måte som dokumentene.
Sammenligning av denne spørringsvektoren og dokumentvektorene gjør at man kan finne grader av likhet mellom spørringen og dokumentene i dokumentsamlingen. Dette gjør at man kan rangere dokumentene og returnere en liste over de mest relevante eller for eksempel framstille dem visuelt i forhold til innbyrdes likhet.

Man får i motsetning til boolsk modell også returnert dokumenter som bare delvis passer med spørringen. Hvilke dokumenter som skal returneres kan bestemmes av hvor lik de skal være med spørringen. Hvordan likheten kan beregnes beskrives seinere i dette kapitlet.

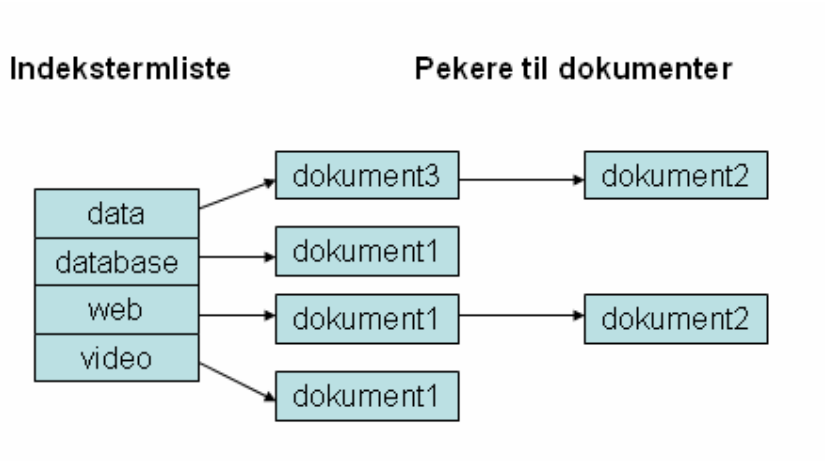
Siden man behandler spørringene og dokumentene som vektorer så kan man som sagt beregne likheter mellom de forskjellige dokumentene og spørringene. Dette gjør at vektormodellen passer godt til min oppgave der jeg skal framstille dokumentene i 3d.
Beregning av likhet mellom hvert dokument i forhold til de andre dokumentene gjør at det er mulig å plassere dokumentene i en 3d graf ved hjelp av en plasseringsalgoritme.

4.3 Indeksering

En indeks er et sett med lovlige termer som brukes for å beskrive dokumenter og spørringer.[Rijsbergen]
Termene er grunnenheten i en indeks og er vanligvis enkelt ord på lovlig form. Lovlig form vil si at termene er analysert og filtrert i samsvar med det som er bestemt for det gjeldende systemet. Termene som brukes til indeksering kalles indekstermer.

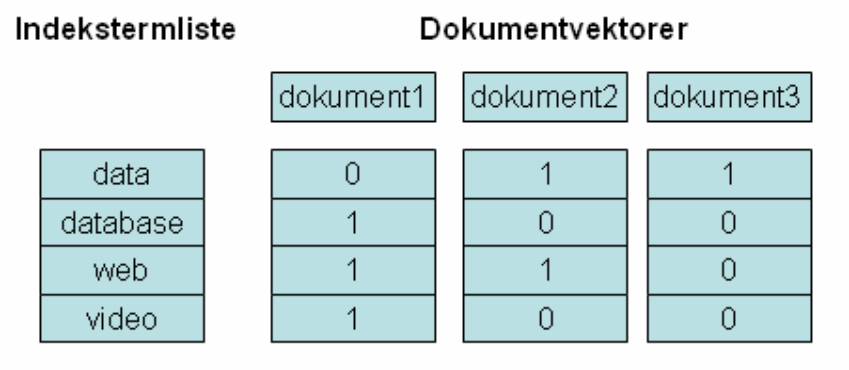
I et IR-system vil man danne en slik indeks av alle termer i dokumentsamlingen som er på lovlig form.
Dokumentene representeres i forhold til indeksen som er laget for dokumentsamlingen. Dette kan gjøres på flere forskjellige måter:

I en boolsk modell brukes det en invertert modell. For å lagre hvilke termer som forekommer i dokumentene er det i indekstermlisten pekere til de dokumentene som inneholder termen. Dette illustreres av figur 4.1



Figur 4.1: Boolsk modell

Når det gjelder vektormodellen så representeres dokumentene som vektorer. Disse vektorene inneholder termvekter. Termvektene gir informasjon om termen er representert i dokumentet eller ikke. Figur 4.2 illustrerer dette.



Figur 4.2: Vektormodell

Indekseringen som skal benyttes i denne oppgave blir gjort automatisk. Denne automatiske indekseringen består vanligvis av leksikalsk analyse, fjerning av stoppord og stemming.

4.4 Leksikalsk analyse

Leksikalsk analyse er det første steget i en automatisk indekseringsprosess.
Den leksikalske analysen har som hovedoppgave å konvertere en strøm av tegn til en strøm av termer.
Analysen fjerner tegn som ikke ønskes. Komma, punktum og tall egner seg dårlig i termer siden de ikke gir noe informasjon om dokumentet.
En slik analyse er kostbar siden alle tegn i en tekst må undersøkes. Dette utgjør opp til 50 % av indekseringskostnadene. [Holme]

4.5 Stoppordlise

Ikke alle termer egner seg som indekstermer. En term som ikke skal benyttes som indeksterm kalles et stoppord. Disse stoppordene danner grunnlag for en stoppordliste som sammenlignes mot termene som blir dannet av den leksikalske analysen. Etter sammenligning så vil man få fjernet de termene som ikke egner seg som indekstermer.
Det er flere fordeler ved å redusere antall termer, man får fjernet termer som ikke skiller dokumenter og man reduserer termlisten som gjør at effektiviteten til IR-systemet øker.

Danning av selve stoppordlisten kan gjøres manuelt, automatisk eller i en kombinasjon av manuelle og automatisk.

Mesteparten av den automatiske indekseringsmetodene starter med å observere ordfrekvensen i dokumenttekstene.
Ord som forekommer i få dokumenter er mer verdifulle enn ord som forekommer i de fleste eller alle dokumentene [Salton og McGill 1983].

Hvilke termer som plukkes ut som stoppord baserer seg ofte på Luhns ideer. Disse idene går ut på å velge termer som skal representere dokumenter i forhold til termfrekvenser [Rijsbergen].

Luhn kom fram til at termer som representeres veldig mange ganger er vanlige, og termer som er representert få ganger er sjeldne. Disse termene som kommer i kategoriene sjeldne og vanlige er irrelevante og bidrar ikke med å skille dokumentene og kan derfor fjernes.

Luhn sine teorier benytter seg av Zipfs lov.

Zipfs lov går ut på at produktet av termens frekvens og termens rang er tilnærmet konstant. Termens frekvens bestemmes av antall ganger det bestemte ordet forekommer i en dokumentsamling, mens rangen blir rangert ut fra frekvensen. Luhn brukte dette til å spesifisere grenseverdier. Et ord som forekommer 10000 ganger i en dokumentsamling kan få rank 1 som hyppigste ord, mens et ord som forekommer bare en gang kan få rangen 1000.
Har man for eksempel 4 dokumenter med følgende termer så vil frekvensen og rangeringen bli følgende:

Dok1: trening, vekter, orientering, maraton.
Dok2: trening, orientering, maraton.
Dok3: maraton, trening.
Dok4: maraton.

Term	Frekvens	Rang
maraton	4	1
trening	3	2
orientering	2	3
vekter	1	4

Tabell 4.1: Frekvens og rang for termer

Hovedpoenget med Zipfs bruk i gjenfinning er at de mest brukte termene og de minst brukte, vil være urelevante og kan fjernes som termer til søk.

Luhn skapte derfor to terskelverdier en for høyfrekvente og en for lavfrekvente termer. Termene som ligger mellom disse terskelverdiene vil egne seg som signifikante termer. En signifikant term vil være en term som brukes for å skille dokumenter fra hverandre.

Matematisk så er Zipf`s lov:

$$Termfrekvens * rang \simeq Konstant$$

Figur 4.3: Zipf`s lov

Frekvensen av ett ord multiplisert med rangen til ordet er lik frekvensen av et annet ord multiplisert med sin rang.

$$F_i * r_i = F_j * r_j = C$$

Figur 4.4: Zzipf`s egenskap.

$$F_j = Frekvensen\ til\ termj$$
$$r_j = rangen\ til\ termj$$
$$C = Konstant$$

Figur 4.5: Beskrivelse av verdier til formel i figur 4.4

4.5.1 Høyfrekvente termer

Høyfrekvente termer er termer som forekommer så ofte i dokumentsamlingen at de er ubrukelige som indekstermer. I følge Baeza-Yates så er 80 % av termene som forekommer i en dokumentsamling ubrukelige som indekstermer.
Dette fordi de forekommer så ofte at termene ikke hjelper til med å skille dokumenter fra hverandre. Det er derfor viktig å fjerne disse termene fra dokumenter.

I en dokumentsamling vil det være en rekke med funksjonsord. Funksjonsord er artikler, preposisjoner og bindeord. Disse ordene vil være til liten hjelp når man skal prøve å skille

dokumentene fra hverandre i en dokumentsamling. Funksjonsordene bidrar heller ikke i stor grad til å beskrive dokumentene fordi funksjonsordene ikke er beskrivende i seg selv. Fjerning av funksjonsordene i en dokumentsamling vil ikke ha noen særlig negativ effekt i beskrivelsen av dokumentene, men fjerningen av disse funksjonsordene vil lette prosessen det tar å skille dokumentene fra hverandre.

Det vil også være en positiv innvirkning på differensiering av dokumenter ved å fjerne andre termer som blir representert ofte i en dokumentsamling. En ulempe med å fjerne disse er at de kan ha en stor betydning i beskrivelsen av dokumentene.

4.5.2 Lavfrekvente termer

Fjerning av lavfrekvente termer er en vanskeligere prosess enn fjerning av høyfrekvente. Dette fordi termene må finnes og fjernes ved undersøkelser av dokumentet. Man kan ikke her lage en liste på forhånd slik som man kan med høyfrekvente termer.
Dette fører til at stoppordlisten må genereres automatisk.
Det er ikke alle lavfrekvente termer som er ubrukelige i en indekseringssammenheng. En term som er representert sjeldent i en dokumentsamling kan være representert ofte i ett eller noen få dokumenter. Fjerning av disse lavfrekvente termene vil redusere systemets evne til å skille dokumenter og redusere evnene til å representere de gitte dokumentene.

4.5.3 Manuelt genererte stoppordlister

Formålet med en manuelt generert stoppordliste er å luke ut ”vanlige” termer. Denne listen blir som oftest laget før man vet hvilken dokumentsamling man skal behandle.
Som nevnt før så er funksjonsord ofte brukt i dokumenter. En manuell stoppordliste er derfor en liste over funksjonsord som kan fjernes.
Disse manuelle stoppordlistene er generelle og kan brukes av de fleste IR-systemer så lenge de behandler samme språk.
Fjerning av funksjonsordene fjerner som oftest ikke betydningen av dokumentet og det vil bli lettere å skille dokumentene fra hverandre når funksjonsordene er fjernet.

Man har derimot noen ulemper ved å bruke manuelle stoppordlister. Siden man i noen dokumentsamlinger kan ha funksjonsord med stor betydning. Som for eksempel Shakespare`s kjente ”to be or not to be” vil bli fjernet som funksjonsord.
Man får ikke med denne metoden fjernet termer som forekommer veldig ofte og veldig sjeldent, dette gjør at reduksjonen av indekstermlisten ikke blir så stor som for automatiske genererte lister.

4.5.4 Automatisk genererte stoppordlister

Danning av en automatisk generert stoppordliste går ut på at man kan fjerne alle vanlige og sjeldne termer. Man får på denne måten redusert indekstermlisten betraktelig, siden man ikke bare fjerner funksjonsord men man fjerner også sjeldne og vanlige representerte termer.
Et problem i denne prosessen er å bestemme hvor man skal sette øvre og nedre terskelverdi.

Rijsbergen sier det slik:

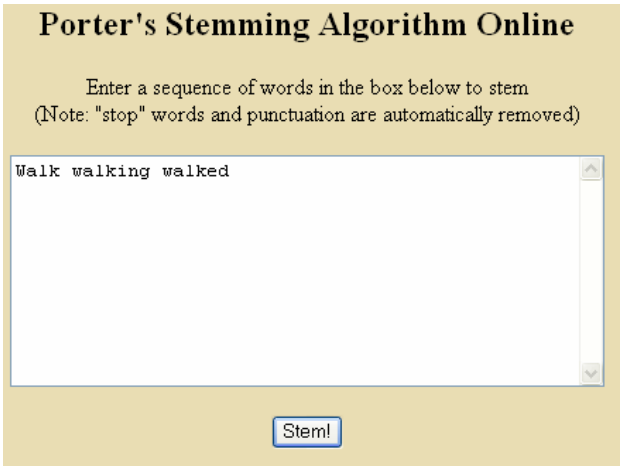
”A certain arbitrariness is involved in determining the cut-offs. There is no oracle which gives their values. They have to be established by trial and error.”[Rijsbergen]

Problemet med automatisk genererte stoppordlister er at man enten får fjernet for mange eller for få termer fra dokumentsamlingen. Justeringen av øvre og nedre grenseverdi må derfor justeres under IR systemets kjøring. Avanserte brukere kan sikkert fininnstille disse verdiene til en gitt dokumentsamling. Men en vanlig bruker vil ha større problemer med å sette disse verdiene under kjøring. Dette gjør at man kan få en for høy brukerterskel på systemer med automatisk genererte stoppordlister.

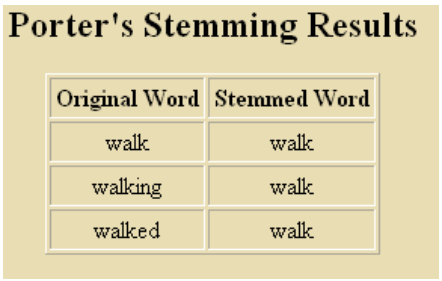
4.6 Stemming

Stemming er en metode som benyttes for å fjerne postfiks fra ord i et dokument eller spørring. Stemmeren finner og slår sammen termer med samme ordstamme til en indeksterm.
For eksempel så vil de engelske ordene walk, walked og walking bli slått sammen til termen walk ved å fjerne endingene.
For å kunne slå sammen termer så må de ha samme ordstamme.
Denne regelen er som oftest korrekt, men vil i følge Rijsbergen i enkelte tilfeller føre til en overforenkling, siden noen termer med samme ordstamme ikke nødvendigvis har samme betydning.

Et eksempel på en slik stemmer er porterstemmeren. Dette er en velkjent algoritme som blir brukt i mange applikasjoner. Kode og informasjon om denne algoritmen finnes på nett i de fleste programmeringsspråkene. Under er det skjermbilder fra en online porterstemmer før og etter stemmingen.



Figur 4.6: Skjermbilde fra online porterstemmer før stemming hentet fra http://maya.cs.depaul.edu/~classes/ds575/porter.html



Figur 4.7: Skjermbilde fra online porterstemmer med resultat etter stemming

4.7 Vekting

Ved bruk av vektormodellen så blir indekstermer vektet for hvert dokument.
Termvektene blir plassert i dokumentvektorer. Vektingen kan være binær eller basert på antall ganger termene finnes i dokumentene.

Ved bruk av binærvekting gis termer som forekommer i et dokument verdien 1, hvis de ikke er i dokumentet får de verdien 0. Denne metoden er effektiv siden den ikke krever noen beregninger.

Docs	t1	t2	t3
D1	1	0	1
D2	1	0	0
D3	0	1	1
D4	1	0	0
D5	1	1	1
D6	1	1	0
D7	0	1	0
D8	0	0	1
D9	0	1	1

Tabell 4.2: Binærvekting

Man kan også bruke en metode kalt ”Raw Term Weights”
I denne metoden er det frekvensen til termene i hvert dokument som angir vekten.

Docs	t1	t2	t3
D1	2	0	3
D2	1	0	0
D3	0	4	7
D4	3	0	0
D5	1	6	3
D6	3	5	0
D7	0	8	0
D8	0	10	0
D9	0	3	5

Tabell 4.3: Raw Term Weights

En vanlig metode i følge [SIMS 202] er å bruke en kombinasjon av termfrekvensen og invertert dokumentfrekvens.

Målet med denne metoden er å gi en vekt T_f * IDF til hver term i hvert dokument.
T_f er her termfrekvensen og IDF er den inverse dokument frekvensen.

Denne vektingen tar hensyn til at termer som er representert i mange dokumenter ikke har særlig gode egenskaper i å skille dokumenter [Baeza-yates].

For å beregne disse vektene bruker man følgene formler og verdier:

$$W_{ik} = tf_{ik} * \log\left(\frac{N}{n_k}\right)$$

Figur 4.8: Formel for beregning av vekt til term.

$$T_k = term\, k\, i\, dokument\, D_i$$
$$tf_{ik} = frekvens\, av\, term T_k\, i C$$
$$N = totalt\, antall\, dokummeter\, i\, samling C$$
$$n_k = antall\, dokummeter\, i C\, som\, inneholder T_k$$
$$idf_k = \log\left(\frac{N}{n_k}\right)$$

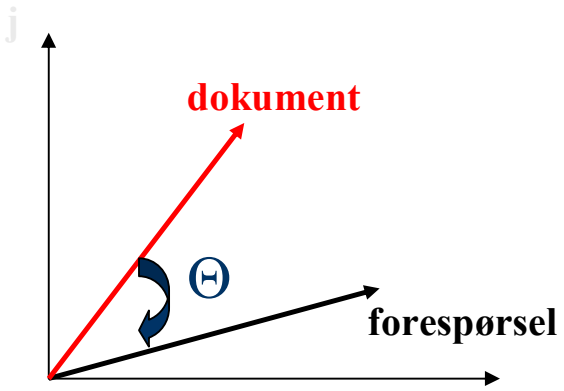
Figur 4.9: Forklaring verdier i formel i figur4.8.

4.8 Søking

En viktig del av et informasjonssystem er mulighetene til å søke etter informasjon. Søkingen foregår ved at brukeren utfører en spørring mot systemet. Dette kan gjøres på forskjellige måter.
Ved boolsk modell representeres spørringen som et boolsk utrykk. Spørringen blir bygd opp med en eller flere termer som man søker etter, ved hjelp av boolske operatorer som and, or og not. Resultatet fra spørringen er et sett med dokumenter som tilfredsstiller det boolske utrykket i spørringen. Resultatene fra spørringer ved boolsk modell kan ikke rangeres, enten så matcher de spørringen eller så matcher de ikke.

Bruker man vektormodellen så blir spørringen gjort om til en vektor av termer i samme vektorrom som dokumentene. En beregner likheter mellom spørringsvektorer og dokumentvektorer. Etter bergningene kan man få returnert en liste som er rangert etter likhet i forhold til spørringen.

Beregning av likhetsverdier mellom to vektorer i vektormodellen kan gjøres ved å finne cosinus av vinkelen mellom de to vektorene(se figur 4.10), et eksempel på en formel for å gjøre dette vises i figur 4.11.



Figur 4.10: Cosinus beregnes mellom de to vektorene

$$simularitet(d_i, d_j) = \frac{\vec{d}_i * \vec{d}_j}{(|\vec{d}_i| * |\vec{d}_j|)} = \frac{\sum_{k=1}^{t} W_{ki} * W_{kj}}{(\sqrt{\sum_{k=1}^{t} W_{ki}^2} * \sqrt{\sum_{j=1}^{t} W_{kj}^2})}$$

Figur 4.11: Formel for å beregne likhet mellom to dokumenter ved å finne cosinus mellom vektorene. Vektorene er her beskrevet som di og dj.

5 Læringsobjekter

Jeg vil i dette kapitelet prøve å definere hva et læringsobjekt er. Læringsobjekter benyttes i hovedsak i nettbasert læring, jeg vil derfor først beskrive de aktørene som inngår i nettbasert læring og beskrive de behovene de har. Jeg vil deretter beskrive hva et læringsobjekt er ved hjelp av en togmetafor.

5.1 Aktører

Det er tre aktører som er viktige i nettbasert læring, den lærende, læreren og produsenten:

Den lærende må på en eller annen få tak i det lærestoffet som er nødvendig for å oppnå en grad i høyere utdanning eller for å kunne fordype seg innen spesifikke områder for å forbedre sin kompetanse.
Læreren er ansvarlig for hva lærestoffet skal omfatte og må passe på at det er i samsvar med fagbeskrivelser. Læreren har også ansvar for å kvalitetssikre lærestoff som produseres og gi informasjon til produsenten om hva som skal produseres.
Produsenten har som oppgave å lage de interaktive læringsressursene.

En person kan ta rolle som mer enn en av disse "aktørene". En lærer kan for eksempel også ta rollen som produsent, han produserer da sitt eget lærestoff.
Dette er som oftest materiale laget på tekstform siden det krever mye tid og kompetanse for å produsere videoer og interaktive animasjoner. Det vil være misbruk av tid og ressurser hvis læreren skulle produsere avanserte læringsressurser. Derfor bør denne jobben gjøres av personer som har trening og kompetanse på å lage interaktive læringsressurser.
Disse 3 aktørene har også en rekke behov.

5.1.1 Aktørenes behov

Den lærendes behov:

- Tilgang til oppdatert lærestoff: Endringer i faginnholdet som har skjedd i seinere tid må bli oppdatert.
- Lære det man har behov for: Slippe å lete etter det man skal ha tak. De må være enkelt å finne den kunnskapen man ønsker. Man bør ikke trenge å måte bla gjennom en lærebok eller dokument for å lære om et spesifikt tema.
- Lære uavhengig av tid å sted: Selv ha mulighet til å velge når og hvor man skal lære.
- Selv finne det lærestoffet som er nødvendig for å nå læremålet eller løse problemstillingen man har fått.
- Lage sin egen læringssti ut fra egen kompetanse.
- Forvalte lærestoffet: Mulighet for å lagre lærestoffet i en personlig kunnskapsbase for seinere bruk.
- Utveksle lærestoff med andre: Gi mulighet til å dele eget lærestoff med andre.

Lærerens behov:

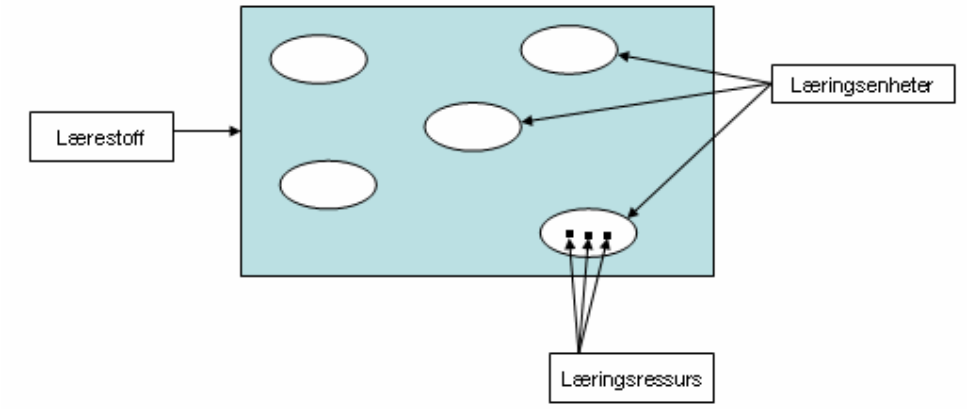
- Oppdatere og revidere lærestoffet: Mulighet for å utføre endringer hurtig og enkelt.
- Gjenfinne lærestoff: Mulighet å finne fram til eksisterende lærestoff som skal brukes i et kurs på en enkel måte.
- Gjenfinne læringsressurser: Må være enkelt å finne læringsressurser som brukes som byggesteiner til å lage lærestoff.
- Gjenbruke lærestoffet: Lærestoffet må være laget på en slik måte at det kan brukes gjentatte ganger og i ulike kontekster.
- Gjenbruke læringsressurser: Disse er kostbare å utvikle og bør derfor ha lang levetid.
- Kombinere læringsressurser til lærestoff.
- Konsistent i utvikling av lærestoff: bruk av maler.
- Individtilpasse lærestoffet: Tilpasse lærestoffet til den lærendes kompetanse.
- Tilby lærestoff i forskjellige former: Lærestoffet kan bestå av et eller flere element av tekst, bilde, video, animasjon, lyd og lignende.

Produsentens behov:

- Motta riktig og relevant informasjon om hva som skal produseres.
- Produsere læringsressurser i formater som er tilgjengelig i alle nettlesere.
- Produsere læringsressurser i et format som er raske å laste ned.
- Gjenfinne læringsressurser.
- Gjenbruke læringsressurser.
- Kostnads og tidseffektiv produksjon.
- Forbedre og revidere læringsresurser.

5.2 Oppbygging av lærestoff

Læringsressurser er den minste byggesteinen i et lærestoff. En læringsressurs kan være en tekstfil, bildefil, lydfil, animasjon, eller lignende. En kombinasjon av flere læringsressurser vil utgjøre et lærestoff.



Figur 5.1: Hvordan lærestoffet er bygd opp.

Et lærestoff kan være sammensatt av mange avanserte læringsresurser. Det å utvikle slike læringsressurser slik som video, animasjon, og spill er kostnadskrevende. Det er derfor viktig at man lager læringsressursene slik at de kan gjenbrukes og har lang levetid.
Det er også viktig at aktørene lett kan finne igjen de ulike læringsressursene og lærestoff. For å gjøre lærestoff lett å gjenbruke og revidere bør hvert enkelt lærestoff ikke være for omfattende. Lærestoffet bør være brutt ned i mindre læringsenheter. Disse læringsressursene og læringsenhetene bør kunne brukes i flere forskjellige kontekster. Man får da en 3 deling av lærestoffet der lærestoffet består av læringsenheter og læringsenhetene består av læringsressurser. Figur 5.1 illustrerer dette.

5.3 Kunnskapsobjekt og læringsobjekt

I min oppgave vil et kunnskapsobjekt være definert på følgende vis:
Et kurs utviklet for internett vil være bygd opp av læringsressurser som settes sammen til et kunnskapsobjekt. Kunnskapsobjekter kombineres til læringsobjekter. Flere læringsobjekter vil utgjøre en hel kursmodul.

I min oppgave skal jeg visualiseres en rekke kunnskapsobjekter(læringsobjekter). I dette kapitlet vil jeg prøve å forklare hva et læringsobjekt er.

Tidlig på 1990 tallet oppstod ideen om læringsobjekter. IEEE har bidratt mye til å utvikle læringsobjektdefinisjoner og metadatadefinisjoner for læringsobjekter. Den definisjonen har vært basis for alle andre definisjoner av læringsobjekter.
I feltet for å utvikle læringsobjekter er det i hovedsak to grupperinger. Man har lærere og folk med undervisningsbakgrunn i den ene gruppen og folk med programmeringsbakgrunn i den andre grupperingen. Forskjellen på disse er ofte at de som kommer fra en undervisnings-bakgrunn diskuterer objekter fra et pedagogiskperspektiv mens de med en programmerings-bakgrunn diskuterer læringsobjekter fra et implementeringsperspektiv.

IEEE definerer læringsobjekt slik:

> *"any entity, digital or non-digital, which can be used, re-used or referenced during technology-supported learning"*

Fra denne definisjonen så refererer man non-digital til personer, organisasjoner og hendelser. Problemet her er at man ekskluderer ikkepersoner eller hendelser fra å bli klassifisert som et læringsobjekt.

David Wiley mente at denne definisjonen var for bred og definerer læringsobjekter slik:

> *"any digital resource than can be used to support learning"*

Kevin Oakes definerer læringsobjekter som:

"a self-contained unit of instruction that can be used for a number of different learning objectives"

Gene Bellinger definerte kunnskapsobjekt slik:

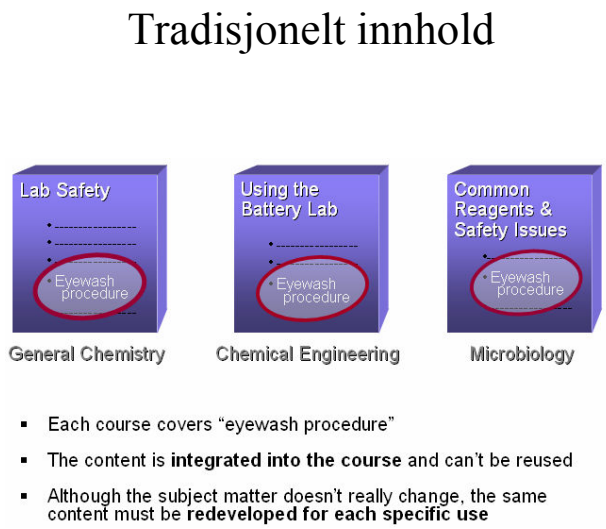
"A Knowledge object is a highly structured interrelated set of data, information, knowledge and wisdom concerning some organizational, management of leadership situation, which provides an viable approach for dealing with the situation." [Gene Bellinger]

Det eksisterer mange definisjoner av et læringsobjekt. Hvorfor er det så vanskelig å finne en som er god?

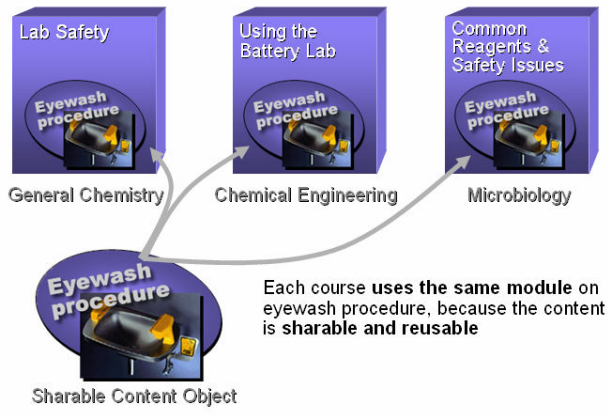
De fleste definisjonene er for generelle, og er ikke brukbare for å hjelpe oss til å forstå hva et læringsobjekt er. De hjelper oss heller ikke til å sette læringsobjekt i en praktisk kontekst istedenfor i en teoretisk.

I følge Wiley så vil lærestoffet brytes ned til små enheter kalt læringsobjekter. De skal være mest mulig kontekstuavhengige slik at de kan gjenbrukes flere ganger i forskjellige sammengheng. De første som benyttet læringsobjekter for å lage undervisningsmateriale var det amerikanske forsvaret.
Problemet de hadde var at samme instruksjon ble beskrevet i forskjellige dokumenter av ulike forfattere selv om instruksjonen omfattet nøyaktig det samme problemet.

Lærestoffet ble på denne måten ikke gjenbrukbart og delbart med andre. David Rush beskriver denne problemstillingen i et slide show der han forklarer om SCORM standarden. Der beskriver han en prosedyre for øyevask[David Rush]:



Figur 5.2: Produksjon av lærestoff på en tradisjonell måte hentet fra [David Rush]



Figur 5.3: Produksjon av et kurs med bruk av læringsobjekt hentet fra [David Rush]

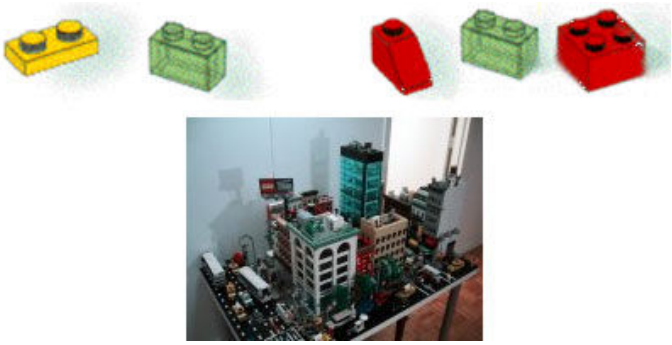
Figur 5.2 viser en prosedyre for øyevask der fagstoffet er delt opp i tre forskjellige kurs, men hver av prosedyrene er her integrert i de forskjellige kursene så det er ikke store muligheter for gjenbruk. Tar man isteden og produserer et selvstendig og delbart objekt som inneholder prosedyren for øyevask slik som i figur 5.3 så kan denne brukes av alle de tre kursene. [David Rush]

5.3.1 Metaforer

For å forklare hva et læringsobjekt er og hvordan det er tilpasset andre elementer så er det laget en rekke forskjellige metaforer.

5.3.1.1 Legoklossmetaforen

I artikkelen ”Lær av Lego” [REN] bruker man legoklossen som en metafor for et læringsobjekt. Legoklosser kan settes sammen på ulike måter og konstruksjonene kan være i ulike størrelser. Ser man på læringsobjekter så kan de også settes sammen på ulike måter til moduler som igjen settes sammen til hele kurs. Læringsobjektene har samme egenskap som legoklosser ved at de kan tas fra hverandre og erstattes men andre læringsobjekter, omgrupperes eller settes sammen på en annen måte for å dekke andre læringsbehov.



Figur 5.4: Med legoklosser kan man bygge avanserte konstruksjoner hentet fra http://careo.prn.bc.ca/losc/mod2t2.html

Med ulike elementer kan man bygge opp komplekse strukturer. Når man har laget en kompleks struktur er det arbeidskrevende og vanskelig å forandre på den siden andre elementer må demonteres før det aktuelle elementet som skal erstattes blir tilgjengelig. Dette er en svakhet i legometaforen, det er behov for en mer strukturert og begrenset metafor. Wiley påpekte også flere andre svakheter med den metaforen.

- Enhver legokloss kan kombineres med en annen legokloss
- Legoklosser kan bli satt sammen på mange ulike måter
- Legoklossene er så morsomme og enkle at til og med små barn kan sette dem sammen.

Ser man dette i forhold til læringsobjekter så blir det litt feil. Et hvert læringsobjekt kan ikke settes sammen til et annet læringsobjekt. Det er heller ikke mulig å vilkårlig sette sammen læringsobjekter for å oppnå et godt læringsopplegg. Han kom derfor opp med en annen metafor som han kaller Atom metaforen.

5.3.1.2 Atommetaforen

Atomet skiller seg fra legoklossen på følgende måter:
Ikke alle atomer er kombinerbare.
Atomer kan bare bli satt sammen i bestemte strukturer bestemt av indre oppbygging.
Det kreves noe trening for å sette samme atomer.

Figur 5.5: Kombinasjon av atomer hnetet fra
http://careo.prn.bc.ca/losc/mod2t2.html

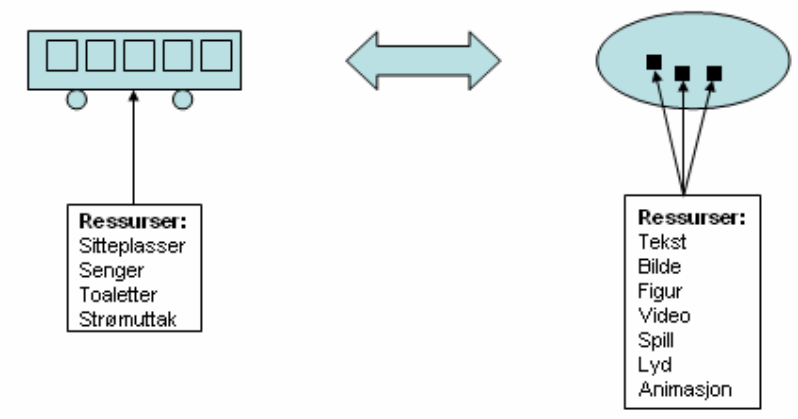
Det er store forskjeller på disse to metaforene. legometaforen er nok noe for enkel men den har en fordel ved at Lego er allment kjent produkt og egner seg godt for å forklare begrepet læringsobjekt. For å forstå Wileys atommetafor trengs det kunnskap om kjemiske kombinasjoner.

Atommetaforen er også svak når man tenker på kjemiske reaksjoner, kombinerer man atomer som reagerer med hverandre så får man et nytt produkt med helt andre egenskaper enn de opprinelige.

5.3.1.3 Togmetaforen

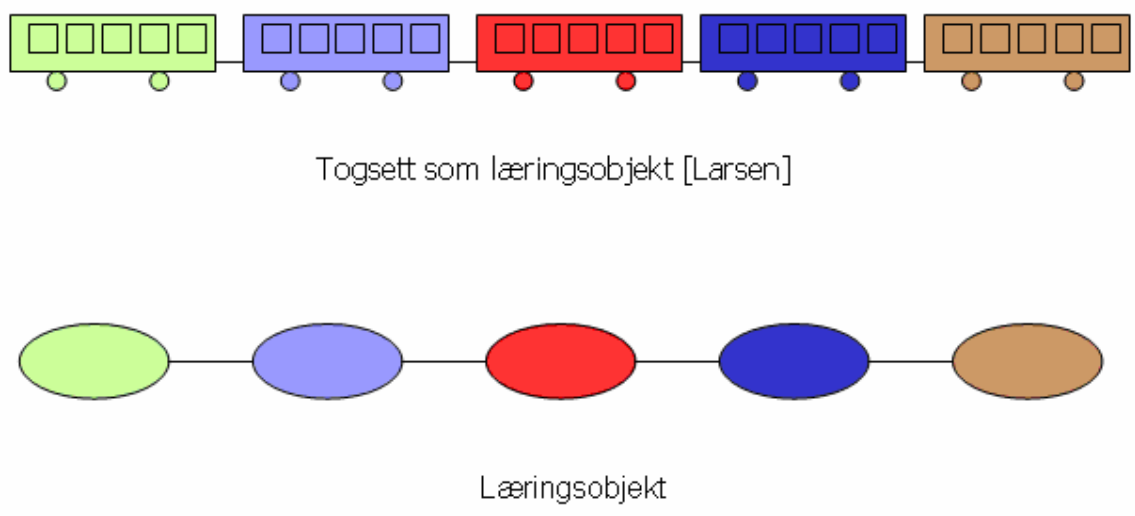
Alle vet hva er tog er og at det kan bestå av flere vogner. I togmetaforen inngår elementet togvogn og togvogner. Disse utgjør et togsett. En togvogn vil være satt sammen av ulike ressurser som sitteplasser, toaletter, senger, strømuttak, etc. I denne metaforen så tilsvarer en togvogn et kunnskapsobjekt. Disse kunnskapsobjektene vil bestå av flere forskjellige læringsressurser som for eksempel tekst, bilde, video, animasjon, lyd, etc. En togvogn har som oppgave å frakte passasjerer på en komfortabel måte. Kunnskapsobjektene har på sin side en oppgave å gi den lærende et læringsutbytte.

Kunnskapsobjektene må være kontekstuavhengige og selvstedige slik at de lett kan benyttes i flere læringssammenheng.



Figur 5.6: Togvogn som kunnskapsobjekt

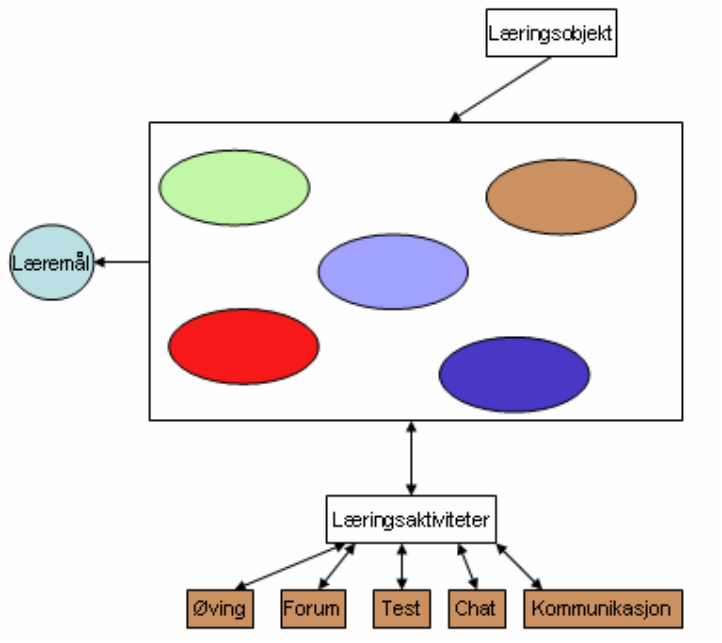
Det eksisterer ene rekke forskjellige togvogner som for eksempel tankvogn, kjølevogn, passasjervogner, godsvogner og lignende. Man vil også ha flere typer kunnskapsobjekter. Ved å kombinere flere kunnskapsobjekter til en større helhet, vil vi få et læringsobjekt.



Figur 5.7: Aggregering av kunnskapsobjekt til læringsobjekt

Siden det er det samme grensesnittet mellom alle togvognene så er det lett å bytte ut en togvogn med en annen slik at man får et nytt togsett. På samme måte skal det være enkelt å bytte ut et kunnskapsobjekt med et annet for å få et nytt læringsobjekt. Dette vil være viktig når man skal individtilpasse lærestoffet til forskjellige elever.

Hvis læringsløpet er strukturert med en bestemt progresjon så vil modellen vist i figur 5.7 være et eksempel på design av et læringsobjekt. Kunnskapsobjektene vil da bli tolket i en bestemt rekkefølge slik som man leser en bok.



Figur 5.8: Læringsobjektet i en kontekst

Figuren over viser at det er koblet et læringsmål og en del læringsaktiviteter til læringsobjektet. Læringsmålet og læringsaktivitetene er her lagt utenfor selve læringsobjektet. Dette gjør at læringsobjektet blir mer selvstendig og vil være lettere å gjenbruke. Aktivitetene vil variere fra om det er campusundervisning eller en ren nettbasert undervisning.
Ved å sette det opp slik vil ikke læringsobjektet låses fast til bestemte aktiviteter, men man kan tilpasse aktivitetene til metode og den lærendes forkunnskaper.

Ved å koble flere læringsobjekter sammen til en større helhet får vi en leksjon og flere leksjoner vil utgjøre en modul. Vi vil da få et hierarki med læringsressurser på laveste nivå og et studieprogram på det høyeste nivå se tabell 5.1.[Larsen]

Innhold	Innholdsbeskrivelse
Studieprogram	Et eller flere kurs
Kurs	En eller flere moduler
Modul	En eller flere leksjoner
Leksjon	Et eller flere læringsobjekt
Læringsobjekt	Et eller flere kunnskapsobjekt
Kunnskapsobjekt	En eller flere ressurser
Læringsressurs	Tekst, bilde, figur, video, lyd, animasjon, spill

Tabell 5.1: Innholdshierarki

6 Visualisering

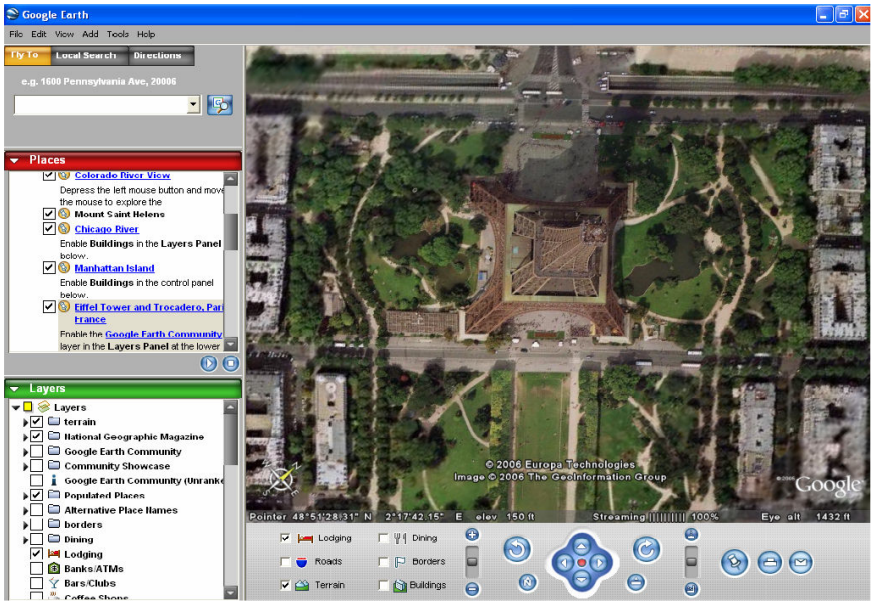
6.1 Visualisering av informasjon

I følge Schneiderman Ben 1996 så kan visualisering sees på som bruk av visuelt interaktive presentasjoner av data for å forsterke forståelse.[Scneiderman, Ben]
Med dette mener han at informasjon presenteres i en visualisert form.

Et ordtak som dette er velkjent: et bilde sier mer enn 1000 ord. For noen oppgaver, men ikke alle, er det bedre med en visuell presentasjon. For eksempel er en visualisering ved bruk av et fotografi eller et kart lettere å forstå enn en tekstlig beskrivelse.

Man kan for eksempel se på et kart over USA som inneholder turistinformasjon. For en bruker som vet hvor byen man leter etter er, så er det enkelt å trykke på byen for å få fram turistinformasjonen. En turist som ikke vet hvor byen er lokalisert vil kanskje ha større behov for en liste der han velger byen for å få fram turistinformasjonen. Kartet bør da kunne vise turisten hvor denne byen er i USA ved å forandre farge eller ha et blinkesignal ved lokasjonen av denne byen.

Med stadig kraftigere datamaskiner og høyere oppløsning så vil informasjonsvisualisering og grafiske grensesnitt bli mer ettertraktet. I dag eksisterer det avanserte kartsystemer som for eksempel Google Earth. I nærmere framtid vil det sannsynligvis dukke opp enda mer avanserte systemer.



Figur 6.1: Skjermbilde fra Google Earth der man viser Eiffeltårnet

Mennesket har meget gode evner når det gjelder å prosessere visuell informasjon. Brukere kan skanne, gjenfinne og huske bilder. De kan registrere forandringer i farger, former, bevegelser, størrelser og teksturer. Programmer som visualiserer informasjon vil derfor være et veldig godt alternativ til den vanlige lineære tekstboka.

Jeg vil i dette kapitlet beskrive hva man bør være oppmerksom på når man skal implementere og bruke en visualiseringsmetode. I tillegg til dette vil jeg nevne en rekke ulike metoder som Shneiderman har foreslått.

6.2 Viktig å vite før man implementerer et informasjonsvisualiseringssystem

For at et informasjonsvisualiserings system skal bli vellykket er det viktig å vite om en del aspekter før man begynner å lage et slikt system.
Disse aspektene kan virke begrensede på hverandre og det vil derfor være opp til hvert enkelt system en vurderingssak hva man vil legge mest vekt på.

6.2.1 Størrelse

Størrelse er viktig ved visualisering. Jo større og høyere oppløsning man har jo flere kunnskapsobjekter kan man vise. Jo mer informasjon som skal visualiseres tar det også lenger tid før programmet får presentert denne visualiseringen.
Store visualiseringer blir ofte uoversiktlige i motsetning til mindre. Brukeren vil lettere kjenne seg igjen i små visualiseringer enn i store. Løsninger som gjør det mulig å vise informasjonene i hierarkier vil gjøre at brukeren lettere klarer å navigere i visualiseringssystemet. Visualisering i hierarkier kan også gjøre at visualiseringsprosessen blir raskere.

6.2.2 Forutsigbarhet

Hver gang man benytter samme visualiseringsteknikk på den samme eller tilnærmet lik informasjon så bør det resultere i tilnærmet lik presentasjon. Det er viktig for brukeren å kunne beholde det mentale bilde fra gang til gang. Forandrer en graf seg hver gang man starter et visualiseringsprogram så vil det være vanskeligere for brukeren å kjenne seg igjen og bruke den enn om den var stabil (konstant bilde). Hvis presentasjonen ikke er forutsigbar, så kan det føre til forvirring og uoversiktlighet.

6.2.3 Tidskompleksitet

I en visualiseringsmetode er det viktig at samhandling mellom brukeren og systemet skjer i sanntid. Dette fordi et informasjonsvisualiseringssystem skal være interaktivt. For å gjøre et visualiseringsprogram til et sanntidssystem så må man sette store krav til ytelse. Hver av metodene som blir brukt til å visualisere må gjøres effektiv, slik at brukeren sitter igjen med en interaktiv følelse.
Man kan også sette krav til nøyaktighet. Jo mer nøyaktig man skal visualisere jo mer datakraft og tid vil man som regel trenge for å få fram en ønsket nøyaktighet.

6.2.4 Utseende og navigasjon

Hvordan man planlegger utseendet i et visualiseringssystem er viktig. Det må være oversiktlig og forståelig nok slik at man kan trekke ut informasjon fra den. Den må være enkelt å navigere i og den må løse eventuelle desorienterings problemer.

I informasjonsvisualiseringssystemer oppstår det vanligvis desorienteringsproblemer når man prøver å visualisere mer en 400-500 objekter. For å redusere orienteringsproblemene så kan man ta i bruk en rekke metoder. [Hypermedia]

«Gå hjem» mulighet

Brukere får som regel en startnode ut fra en spørring. Ved hjelp av en «gå hjem» knapp kan brukeren når som helst komme tilbake til denne startnoden. Hvis brukeren roter seg bort eller føler at de navigasjonsvalgene kan har tatt ikke har fungert, så kan man benytte denne «gå hjem» muligheten. Hvis derimot tidligere noder har vært av interesse så kan «gå tilbake» mulighetene benyttes.

«Gå tilbake» mulighet

Systemet lagrer brukerens bevegelser. Alle noder som er besøkt blir lagret. Dette gjør at alle tidligere besøkte noder kan nåes ved å velge «gå tilbake» metoden. Hvis brukeren roter seg bort, kan han redde situasjonen ved å bevege seg bakover til forrige node inntil han kommer over en node han kjenner igjen og brukeren kan forsette navigasjonen ut fra denne noden.

«Forrige noder » mulighet

En annen variant av «gå tilbake» muligheten er å la brukeren gå direkte tilbake til en hvilken som helst av de foregående nodene. Dette gjøres ved at brukeren får opp en liste over besøkte noder og kan velge hvilken av disse han vil gå til.

Øyeblikksbilder

Øyeblikksbilder oppnås ved at visualiseringen fryses i den tilstanden den er i. Brukeren kan seinere komme tilbake til denne tilstanden for så å fortsette videre.

Personlige stier

Personlige stier er en mulighet som tilatter å lagre en sekvens av noder. Disse stiene blir laget av veiledere eller av studenten selv. Disse stiene kan seinere åpnes opp slik at brukerne kan følge den lagrede sekvensen av noder.

Bokmerker

For å enkel kunne gå til utvalgte noder, så kan man lagre disse som ”bokmerker”. Brukeren merker noder som han seinere vil gå tilbake til. Disse kan fra systemet hentes opp igjen, de kan for eksempel presenteres som en liste som brukeren kan velge fra. På samme måte som favoritter brukes i Internet Explorer.

Mulighet for å holde viktige noder åpne

Mulighet for å ha flere noder oppe av gangen er viktig for å orientere seg rundt i systemet. Grunnen til dette er at hvis man kun kan ha en node oppe av gangen så fører dette til forvirring hos brukeren, siden han ikke får et overordnet syn på informasjonen. Dette kan videre føre til at brukeren ikke forstår hvorfor han har kommet dit han er, eller hvordan han kan komme videre.

Merking av besøkte noder.

Ved å merke de nodene som har vært besøkt så kan brukeren lett se hvilke noder som er besøkt og hvilke noder som ikke er besøkt. Det er to teknikker som benyttes her:

Frekvensbilde: Gi brukeren mulighet til å få fram et oversiktsblide der det vises hvilke noder som er besøkt og hvilke noder som ikke er besøkt.

Tidsmerker: En annen måte å vise at en bruker har besøkt noden tidligere er å merke selve noden med tidspunktet noden ble besøkt sist. Dersom det ikke er registrert noe tidspunkt så har ikke noden blitt besøkt..

Nodenavn

Vise navnet på noden. Hver node bør ha et unikt navn som brukerne kan forbinde innholdet av noden med. Ved å bruke gjennomtenkte titler på hver node så vil brukeren ha lettere å forstå hva som ligger i den noden som brukeren vurderer å åpne.

Framgangsindikator.

Man kan vise brukerens innsats i et system ved å vise en fremgangsindikator. Dette kan være prosentantall av totalt antall noder, eller andre markører som viser hvor mange av nodene som har vært besøkt.

6.2.5 Oppgaver i visualiseringssystemer

Shneiderman beskriver 7 viktige oppgaver som informasjonsvisualiseringssystemer bør inneholde. Ikke alle oppgavene er like relevant for alle visualiseringssystemer eller like lett å implementere, men de bør vurderes i planlegging av et visualiseringssystem.

Overblikk: Mulighet for å gi et overblikk over alle objektene som visualiseres av systemet. Dette blir ofte vist som et konseptkart eller andre graflignende presentasjoner.

Zoom: Mulighet for å zoome inn på objekter som er av interesse. Brukere har som regel interesse for en del av informasjonsmengden. Mulighet for å kontrollere fokus ved å stille inn en zoom faktor er derfor viktig.

Filter: Mulighet for å filtrere ut uønskede objekter. Gi brukeren mulighet til raskt å fokusere på objekter av interesse ved å eliminere uønskede objekter.

Detaljer på kommando: Mulighet for å velge objekter eller grupper av objekter for å hente ut detaljer når det behøves.

Relasjoner: Vise sammenheng mellom objektene som representeres i visualiseringssystemet.

Historie: Lagre historieinformasjon om besøkte noder. Mulighet for å kunne gå tilbake til tidlige besøkte noder, eller få fram informasjon som viser hvor ofte noder har blitt besøkt.

Ekstraktere informasjon: Tillatelse til å hente ut informasjon fra subkomponenter i visualiseringssystemet, mulighet for å lagre disse subkomponentene for seinere å hente de opp igjen eller sende dem til andre via e-mail eller bruke dem i andre programmer.

6.3 Forskjellige informasjonsvisualiseringstekniker (datatyper)

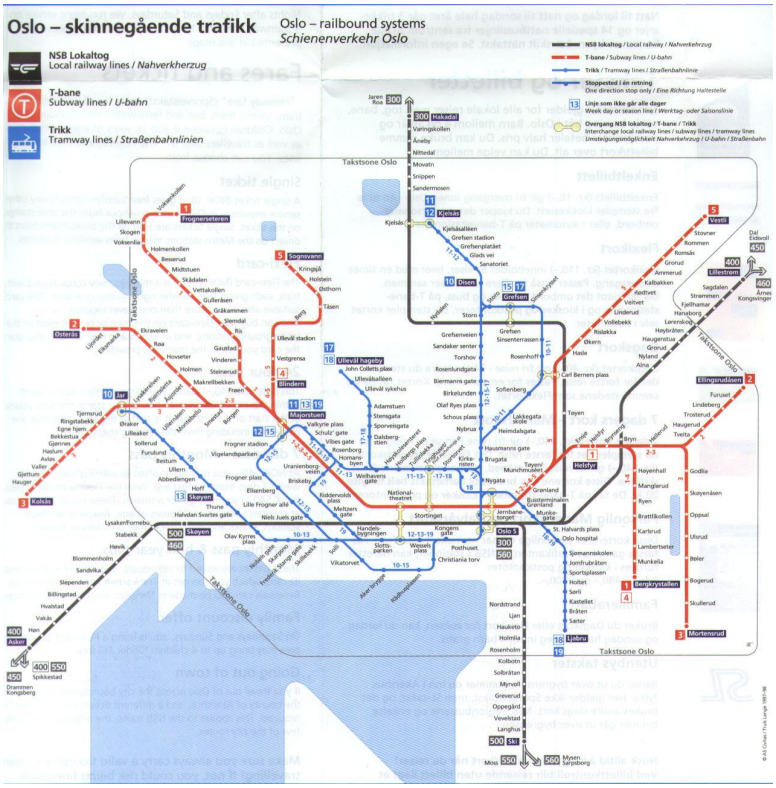
Schneiderman foreslår en måte å sortere ulike typer informasjonsvisualiseringsteknikker i forhold til datatyper.
Jeg vil her kort beskrive de sju typene som Shneiderman nevner.

6.3.1 Endimensjonal

En lineær datatype. Den inkluderer tekstdokumenter, programkode og alfabetiske lister. Hvert element i samlingen er en linje med tekst som inneholder en rekke bokstaver.
For å vise store mengder med endimensjonale data så skapte man systemer der man gir detaljert informasjon om et fokusert område og mindre informasjon om område utenfor det fokuserte området.

6.3.2 Todimensjonal

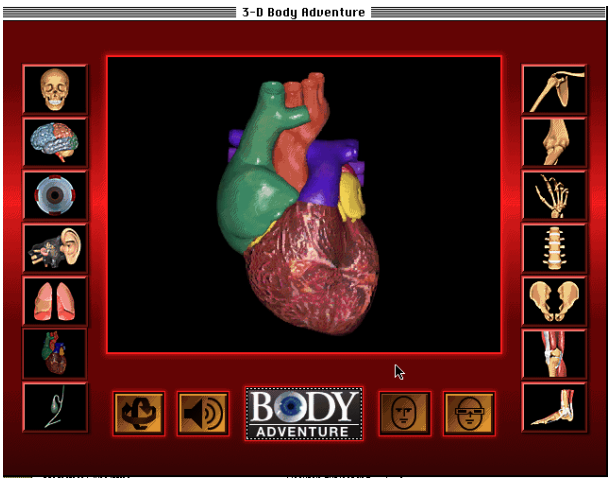
Typiske eksepler på todimensjonale presentasjoner kan være geografiske kart, brannkart som viser rømningsveier og avislayout. Hvert element i en todimensjonal presentasjon har som regel navn, eier, verdi og lignende attributter. De har kjennetegn som størrelse, farger og lignende. Et eksempel på en slik todimensjonal presentasjon vises i figure 6.2 fra Oslos t-bane.



Figur 6.2: Todimensjonal presentasjon av Oslo T-bane hentet fra http://www.tbane.no/

6.3.3 Tredimensjonal

Brukes for å representere objekter fra den virkelige verden slik som for eksempel molekyler og deler av den menneskelige kropp. Arkitekter bruker objekter i avanserte 3d programmer for å konstruere bygninger. I figur 6.3 er det et skjermbilde hentet fra programmet 3d body adventure som gir brukeren mulighet til å se 3d modeller av den menneskelige kropp.



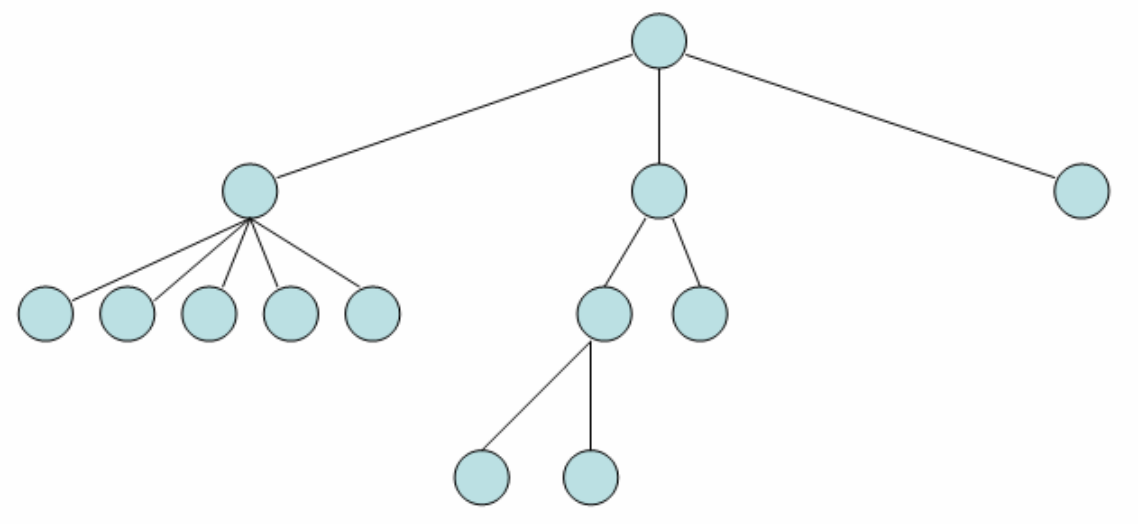
Figur 6.3: Skjermbilde fra Body adventure. Illustrasjon av et hjerte i 3d hentet fra http://www.uvm.edu/~jmorris/3d.gif

6.3.4 Mulitdimensjonale

Relasjon og statistikkdatabaser blir manipulert som multi-dimensjonale data der objekter med n attributter blir til punkter en n-dimensjonalt rom. Interface representasjonen kan være i 2d eller 3d form vist som spredningsdiagrammer som blir kontrollert av en rekke glide knapper. For eksempel så laget FilmFinder en slik teknikk for å presentere multidimensjonale data over filmer, ved hjelp av knapper og glidemenyer.

6.3.5 Trær

Trær er en bestemt type graf, de blir representert ved hierarkier eller trestrukturer der samlinger av objekter har en link til en forelderobjekt. Det er kun hovedforeldren (rotnoden) som ikke har en link til en forelder. Et eksempel på et simpelt tre vises i figur 6.4.



Figur 6.4: Eksempel på et tre

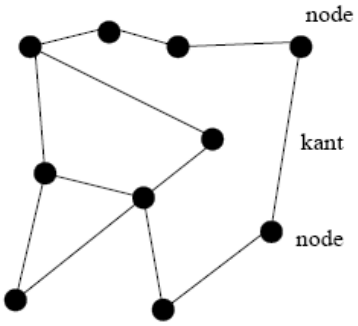
6.3.6 Nettverk

Det eksisterer en rekke relasjoner som ikke kan vises godt ved å bruke en trestruktur
Disse representasjonene blir ofte definert som nettverk. I min oppgave så vil jeg bruke ett nettverk kalt Springembedder.

6.4 Grafer

I min oppgave vil det bli laget en tredimensjonal graf. Jeg vil derfor i dette kapitlet forklare hvordan en graf blir bygd opp og hvilke forskjellige typer grafer som er mest vanlige.

En graf består av noder og kanter. Kantene i grafen forbinder to noder med hverandre. Data blir representert av nodene i grafen mens kantene representer relasjonen mellom nodene. En node kan inneholde enkel data som for eksempel et tall eller være konteiner for store mengder med forskjellig data. En node kan for eksempel inneholde en peker til en avansert interaktiv presentasjon.
Graf er en fleksibel datastruktur der nesten alt kan moduleres. Ser man for eksempel på et kart over gatene i en by, så inneholder kartet gater og gatekryss. Hver gate forbinder to gatekryss. I dette kartet vil gatekryssene være noder og gatene vil være kanter.



Figur 6.5: Graf som viser et veinettverk

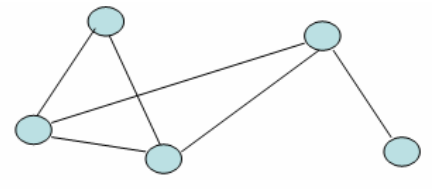
Et slikt kart brukes til å finne fram i byen. Befinner man seg for eksempel på Stortorget i Oslo og vil til Nils Abels hus på Blindern, tar man en kikk på kartet. Man bruker kartet til å finne korteste vei fra Stortorget til Nils Abels hus på Blindern. I grafer har man også begrepet vei som er en sekvens av påfølgende kanter som forbinder to noder. I grafverdenen søker man svar på spørsmål som:

- Hvordan finne en vei mellom to noder eller bevise at en vei mellom disse nodene ikke eksisterer?
- Hvordan finne korteste vei mellom to noder? Har man for eksempel en viss lengde knyttet til en kant kan man med algoritmer finne optimal rute.

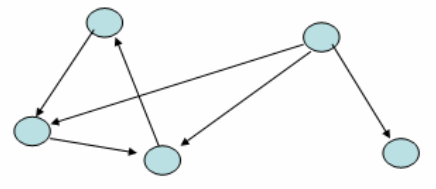
Disse problemene løses ved hjelp av algoritmer og prosedyrer, jeg vil ikke komme inn på disse algoritmene i min oppgave men jeg vil beskrive forskjellige typer grafer.

6.4.1 Forskjellige typer grafer

En graf som har noder som er ordnet, der rekkefølgen på nodene har betydning blir kalt en rettet graf. I motsatt fall kalles grafen urettet.

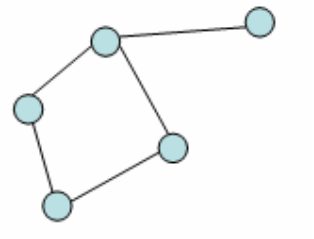


Figur 6.6: Urettet graf

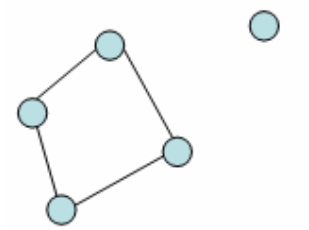


Figur 6.7: Rettet graf

En urettet graf er sammenhengende dersom det er en vei fra hver node til en annen node, hvis der eksisterer noder i grafen som ikke er koblet til andre noder i grafen kalles den en ikke sammenhengende graf.



Figur 6.8: Sammenhengende graf



Figur 6.9: Ikke sammenhengende graf

6.4.2 Graftype brukt i min oppgave

I min oppgave vil nodene representere dokumenter, læringsobjekter eller kunnskapsobjekter. Kantene i grafen vil representere kobling mellom to noder.

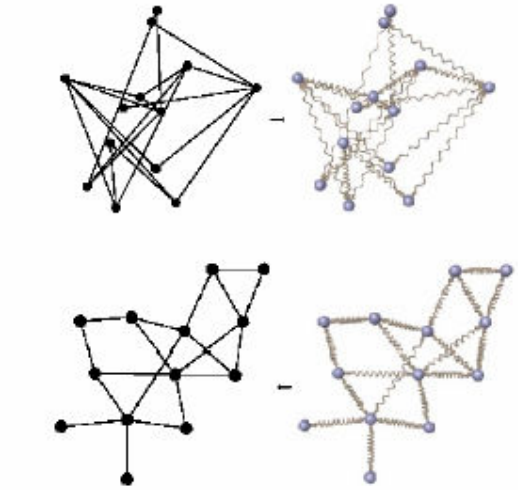
Hvis programmet tegner opp kanter mellom alle nodene i programmet vil man får en sammenhengende urettet graf. En slik graf blir fort uoversiktlig så det vil bli benyttet terskelverdier for å bestemme om det skal tegnes en kant mellom to noder eller ikke. Ved hjelp av disse terskelverdiene vil man transformere grafen fra en sammenhengende urettet graf til en ikke sammenhengende urettet graf.

Jeg velger derfor å si at jeg konstruerer en ikke sammenhengende urettet graf. For å konstruere denne grafen benytter jeg meg av en plasseringsalgoritme kalt spring embedder.

6.5 Spring embedderalgoritmen

6.5.1 Eades

I 1984 laget Eades en algoritme for å tegne opp grafer basert på et mekanisk system. Nodene blir byttet ut med stål ringer og kantene blir byttet ut med fjærer knyttet til ringene.



Figur 6.10: Omforming av graf til fjær og ringer

Alle fjærene har den samme naturlige lengden k og hver fjær har lengden d. Behandling av et par ringer som er koblet til en fjær skjer ved å sammenligne k og d. Er k>d så trekker fjæren ringene sammen. Hvis k<d så frastøtter fjæren ringene. Når k=d så er ringene stabile. Algoritmen fortsetter å kjøre til systemet oppnår en minimal energitilstand. Styrken til kreftene blir bestemt av k og d. Eades benytter to formler for å beregne kreftene. En for den frastøtende kraften og en for den tiltrekkende kraften.

Den frastøtende kraften gjelder for noder som ikke er direkte koblet og blir beregnet med formelen:

$$f_r(d) = \frac{k_r}{d^2}$$

For å regne ut den tiltrekkende kraften så brukes formelen:

$$f_a(d) = k_a \log(d)$$

6.5.2 Fruchterman og Reinhold

Fruchterman og Reinholdt gjorde en del modifikasjoner på den originale spring embedderen som ble laget av Eades.
Fruchtman baserer seg på å tegne opp en connected undirected graphs (en graf hvor alle noden er koblet til minst en annen node og at alle nodene er koblet indirekte)

For å tegne opp en connected undirected graph så trengs det antall noder minus 1 dimensjon for at alle kantene skal kunne tegnes opp med sin eksakte lengde (Brandes, Ulrik 2001).
Det vil si at hvis man skal tegne opp en graf i tre dimensjoner så kan det bli umulig å tegne opp eksakte lengder på kantene hvis det er flere en 4 noder i grafen.
Siden man ikke klarer å se mer en i tre dimensjoner blir oppgaven å redusere grafens dimensjoner til tre og forsøke å få den til å bli så lik den reelle grafen som mulig.

Ut fra kreftene som virker inn på den visuelle grafen, tiltrekking ved fjærer og frastøting mellom objekter er målet å redusere grafens energinivå til et minimum, jo lavere energinivå man har i den visuelle grafen desto nærmere er man sannsynligvis den reelle grafen.

Hovedbegrunnelsen for å modifisere algoritmen var å få en mer effektiv algoritme slik at beregningene kunne gå raskere.
Denne metoden bruker også frastøtende og tiltrekkende krefter.
Alle noder frastøter hverandre men de som er forbundet men en kant tiltrekker hverandre i tillegg.

Formlene for å beregne tiltrekkende og frastøtende krefter er:

$$f_a(d) = \frac{d^2}{k} \qquad \text{Tilltrekende kraft}$$

$$f_r(d) = -\frac{k^2}{d} \qquad \text{Fratøttende kraft}$$

Kreftene summeres for å beregne kraften mellom nodene.
Forskjellen på metodene er at økningen i kraft er mye raskere i Fruchtermanmetoden enn i metoden til Eades når nodene ikke er plassert i sine naturlige posisjoner. Dette fører til en raskere konvergering mot naturlig posisjon som gjør at algoritmen blir raskere

6.5.3 Metoden som er tatt i bruk i programmet

I programmet bygges det en undirected complete graph, dette er en graf hvor alle nodene er koblet til andre noder. I motsetning til Fruchtman som benytter seg av en undirected connected graph (en graf hvor kantene ikke har retning og nodene er koblet direkte eller indirekte gjennom andre noder)

Selve metoden er en videre modifikasjon fra Fruchtermanalgoritmen og er laget av Tore Vestues.

I denne implementasjonen dannes det en complete graf. Algoritmen har blitt forenklet ved at man ikke trenger å beregne frastøtende krefter på samme måte som i Fruchtermanalgoritmen. Dette fordi det ikke finnes noder som ikke er koblet til hverandre. Ideallengden mellom nodene er ulik og beregnes ut fra likhetsverdiene mellom nodene og fjæra som er koblet mellom dem.

Funksjonen for kraften i fjæra blir beregnet med formelen:

$$f(d_{(i,j)}) = d_{i,j} * (s_{(i,j)})^3 * d_{(i,j)} - 1/d^2_{(i,j)}$$

Hvor S(i,j) er likhet mellom nodene i og j (verdi mellom 0 og 1)
D(i,j) er den euklidske distansen mellom nodene i og j

Distansen regnes ut med formelen:

$$distanse = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Selve metoden itererer et gitt antall ganger, bestemt av en konstant satt i programkoden.
For hver iterasjon justeres hver node i grafen slik at den totale energien i systemet reduseres.
Det vil si at for hver derivert med hensyn på x, y og z skal man prøve å finne nullpunktet.
Dette blir ikke gjort ved en enkel utregning, men blir gjort ved Newton Raphsonmetoden som iterativt prøver å nærme seg nullpunktet.

Newton Raphsonmetoden er en iterativ metode som beregner seg fram til en gitt verdi som ikke er gitt i utgangspunktet.

Funksjonen til Newton-Raphson:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Her er x_n den nåværende kjente x verdien.
$f(x_n)$ representerer verdien til funksjonen ved x_n og $f'(x_n)$ er den deriverte ved x_n.
x_{n+1} representerer den neste x verdien som man prøver å finne.
f'(x) representerer f(x)/dx derfor er termen f(x)/f'(x) en representant for verdien dx.
Jo flere iterasjoner som blir kjørt jo nærmere vil dx bli null.

6.5.4 Fordeler og ulemper med spring embedderen

Fordeler:

- Programmeringsmessig en intuitiv modell som relaterer til en fysikk som er reell.
- Er enklere å implementere en andre visualiseringsteknikker
- Layouten til grafen blir ofte god og nøyaktig, nodene sprer seg jevnt utover når grafen ikke blir for stor.

Ulemper:

Spring embedder metoden er iterativ og en rekke kalkulasjoner utført mange ganger. Metoden blir derfor i visse tilfeller ikke rask nok når grafen blir for stor, og man mister da den interaktive følelsen.

Ved visualisering av to nesten identiske grafer kan man få helt ulike resultater siden nodene blir plasser tilfeldig og kreftene kan virke svært ulikt på plasseringen. Dette gjør grafen svært uforutsigbar.

7 Kunnskapsrepresentasjon

Kunnskapsrepresentasjon er den måten vi utrykker kunnskap på slik at andre kan forstå den. [UIO]

Kunnskapsrepresentasjon går ut på hvordan man lagrer og manipulerer kunnskap i et informasjonssystem i et slikt format at det kan bli brukt av mekanismer for å fullføre en gitt oppgave.[WIKIPEDIA]

Kunnskapsrepresentasjon blir brukt i kunstig intelligens ved problemløsning. Kunnskap blir med datamaskiners hjelp her gjort om til et format slik at problemløsningen blir lettere å utføre, andre format kan derimot gjøre at problemløsningen blir vanskeligere å utføre[Davis]

Kunnskapsrepresentasjon (KR) blir brukt til å referere til representasjoner som er laget for å kunne prosesseres av en moderne datamaskin. Slike representasjoner består av eksplisitte objekter og informasjon om objektene. Man kan ha objekter som for eksempel dataprogrammer og Microsoft Word. Informasjonen eller forholdet mellom dem kan være at Microsoft Word er et dataprogram. Ved å bygge opp forhold mellom objekter så kan datamaskinen trekke konklusjoner ut fra forholden mellom objektene basert på spesifiserte regler.

På 1980 tallet oppstod det en rekke kunnskapsrepresentasjons språk og systemer. Et Kunnskapssystem består av en måte å representere kunnskap på i tillegg til en rekke metoder som kan brukes på KR-systemet for å avlede ny kunnskap eller mulighet for å søke etter kunnskap i representasjonen. Et representasjonsspråk som for eksempel Prolog, bruker for eksempel proposisjoner og logikk for å trekke konklusjoner basert på gitte premisser.

Davis beskriver i artikkelen sin kunnskapsrepresentasjon gjennom 5 roller[Davis]:

Den første rollen beskriver han kunnskapsrepresentasjon som en stedfortreder. Dette er en erstatning for den virkelige kunnskapen. Tanken er her å representere kunnskapen i et annet format som gjør at man kan bedømme konsekvenser ved tenkning isteden for handling. Man resonerer om verden istedenfor å utføre handlinger i den.

Rolle to går på å svare på spørsmålet: Hvilke termer kan jeg bruke for å resonere om verden? Systemer vil gi mulighet for å fungere som et sett med sterke briller, som bestemmer hva man kan se ved å vise visse deler av verden i skarp fokus på bekostning av andre deler av verden.

Den tredje rollen går på hvordan personer resonerer i det virkelige liv, og hva som menes med å resonere intelligent.

Den fjerde rollen går på at KR er et medium for effektiv beregning.
Resonering i datamaskiner en resonerende prosess der man benytter beregninger for å finne svarene man er ute etter. For å bruke en representasjon så må man kunne gjøre beregninger mot den.

Den femte rollen går på medium for menneskelig utrykk: et språk der man sier ting om verden.[Davis]

Målet med min oppgave er å presentere kunnskap på en slik representasjon at brukeren av systemet skal enkelt kunne hente ut kunnskap fra den.

Det må lages en metode for å representere brukerens kunnskap på en datamaskin, det må også lages en metode for å presentere denne kunnskapen visuelt. Siden datamaskinen ikke skal avlede kunnskap fra denne representasjonen så blir det ikke så interessant å snakke om metodene i kunnskapsrepresentasjon i min oppgave, derfor vil jeg ikke legge stor vekt på dette.

7.1 Kunnskap

Kunnskap er informasjon som en person vet om. Kunnskap brukes også for å betegne sikker forståelse for et emne. Potensielt for å bruke denne ferdigheten for et spesifikt formål.[WIKIPEDIA]

Når det er snakk om kunnskap i kunnskapsrepresentasjon så er det to forskjellige hoveddeler man kan dele kunnskapsrepresentasjonen i, disse kalles deklarativ og prosedural kunnskap. [IT 2702]

Deklarativ kunnskap representerer generelle sammenhenger og konkrete fakta, uavhengig av hvordan de er tenkt utnyttet.
Prosedural kunnskap representerer kunnskap i form av en prosedyre som hvis den utføres med suksess, gir den aktuelle kunnskapen som resultat.

```
(for-all ?x ((is-a bird ?x) <- (is-a animal ?x)
AND (lays eggs ?x)
AND (has feathers?x))
```

Eksempel kode for å beskrive forskjellene mellom deklarativ og prosedural kunnskap

Hvis man tolker koden ovenfor så tolkes den forskjellig i forhold til hvilken kunnskapstype man har valg. En deklarativ tolkning sier at: Alle dyr som har fjær og legger egg er fugler. Mens den prosedurale tolkningen forklares slik: For å finne ut om noe er en fugl, sjekk først om det er et dyr, deretter om det har fjær, og til slutt om det legger egg.

Roger T Hartley diskuterer i sin artikkel Representation of procedural knowledge hvordan disse to kunnskapstypene kan presenteres i samme representasjon. Han mener at en form for kunnskap er utilstrekkelig uten den andre. Med dette mener han at hvis man velger en form for kunnskap over en annen form så kan det lede til en filosofisk falskhet på det verste. [Hartley]

Han forklarer forskjellen mellom de to kunnskapstypene slik:
Verden inneholder objekter og handlinger; romlig forhold mellom objekter er deklarative; temporære forhold mellom handlinger er prosedurale.

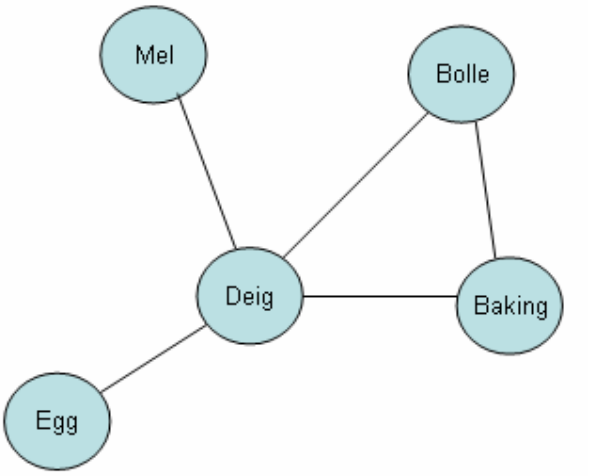
Deklarativ kunnskapsrepresentasjon har relasjoner som plasseres i rom, mens prosedural kunnskapsrepresentasjon beskriver forbindelser som har med tid å gjøre.

I for eksempel databaseteknikk så er deklarative kunnskap å kjenne til de forskjellige normaliseringsformene, prosedural kunnskap benyttes når man skal bruke kunnskapen om normalformen for å normalisere en relasjon.

7.1.1 Deklarativ kunnskap

Harley definerer deklarativ kunnskap som et sett med deklarasjoner av objekter og relasjoner mellom objektene.
Disse deklarasjonene inneholder ikke informasjon om hvordan man skal utlede kunnskap, den beskriver kun kunnskapen om objektene og relasjonene. For å presentere denne typen kunnskap trenger man ikke mer en et sett med objekter og et sett med kanter, mellom objektene. En slik graf kan i sin enkleste form være en graf uten retningsbestemte kanter, der kantene forteller at det er en relasjon mellom nodene. [Hartley]

Den deklarative kunnskapstypen gir ikke informasjon om hvordan, men gir en oversikt over hvilke konsepter som eksisterer og hvordan de henger sammen. Ser man på figur 7.1 så viser man her en representasjon av en persons deklarative kunnskap om baking. Ut fra grafen kan vi lese at mel og egg har en relasjon til deig, mens bolle og deig er knyttet til baking.
Som sagt så gir dette ikke noe informasjon om hvordan man skal bake, men man får en oversikt over konseptene og sammenhengen mellom dem.



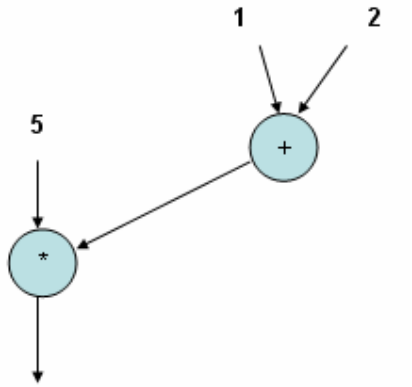
Figur 7.1:Deklarativ kunnskap om baking

7.1.2 Prosedural kunnskap

Deklarativ kunnskap sier kun hva slaks konsepter man har med å gjøre og hvordan de henger sammen. Prosedural kunnskap er kunnskap om hvordan man skal uføre en oppgave som for eksempel baking.

Skal man for eksempel utføre en matematisk beregning kan man representere denne aktiviteten som prosedural kunnskap. Å beskrive hvordan man løser et enkelt regnestykke kan gjøres ved hjelp av en trestruktur.
Man kan ta regnestykket 5*(1+2) og representere det som prosedural kunnskap som vist i figur 7.2.
Det er regler som må følges i denne representasjonen, man kan bare utføre en operasjon når alle input er tilstede.

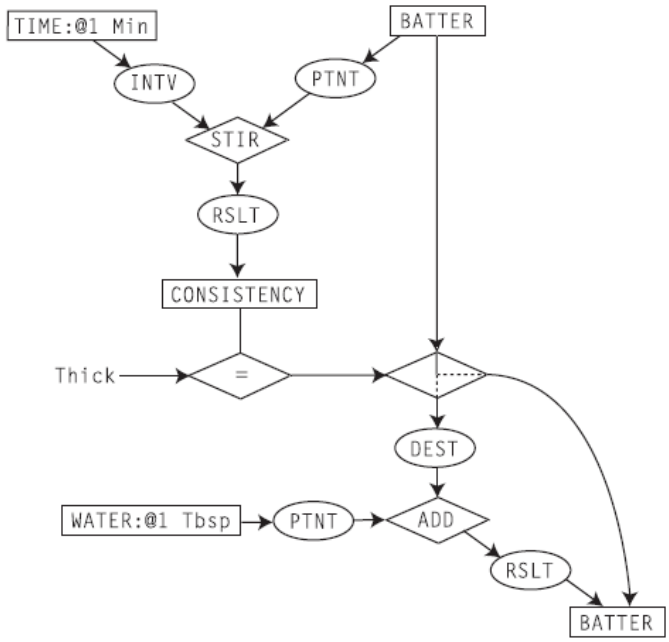
I eksemplet har man 2 input til multiplikasjonen men den kan ikke utføres før alle inputverdiene er klare. Derfor må addisjonen 1+2 utføres først. Når addisjonen er utført så vil multiplikasjonen ha alle inputverdiene og kan derfor utføre sin operasjon



Figur 7.2: prosedural kunnskap representert som et tre

Grafer som representerer prosedural kunnskap inneholder mer informasjon enn grafer som inneholder deklarativ kunnskap. Når kantene er retningsbestemte viser grafen hvor man skal begynne og hvor man skal slutte. Det er derfor vanskelig å representere prosedural kunnskap i en graf uten retningsbestemte kanter, siden grafen da ikke vil kunne inneholde informasjon om når og i hvilken rekkefølge prosedyren skal utføres.

Roger Hartley brukte i sin artikkel en måte å representere prosedural kunnskap. Et eksempel på en slik graf vises i figur 7.3. Den inneholder også informasjon om baking, men er mye mer avansert og forklarer hvordan man skal bake i motsetning til den deklarative grafen som kun forklarer hva slaks konsepter som inngår og hvilke relasjoner de har til hverandre.[Hartley]



Figur 7.3 Graf som viser prosedural kunnskap om baking hentet fra [Hartley]

Grafer som representerer deklarativ kunnskap er enkle og oversiktlige og derfor lettere å tolke. Siden disse grafene ikke er så avanserte så holder det med grafer uten retningsbestemte kanter. Denne typen kunnskap kan for eksempel representeres i et konseptkart.

Grafer som representerer prosedural kunnskap er mer avanserte siden de inneholder mer informasjon, de trenger å inneholde informasjon om når og i hvilken rekkefølge prosedyren skal utføres. For å tegne slike grafer må man derfor bruke retningsbestemte kanter.
Skal man representere prosedural kunnskap er man nødt til å bruke mer nyanserte representasjonsmetoder som for eksempel en rettet graf.

7.1.3 Strukturell kunnskap

Det finnes en tredje type kunnskap, kalt strukturell kunnskap. Strukturell kunnskap forklarer hvorfor.

> ”It is not enough to know *that.* In order to know *how*, you must know *why.* Structural knowledge provides the conceptual bases for knowing why.” [Knee, Richard H.; Hafer, Rena]

Strukturell kunnskap kan bli representert i semantiske nett.

Strukturell kunnskap vil ikke bli vektlagt i min oppgave da denne kunnskapstypen ikke vil bli benyttet her.

7.2 Konsept

Konsepter blir brukt i mange forskjellige kunnskapsrepresentasjoner.
I min oppgave skal det framstilles en rekke kunnskapsobjekter(læringsobjekter) disse skal framstilles i et konseptkart ved at man representerer kunnskapen som deklarativ kunnskap. Konseptkart bruker konsepter som byggesteiner. Konsepter i min oppgave blir dokumenter(tradisjonelle dokumenter, læringsobjekter og kunnskapsobjekter), disse skal representeres visuelt i konseptkartet hvor man viser likheten mellom konseptene ved hjelp av koblinger.
Jeg vil i dette kapitlet prøve å forklare hva et konsept er eller kan være i en kunnskapsrepresentasjon.

Konsepter er de grunnlegende elementene for proposisjoner på samme måte som et ord er et grunnlegende semantisk element for en setning. Et konsept kan uttrykkes i flere forskjellige språk. For eksempel konseptet dog kan bli utrykt i engelsk som dog, i tysk som Hund eller som chien på fransk.
Konseptene kan på denne måten bli oversatt til et annet språk siden ord i forskjellig språk har den samme betydning. De vil da utrykke det samme konseptet[WIKIPEDIA]

Novak beskriver konsepter slik:

”We define concept as a perceived regularity in events or objects, or records of events or objects, designated by a label” [Novak, Joseph D]

Immanuel Kant definerte konsepter som en representasjon eller et mentalt blide av det som er felles for flere observerte objekter.[WIKIPEDIA]

Mennesker grupperer ting sammen i henhold til de egenskaper tingene har. Grupperinger som dette gis et enkelt ord. Ordet er navnet på konseptet.
For eksempel kan man si at alle objekter som har en horisontal flate der det er mulig å sette fra seg objekter og støttes av en eller flere bein kan samlet beskrives som konseptet bord.[McMillan, Bill]

Et konsept er noe som kan beskrives, det er et begrep. Konsepter er meningsbærende begreper som for eksempel ”en stol”, ”en datamaskin”, ”et sted” forståelsen av ”addisjon” og lignende begreper.
Et konsept er en abstrakt ide eller et mentalt symbol. Disse blir brukt til representasjon i språk som angir en spesifikk kategori eller klasser av entiteter, handlinger fenomener eller koblinger mellom dem.

Konsepter oppstår ved at man definerer dem i forhold til andre konsepter. Før man kan forstå konseptet baking må man forstå konseptene deig og bolle. Dette gjør at konsepter ikke har en absolutt definisjon. Man forstår konsepter gjennom relasjoner til andre konsepter. Et konsept vil derfor aldri være statisk. Konseptet forandrer mening når de konseptene man baserer seg på for å forklare konseptet forandrer mening. Dette gjør at konsepter er dynamisk.

Konsepter er også domenerelatert. Ser man på konseptet mygg så vil det beskrives på forskjellig måte av en biolog og av en tysk turrist på Hardangervidda. Et konsept kan ha ulik betydning i forhold til hvilket domene som behandler konseptet. En biolog kan for eksempel lage en biologisk modell, denne modellen kan ikke ha med seg turistens definisjon av mygg siden definisjonene til turisten ikke vil passe inn i domenet biologi.

Konsepter kan også bygges i hierarkier, et konsept kan inneholde flere andre konsepter. For eksempel konseptene bil og tog er en del av konseptet framkomstmiddel

Konsepter kan inneholde og representeres som deklarativ og prosedural kunnskap.
Som i eksempelgrafene kan konseptet baking bli beskrevet som prosedural kunnskap i figur 7.3 og som deklarativ kunnskap i figur 7.1

Konsepter er ekstremt viktig når man skal utvikle eller bruke vitenskap. Det vil være vanskelig å se for seg vitenskap uten konsepter som: energi, akselerasjon, tyngdekraft og lignende.
Konsepter hjelper å integrere urelaterte observasjoner og fenomener til gjennomførbare hypoteser og teorier.
Det å kunne finne sammenhenger mellom konsepter og lære nye konsepter er viktig når man skal tilegne seg ny kunnskap

Men disse konseptene er ikke vektlagt når de blir introdusert i skolen. Resultatet er at mange studenter blir forvirret. En rekke folk har forstått dette og har introdusert konseptkart som hjelper studenter til å lære forholdet mellom forskjellige konsepter.

7.3 Konseptkart

Konseptkartlegging er en teknikk for å visualisere forholdet mellom forskjellige konsepter.[WIKIPEDIA]

Et konseptkart er en grafisk representasjon hvor nodene representerer konsepter og linker representerer forholdet mellom konseptene.[Eric, Plotnick]

Konseptkart er et nettverk av konsepter og relasjoner som gir en meningsfylt fremstilling. Konseptkart kan brukes til organisering og strukturering av kunnskap, utvikle forståelse for sammenhenger, være grunnlag for diskusjoner, i gruppeprosesser kan det gi felles forståelse eller være en støtte for egne tankeprosesser.
[Hveem.pdf]

Konseptkart er en teknikk som blir brukt for å visualisere forskjellige konsepter og forbindelsen mellom dem.
Konseptkart ble utviklet på 1960 tallet av J. D Novak som en metode for å organisere og strukturere kunnskap. Bakgrunnen for dette var å vise og forstå endringer i barn sin kunnskap og forståelse. Konseptene ble plassert i tekstbokser og noder.
Meningen eller forståelsen av disse konseptene ble ikke synlig før man relaterte dem til andre noder(konsepter). Relasjonene ble knyttet opp med linjer med tekst. Disse kan framstilles med ulik farge tykkelser, lengder og vise retninger. (a påvirker b men b påvirker ikke a)

Konseptkart brukes for å representere kunnskap i en graf. En slik graf danner et nettverk av konsepter. Dette nettverket inneholder noder og kanter. Nodene representerer konsepter mens kantene representerer koblingene mellom konseptene i grafen. Koblingene kan ha forskjellig betydning, en sterkere farge kan for eksempel beskrive en større likhet til et annet konsept mens en stiplet linje kan ha en annen betydning en retningsbestemt pil. Konseptene har alltid et navn, kantene kan også ha navn men det er ikke noe krav om at de skal ha det. Kantene kan være retningsbestemt i en eller begge retninger. [Eric, Plotnick]

Det finnes mange forskjellige typer konseptkart. De mest vanlige er konseptkart, Mind map, mental mapping, concept mapping og semantisk nett.
Felles for disse er at de alle tilbyr en metode for å presentere kunnskap visuelt. Dette er en fordel siden visuell presentasjon gjør det lettere for hjernen å forstå meningen når kunnskapen presenteres i et visuelt format.

7.3.1 Formalitet

Konseptkart kan ha en helt uformell syntaks eller en streng syntaks som kan omskrives til et språk. Jo mer formelt et konseptkart er desto mer opplæring trengs for å kunne bruke det effektivt. Et problem er at brukere ofte forholder seg til formelle kart på samme måte som de forholder seg til uformelle kart. Dette gjør at brukeren ofte genererer en representasjon som har mening for dem, men ikke gir mening for andre eller datasystemet som skal tolke representasjonen. (Gaines, Brian R; Shaw, Milfred)

Med en for streng syntaks kan man få konseptkart som kun kan benyttes av eksperter. Det er derfor viktig å unngå dette. Man bør heler benytte er visuelt språk som er lettforståelig og formelt gir muligheter til fattbarhet og redigering.

Med en formell syntaks går det an å avlede kunnskap fra grafen på en formell måte, dette gjør at datamaskiner kan tolke grafen.

7.3.2 Pedagogikk

Konseptkart er et godt verktøy når man skal se eller forstå eksisterende kunnskap i sammenheng med ny kunnskap. Meningsfull læring baserer seg på at nye konsepter settes i sammenheng med den eksisterende forståelse brukeren har for de andre konseptene i konseptkartet og som brukes for å forklare/beskrive det nye konseptet. Visuelt gir konseptkart en god forståelse. Man kan med konseptkart fremstille kompliserte ideer på en slik måte at de lettere kan forståes av andre. En god visuell framstilling er ofte lettere å fatte en flere hundre ord i en bok som forklarer det samme problemet.

7.3.3 Manuelt generert konseptkart

Ved manuelle konseptkart bygger brukerens selv opp konseptkartet på lignende måte som man gjorde på skolen når man lagde såkalte tankekart før man begynte å skrive en stil.
Tony Buzan utviklet denne notatteknikken med stikkord, figurer symboler og ikoner.
Tankekart skal være raskt å lage og enkelt å huske. Tankekart kan benyttes for å sikre resultater fra kreative prosesser.
Slike manuelt genererte konseptkart gir brukeren evne til å søke etter og finne gode koblinger mellom konseptene. En slik egenskap stimulerer brukerens kreative tankegang og hjelper den lærende til enklere å se sammenhenger.

En datamaskin har flere fordeler ved generering av konseptkart. Disse egenskapene gjelder for manuelt genererte og automatisk genererte konseptkart og ble foreslått av [Eric, Plotnick].

Enkel tilpassing og manipulasjon: Plassering av et nytt konsept i et eksisterende konseptkart er vanskelig når du har skrevet det ned på papir. Et dataverktøy kan enkelt fjerne denne vanskeligheten. Novak Joseph D sier at ”et konseptkart blir aldri ferdig”

Dynamiske koblinger: De fleste dataverktøy for manuelle konseptkart gir brukeren mulighet til å flytte på konsepter, koblinger eller grupper for å gi konseptkartet et nytt utseende. På samme måte kan vi si at de fleste automatiske genererte konseptkart også er dynamiske siden de forandrer seg når konsepter og koblinger forandrer mening.

Kommunikasjon: Fordelene med et elektronisk konseptkart består blant annet i at det kan gjøres lett tilgjengelig for andre.

Lagring: Digital lagring på datamaskiner tar mye mindre plass i forhold til å bruke tavler og ark. Søking blir enklere og raskere ved hjelp av datamaskiner og dette gjør at man kan bruke konseptkart i en større skala.

7.3.4 Automatisk genererte konseptkart

I min oppgave blir konseptkartet generert automatisk. Stegene som ble forklart i kapittel om informasjonsgjenfinning blir brukt til å generere en likhetsmatrise over konseptene som skal vises i konseptkartet. Programmet tar i bruk denne likhetsmatrisen og plasserer konseptene i et 3d rom i forhold til likhet mellom konseptene. Brukeren av systemet kan indirekte forandre innholdet i konseptene (så lenge de ikke er statiske filmer, lydfiler eller internett koblinger).

Forskjellen på manuelle og automatisk genererte konseptkart er ikke nødvendigvis så stor. Men den største forskjellen er at de manuelle konseptkartene legger pedagogikken i plassering av konsepter og koblinger, mens de automatisk genererte konseptkartene vektlegger tolkning og forandring av konseptkartet ved å forandre på grunnlagskonseptene.

De manuelt genererte konseptkartene inspirerer til kreativ tankegang ved at brukeren selv plasserer konsepter og koblinger. De automatiske genererte konseptkartene har som oppgave å vise sammenhenger som allerede eksisterer mellom konseptene, og gi brukeren mer tid til å reflektere over denne strukturen. Brukeren kan indirekte forandre på strukturen ved å forandre på konseptbeskrivelsen som danner grunnlaget for strukturen.

7.3.5 Eksempler på forskjellige kart

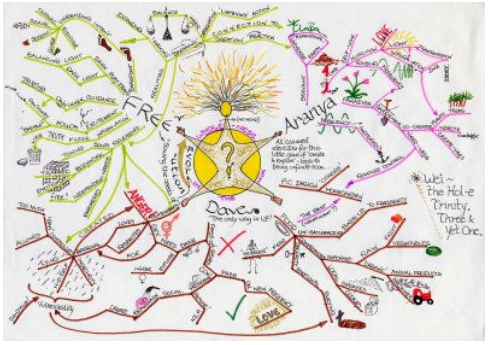
7.3.5.1 Mind map

Mind maps er et diagram som brukes for å linke ord og ideer til et sentralt nøkkelord eller ide. Det brukes til å visualisere, klassifisere, strukturer og generere ideer. De blir hyppig brukt av studenter innenfor problemløsning og for å enklere kunne ta beslutninger.

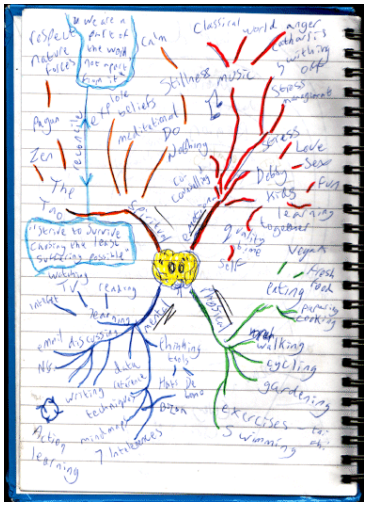
Dette har vært bruk i lang tid til brainstorming, minne og visuell tenking og problemløsing av lærere, ingeniører og psykologer. Første eksemplet på et mind map ble utviklet av Porphyry of Tyros, han brukte mind map til å grafisk visualiserer konsept kategoriene til Aristoles (Filosof og lærer studerte alt som kunne studeres på sin tid. Blant annet lærer til Alexander den store og elev av Plato).

Mind maps brukes i dag som oftest til personlige, familie, læring og business situasjoner.

Mest brukt blir mind maps brukt som et notat teknikk modifisert til en type brainstorming. For eksempel kan man ta notater fra en forelesning ved å bruke mind map for de viktigste poengene og nøkkelordene.



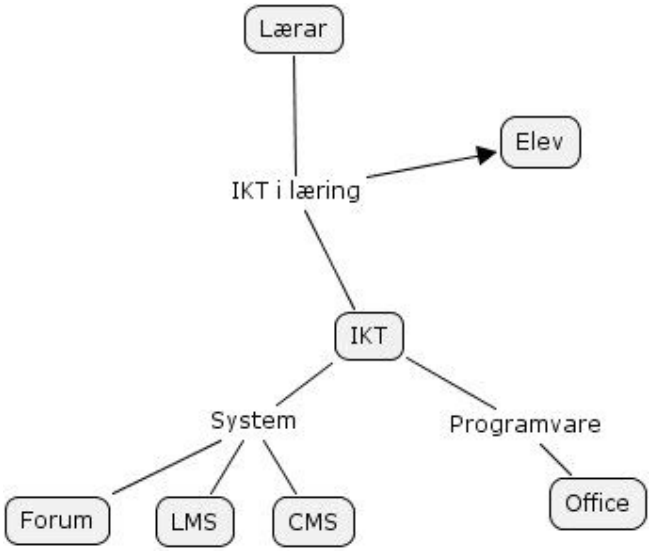
Figur 7.4 Mind map etter en forelesing hentet fra
http://en.wikipedia.org/wiki/Mind_mapping



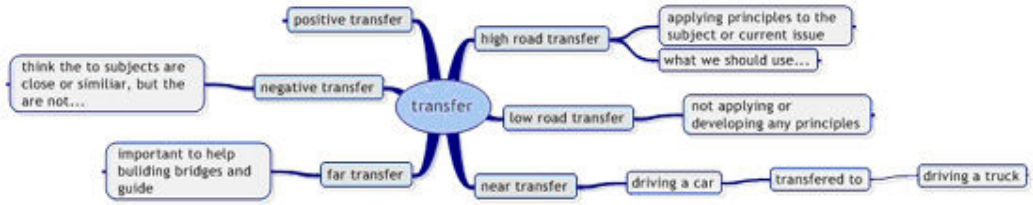
Figur 7.5: Hånd tegnet mind map hentet fra
http://en.wikipedia.org/wiki/Mind_mapping

7.3.5.2 Tankekart

Utviklet av Tony Buzan på 1960 tallet, en grafisk notatteknikk med stikkord, figurer, symboler og ikoner. Tankekart skal være raskt å lage og enkelt å huske. Tankekart brukes for å sikre resultat fra kreative prosesser. De kan brukes til å lage strukturerte representasjoner av resultat fra en ustrukturert prosess. Visualiseringer gir ofte en bedre oversikt enn tilsvarende lineære tekster.



Figur 7.6: Tankekart laget I CMap hentet fra [Hveem.pdf]



Figur 7.7: Tankekart laget i Mindmanger fra [Hveem.pdf]

7.3.5.3 Konseptkart

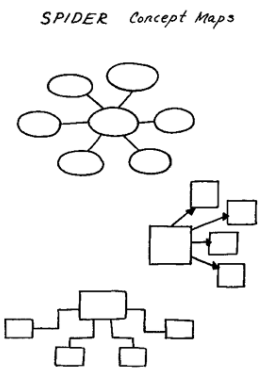
Det er fire hovedkategorier av konseptkart. Disse kan bli skilt fra hverandre siden de har forskjellig format for å representere informasjon. [UIUC]

Edderkoppkonseptkart er organisert på en slik måte at man plasserer det sentrale temaet i midten av karter. Undertemaene blir plassert rundt de sentrale temaene. Se figur 7.8.

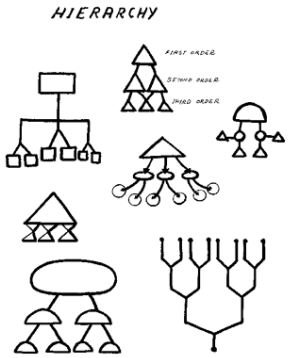
Det hierarkiske konseptkartet presenterer informasjon i en nedstigende rekkefølge i forhold til viktighet. Den viktigste informasjonen blir plassert på toppen. Se figur 7.9.

Flytkartkonsept kartet organiserer informasjon i et lineært format. Se figur 7.10.

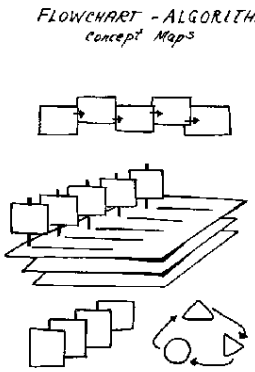
Systemkonseptkart organiserer informasjon i er format som er lik et flytkart men har også inputs og outputs. Se figur 7.11.



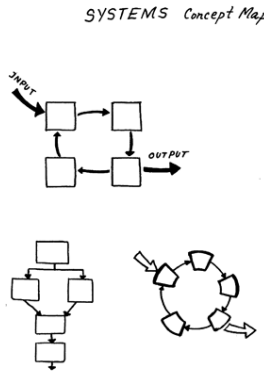
Figur 7.8 Edderkopp konsept kart



Figur 7.9 Hierarkisk konseptkart



Figur 7.10 Flytdiagram

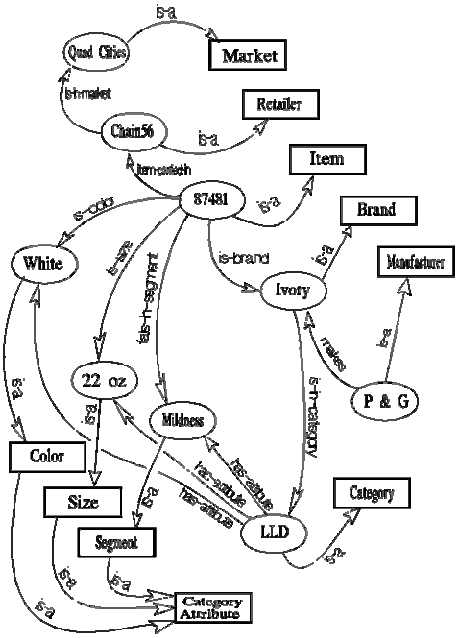


Figur 7.11 System konseptkart

7.3.6 Semantiske nett

Semantisk nett er et formelt konseptkart som kan inneholde mer enn deklarativ kunnskap.
Et semantisk nett er en formell kunnskapsrepresentasjon som har metoder og egenskaper som gjør det mulig å avlede kunnskap fra nettet på en formell måte.
Semantiske nett er konseptkart med formell syntaks. I sin enkleste form så kan et slik nett være i form av en graf med noder og kanter som utrykker en semantikk (en mening). Man kan si at det semantiske nettet er en symbolsk kunnskapsrepresentasjon. Semantiske nett har inferensmetoder som gjør at man kan trekke kunnskap ut av nettet. Dette framstilles ved symboler som er kjente symboler slik at et menneske kan forstå meningen i nettet og kan utlede det samme ut av det som en datamaskin.
Semantiske nett blir ofte sett på som nett med konsepter representert som noder, og sammenligningen mellom konseptene blir representert som retningsbestemte titulerte kanter. For eksempel konseptene dyr og hund kan ha en kant som forklarer at en hund er et dyr.[UIO, grafer]
Vanligvis er et semantisk nett en rettet asyklisk graf. En slik graf inneholder ingen looper/løkker i grafen og blir kalt DAG Directed Acyclic graph. Når grafen er på et slikt format uten loop i grafen så er det mer effektivt å søke i den. En slik graf fjerner muligheten for evige løkker i programmering.

Semantiske nett er en spesialisert type av konseptkart. Forskjellen på dem er at semantisk nett brukes til å representere kunnskap formelt. Semantiske nett blir vanligvis representert i en DAG mens et generelt konseptkart ikke behøver å ha retningsbestemte kanter og tar som oftest form som en generell graf.
Semantiske nett defineres i en formell kunnskapsrepresentasjon som inneholder kunnskap og metoder for å arbeide med kunnskapen. Konseptkartene trenger ikke å skrives om til et formelt språk, men dette er en egenskap som semantiske nett har.



Figur 7.12: Eksempel graf. Semantisk nett.

Nettverket i denne figuren representerer informasjon om node 87481. Det er en gjenstand. Det er en form for elfenben som blir laget av Procter og Gamle, hvit farge osv.

8 Beskrivelse av prototyp

Jeg vil i dette kapitlet først forklare hva som skal visualiseres i mitt program, jeg vil også komme innom viktige egenskaper en slik visualisering bør ha med. Jeg vil i resten av kapitlet beskrive prototypen jeg har implementert.

8.1 Visualisering av en kunnskapsbase

Jeg ønsker i min oppgave å konstruere en tredimensjonal kunnskapshimmel på bakgrunn av en kunnskapsbase. En kunnskapsbase kan her for eksempel være den kunnskapen brukeren har tilegnet seg og kan også kalles en personlig kunnskapsbase.

Denne kunnskapsbasen vil bli visualisert som et konseptkart (kapittel 7.3).

Den lærende er innom en stor rekke med forskjellige ressurser når han skal lære ny kunnskap. Dette kan være dokumenter, lydfiler og interaktive modeller for å nevne noen.
Når selve læringsprosessen er ferdig sitter den lærende igjen med en del skriftlig materiale som ofte aldri blir tilegnet, men mindre det blir organisert på en oversiktlig og enkle måte. Kunnskap vil også forsvinne over tid hvis den ikke blir oppfrisket. Det er også svært vanskelig for andre å få tak i den informasjonen den lærende har tilegnet seg hvis den ikke er organisert og framstilt på en enkel og oversiktlig måte.

Prototypens hovedoppgave er å ta seg av organiseringen av kunnskapen, slik at den kan oppfriskes, og presenteres for den lærende og andre på en oversiktlig og enkel måte.

Det er også et hovedpoeng at den lærende skal kunne få mulighet til å få ny innsikt og forståelse av hvordan disse kunnskapsbitene henger sammen ved å organisere dem som et konseptkart.

8.1.1 Viktige egenskaper til slike visualiseringer av kunnskapsbaser

Jeg vil her beskrive en rekke egenskaper en slik visualisering av en kunnskapsbase bør ha med.

Konsepter

Det som skal representeres i kunnskapshimmelen er konsepter.
I min oppgave så vil et konsept kunne være alt fra et dokument til en avansert interaktiv simulering. Konseptene vil peke til ressursenes url-er.

Visualisering

Det er viktig å kunne visualisere kunnskapen for brukeren. Det er viktig å få en oversiktlig, lettfattelig visualisering av kunnskapsbasen presentert på skjerm. Dette gir gode muligheter for å søke etter kunnskap og gir brukeren en lettere måte å se sammenhengen i kunnskapen.

Automatisk generering

I den tredimensjonale kunnskapshimmelen vil konseptenes plassering og relasjonene mellom konseptene genereres automatisk.

En manuelt generert kunnskapshimmel ville ha gitt brukeren mulighet til å sette opp konseptene og koblingene selv. Målet i min oppgave er å generere dette automatisk.

Ved å benytte automatisk generering vil man få fram sammenhengen i forhold til den tekstlige beskrivelsen av konseptene.
En automatisk generering av kunnskapsbasen gir mulighet for å tolke nye sammenhenger ut av kunnskapsbasen umiddelbart.

En automatisk generering vil også forenkle arbeidsprosessen. Hvis brukeren må danne visualiseringen av kunnskapsbasen manuelt, så vil han lett miste fokuset på definisjonene av konseptene siden det vil kreve mye arbeid med å plassere konseptene og sette opp relasjoner mellom dem.

Tredimensjonalitet

I min oppgave vil det dannes en tredimensjonal visualisering av kunnskapsbasen. Disse grafene oppfattes som mer behagelig å se på, tredimensjonale grafer gir også bedre mulighet for visualisering enn to dimensjonale grafer.

Selv om man har store muligheter til å få til bedre visualisering i 3d grafer må man passe på at dimensjonene brukes med forsiktighet. Man må passe på at de tre dimensjonene brukes til å forenkle presentasjonen, de bør ikke brukes for å gjøre visualiseringen mer komplisert.

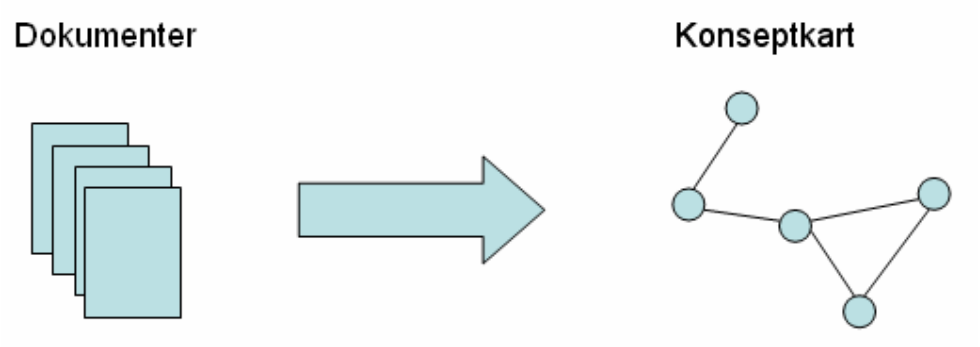
Synsvinkler

Ved å legge inn flere synsvinkler blir det lettere for brukeren å finne fram i modellen enn om den var helt statisk.

8.2 Beskrivelse av Prototyp

For å få visualisert et tredimensjonalt konseptkart så trenger man data for å kunne visualisere.

I min oppgave skal det dannes en tredimensjonal kunnskapshimmel. For å kunne visualisere den trengs det en rekke genererte data.
Disse dataene trekkes ut fra et sett med dokumenter. Disse dokumentene er konsepter som tilhører et kunnskapsdomene(kunnskapsbase). Hovedpoenget er å kunne presentere disse i et tredimensjonalt konseptkart.

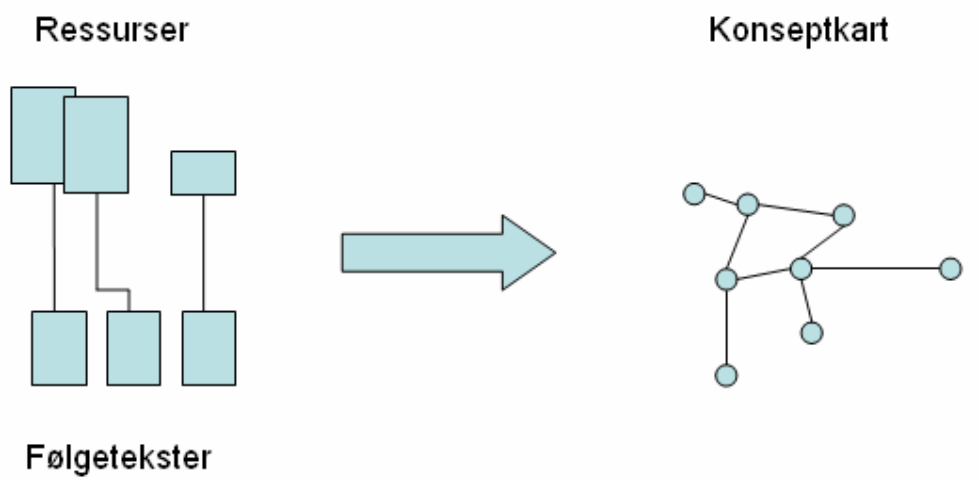


Figur 8.1: Dokumenter representeres som konsepter i konseptkartet

Ikke bare dokumenter som kan visualiseres

Som sagt i kapitlet om informasjonsvisualisering så vil man kunne skape vektorer ut fra teksten til dokumenter. Legger man til en følgetekst til ressurser som egentlig ikke er tekstbasert så kan man benytte denne følgeteksten som beskriver ressursen til å lage vektorer. Man kan da visualisere alt fra interaktive simuleringer til enkle dokumenter. Det kreves selvfølgelig her en del arbeid med å lage gode følgetekster. Disse tekstene må beskrive ressursene på en slik måte at de egner seg til å bli vektet opp mot dokumenter og andre ressurser som man ønsker å presentere.

Man benytter følgeteksten til å vekte og beregne data som trengs for å visualisere disse ressursene. Disse ressursene vil bli framstilt i kunnskapshimmelen som konsepter figur 8.2 illustrerer dette.



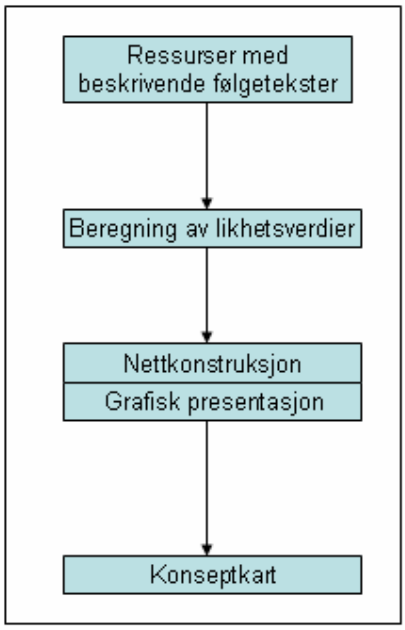
Figur 8.2: Man bruker følgetekster til å generere vektorer som brukes til å beregne data for å visualisere ressursene i konseptkartet.

8.2.1 Opprettelse av konseptkart

En sekvensiell framstilling er den mest vanlige måten å presentere tekst på. Ser man for eksempel på min besvarelse så er den sekvensielt delt opp i kapitler og underkapitler.

I min prototyp vil jeg gå vekk fra den tradisjonelle sekvensielle framstillingen å presentere kunnskap på. I stedet for å presentere kunnskapen i en sekvensiell struktur så vil jeg presentere den i form av et konseptkart.

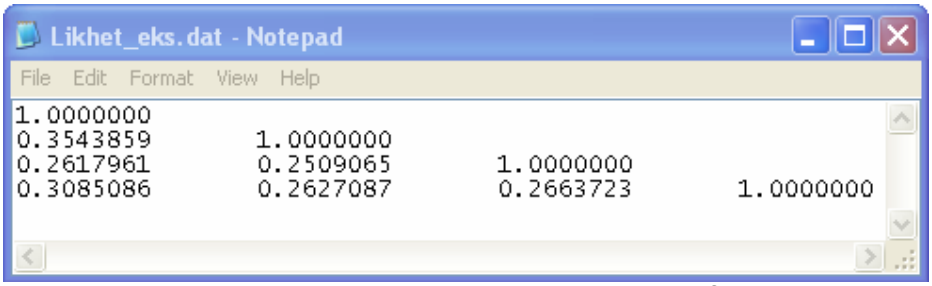
Hovedproblemet ligger i å konvertere et sett dokumenter eller ressurser det er knyttet en beskrivende følgefil til og omforme dette til et konseptkart. Denne prosessen illustreres i figur 8.3.



Figur 8.3: Fra følgefiler til konseptkart

Det hele starter med følgefilene som representerer ressursene. Ressursene representerer konseptene i konseptkartet. Ut fra disse konseptene må det dannes likhetsmål mellom alle par i samlingen av konsepter. Dette likhetsmålet trengs for å kunne plassere konseptene i forhold til hverandre i konseptkartet. Dette likhetsmålet bestemmer også om det skal opprettes koblinger mellom konseptene eller ikke. Disse likhetsverdiene blir dannet av et IR-system som er utviklet av Per Kristen Fredlund i sin masteroppgave: *Building a Knowlegebase from Learning objects 2005.*

I min prototyp blir disse likhetsverdiene framstilt i en tekst fil. Illustrert i figur 8.4.



Figur 8.4: Likhetsverdier presentert i tekst fil

Når alle likhetsmålene er tilgjengelig mellom alle par av konsepter så kan man danne et nett. Dette nettet vil være en representasjon av konseptkartet. Jeg har i kapittelet om visualisering

kommet innom en rekke egenskaper en slik visualisering burde ha med. Jeg vil komme nærmer inn på hvilke av disse egenskapene jeg har implementert og vil forklare hvordan dette er løst.

I prototypen har jeg benyttet Java og Java 3D for å lage den grafiske presentasjonen.
Når prototypen har gått gjennom alle fasene vil brukeren sitte igjen med en tre dimensjonal kunnskapshimmel representert som et konseptkart.

8.2.2 Problemstillinger som må løses

Jeg har ikke laget likhetsmatrisen selv denne er blitt generert av IR-systemet til Per Kristen Fredlund.
I dette systemet er det teksten til dokumentene som blir brukt til å danne likhetsmål.
Jeg har beskrevet hvordan man kan bruke vektorer til å beregne likhetsmål i kapittelet om kunnskapsrepresentasjon. Noe forenklet kan man si at systemet danner vektorer fra dokumenter. Disse vektorene blir sammenlignet og det blir dannet likhetsverdier mellom hvert par av dokumenter. Disse likhetsverdiene danner en likhetsmatrise der det er knyttet en likhetsverdi til hvert av dokumentene.
Problemet videre for min prototyp er å konvertere denne likhetsmatrisen om til et konseptkart. Dette krever at en del problemstillinger blir løst. Blant problemstillingene som må løses er:

Fra *n* til tre dimensjoner

Vektorene representerer n-dimensjoner. På en eller annen måte må disse n-dimensjonene bli omformet til en tredimensjonal presentasjon som er forståelig for oss.

En graf med n-dimensjoner må transformeres til en graf i tre dimensjoner samtidig som den må inneholde den samme informasjonen som den originale grafen.

Dette gjøres ved at man beregner likhetsmål mellom hvert av konseptene (hvert konsept er gjort om til en vektor) som skal visualiseres. For å få fram denne graden av likhet så må disse likhetsmålene transformeres om til avstand. Jo mer likt et konsept er et annet konsept jo nærmere vil disse konseptene være hverandre i konseptkartet. Problemet ligger i å finne en måte å plassere disse konseptene slik at alle har riktig avstand i forhold til hverandre ut fra likhetsmålet. Den ideelle måten å gjøre dette på er å bruke en objektiv representasjon. I følge [Simon, Steinbrückner, Lewerentz] så krever en objektiv representasjon maksimalt antall noder minus en dimensjon for å kunne tegnes opp korrekt. Har man for eksempel 15 noder så trenger man opp til 14 dimensjoner for å klare å tegne den objektive representasjonen korrekt.

Man kan presentere flere enn 3 dimensjoner matematisk men øyet vil aldri klare å oppfatte det på fornuftig vis. Prototypen er derfor helt nødt til å redusere antall dimensjoner ned til 3 slik at de kan presenteres i et tredimensjonalt konseptkart.
Ved å benytte likhetsmålene som er beregnet med en plasseringsalgoritme som spring embedder er det mulig å presentere konseptkartet i 3 dimensjoner.

Automatisk plassering på skjerm

Kunnskapsbasen som skal representeres som et konseptkart skal skje automatisk. Det å plassere noder automatisk på skjerm for å vise likheten mellom dem er ikke en lett oppgave når man kun har et likhetsmål som utgangspunkt.

Fornuftig plassering og lettfattelig

Nodene skal plasseres automatisk på skjerm, de må også plasseres på en fornuftig måte slik at konseptkartet blir lettfattelig.

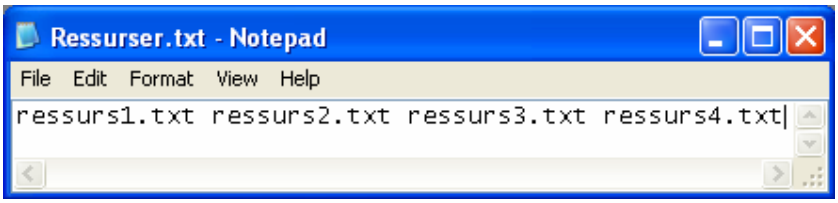
8.2.3 Beskrivelse av programflyt

Basert på en fil over url-er til ressurser og en likhetsmatrisefil så utfører min prototyp alle beregninger som trengs for å visualisere konseptene i et tredimensjonalt konseptkart. Konseptene blir representert som kuler i et konseptkart. Brukeren kan høyreklikke på kulene som representerer konseptene i kartet og åpne ressursene knyttet til kulene. Åpning av disse ressursene startes som en egen prosess som kjøres ved siden av Java 3d applikasjonen.
For å spre ressursene i konseptkartet brukes det en spredningsalgoritme som sprer ressursene basert på likhetsmål.

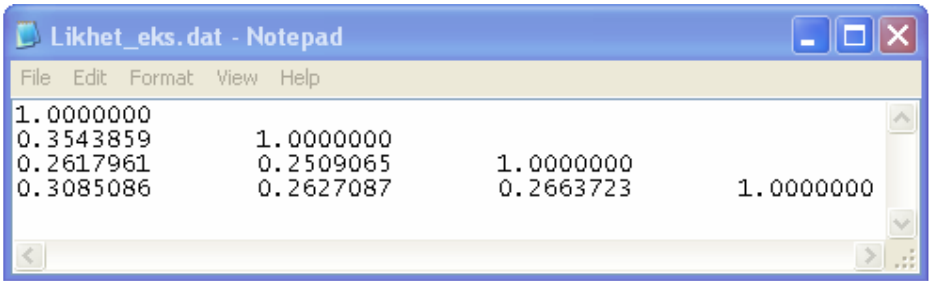
8.2.4 Fra inputdata til kunnskapshimmel

Jeg vil her forklare noe forenklet hvordan prototypen min klarer å konvertere inputdata til et konseptkart.
Inputdata til systemet er som sagt to filer. Det er en fil som inneholder url-er til ressurser og en fil som inneholder beregnet likhetsmål til disse ressursene.

Formatet på disse filene illustreres nedenfor:



Figur 8.5: List over url-er til ressurser som skal framstilles i konseptkartet



Figur 8.6: ½ likhetsmatrise som inneholder likhetsmål mellom alle ressursene

Disse filene leses inn av programmet. Likhetsmatrisen lagres i programmet for lettere å kunne hente ut verdier fra matrisen. For hver av ressursene så dannes det en egen node.
Til disse nodene lagres url adressen til ressursen.
I disse nodene så genereres det også tre tilfeldige koordinater(x, y og z), dette gjøres siden det er første steget i plasseringsalgoritmen som benyttes.

Når innlesningen av alle data er utført så vil programmet utføre plasseringsalgoritmen. Den beregner beste plassering ut fra likhetsmålene til hver av nodene og oppdaterer x, y og z koordinatene i nodene.

Når selve innlesingen av data er utført så starter prototypen å opprette 3d objektene som skal visualiseres. Den oppretter kuler som representerer ressursene som skal visualiseres. Disse kulene vil bli koblet til datanodene som nå inneholder url-en til ressursen og de beregnede x, y, og z verdiene som springembedder algoritmen har gitt noden. Det opprettes også kanter som representerer koblinger mellom kulene. Om det skal tegnes opp en kobling eller ikke mellom to kuler bestemmes av en terskelverdi

Terskelverdi

Alle konseptene har fått et likhetsmål beregnet fra IR-systemet. Alle konseptene har en kobling til hverandre. Disse likhetsmålene har en verdi mellom 0 og 1. Vi har da en urettet sammenhengende graf (se kapittel 6.4.1 forskjellige typer grafer) (en graf der alle nodene er koblet direkte med alle andre noder i graf).

Hvis man visualiserer alle koblingene mellom alle nodene blir det uoversiktlig og meningsløst. Det som er interessant er å vise koblinger mellom de konseptene som har en høy nok grad av likhet.

For å styre at ikke alle konseptene som tegnes opp har knyttet kant til alle de andre konseptene så har jeg opprettet en terskelverdi. Denne terskelverdien bestemmer om det skal tegnes opp en kobling mellom to konsepter eller ikke. Er likhetsmålet mellom to noder over denne terskelverdien så vil det bli tegnet opp kobling, er den mindre enn terskelverdien så tegnes det ikke opp kobling mellom nodene.

Denne terskelverdien bestemmer hvor mange koblinger som presenteres i konseptkartet og er en verdi som brukeren kan velge før 3d visualiseringen starter.

8.2.5 Navigering og visualisering implementert i prototypen

I kapittel 6 visualisering, kom jeg innom en rekke aspekter som er viktig å tenke over før man implementerer et visualiseringssystem. Jeg beskrev også en rekke metoder som egner seg til å løse eventuelle desorienteringsproblemer. Jeg vil i dette kapitlet beskrive hvordan jeg har løst disse aspektene og hvilke metoder jeg har lagt til for å løse desorienteringsproblemer som kan oppstå.

8.2.5.1 Aspektene

Forutsigbarhet

Som nevnt i kapittel 6 så er forutsigbarhet viktig for brukeren. Det er viktig at informasjon og tilnærmet lik informasjon blir presentert likt. Hvis grafen er lik hver gang man starter programmet så vil brukeren lett danne seg et mentalt bilde. Det vil da være enklere for brukeren å navigere i grafen å finne igjen informasjon som han har besøkt før.
En presentasjon som ikke er forutsigbar kan føre til forvirring og uoversiktlighet. Jeg benytter i mitt program en plasseringsalgoritme. Denne algoritmens har som utgangspunkt at alle

nodene som skal visualiseres først får en tilfeldig generert plassering før algoritmen begynner å beregne optimal plassering. Dette fører til at grafene aldri blir helt like fra gang til gang man kjører algoritmen.

I grafer med få noder så ser grafene relativt like ut, men når man introduserer mange nok noder så klarer ikke spring embedderen å spre nodene likt fra gang til gang (visuelt sett). Problemet består i at nodenes plassering fra start har innvirkning på hvordan nodene blir dyttet og tiltrukket hverandre.

Spring embeddermetoden starter ved å danne en tilfeldig plassert graf, som justeres med Newton-Raphsonmetoden.
Springembedderen ønsker hele tiden å finne en konfigurasjon med minst mulig energi ved å hele tiden finne en ny graf som kan bytte ut den eksisterende grafen hvis den nye grafen har mindre energi i seg. Antall ganger man velger å bytte ut grafen med en forbedret graf styres av en iterasjonsvariabel i prototypen. Selv om man benytter denne metoden så vil ikke grafene se identiske ut fra gang til gang.

Jeg har derfor i min prototyp løst problemet ved å lagre plasseringsdata til fil. Første gangen programmet starter så eksisterer det ikke noen data på fil over hvor nodene skal plasseres i den tredimensjonale kosneptkartet. Plasseringsalgoritmen blir da kjørt og plasseringsdata blir lagret til fil. Neste gang man kjører programmet så eksiterer dataene på fil slik at man vil få visualisert det samme konseptkartet.
Siden plasseringsalgoritmen kun blir kjørt første gangen man starter programmet så har jeg valgt å øke antall ganger man forsøker å bytte ut den eksisterende grafen mot en forbedret graf med mindre energi. Jo mindre energi desto mer riktig blir konseptkartet tegnet opp. Dette gjør at prototypen bruker lenger tid ved første kjøring, men grafen blir mer korrekt. De neste kjøringene blir mye raskere siden man nå kun leser inn de plasseringsdataene som ble lagret til fil fra første kjøring.

Tidskompleksitet og størrelse

Det er som sagt viktig at brukeren sitter igjen med en interaktiv følelse av bruk av systemet. Størrelse på grafen og antall elementer som skal tegnes opp avgjør om den interaktive følelsen er god nok. Både hastighet i opptegning og oversiktlighet er viktig her. Jeg vil komme innom dette i kapittelet om testkjøring.

8.2.5.2 Teknikker og metoder implementert for å løse desorienteringsproblemer

Navigasjon

Orientering i 3d-grafen

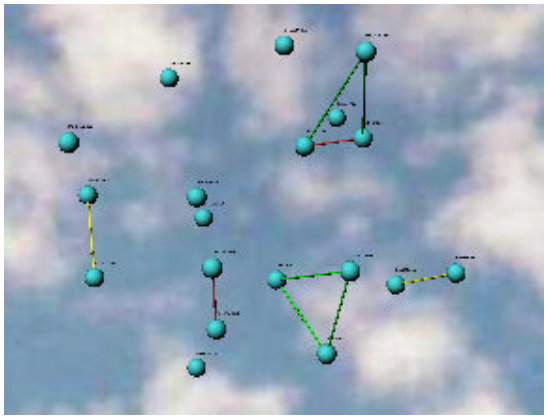
For at brukeren skal finne et konsept og enkelt kunne se sammenheng mellom konseptene så er det lagt til forskjellige orienteringer i grafen. Det er mulig for brukeren å:

- Rotere grafen rundt origo ved å holde nede venstre musknapp og bevege musa.
- Flytte grafen nord, sør, øst eller vest ved å holde nede høyre musknapp.
- Zoome inn og ut på grafen ved å bruke midtre musknapp.

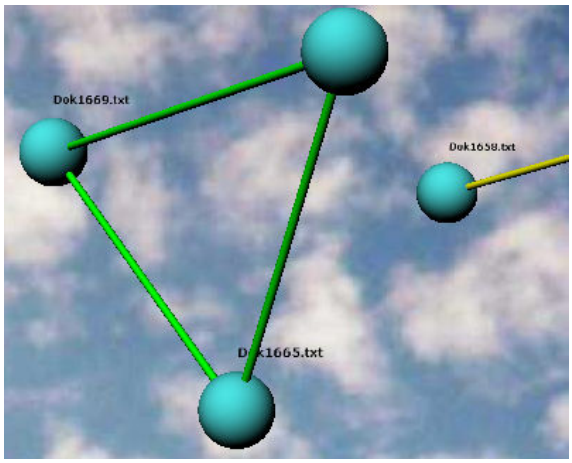
Med disse orienteringsmulighetene er det mulig å manøvrere konseptkartet fysisk.

Det gir brukeren mulighet til å både zoome inn og ut, flytte grafen sidelengs og rotere konseptkartet.

Denne egenskapen er viktig for å gi brukeren mulighet til å se konseptkart fra forskjellige synsvinkler.



Figur 8.7: Skjermbilde før zooming



Figur 8.8: Fokuseringsteknikk zoom. Det er her zoomet inn på noder av interesse.

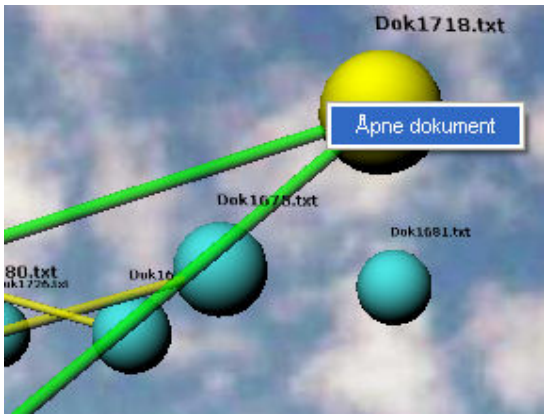
Nodenavn

Som sagt i kapittel 6 *visulisering* så er det viktig å vise navnet på noden. Dette gjøres i prototypen ved å vise en merkelapp som roterer sammen med grafen. Nodenavnet som brukes i prototypen er navnet på ressursen som er knyttet til kulen. Dette gjøres for at brukeren skal lettere forstå hva som ligger i noden som brukeren velger å åpne.

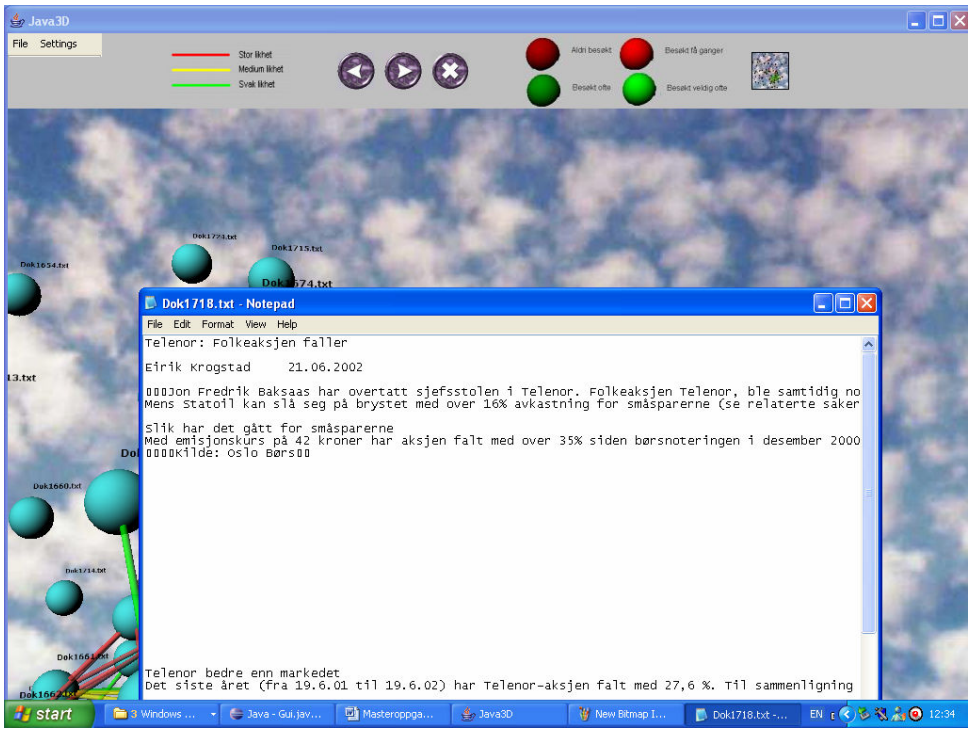
Valg av kuler

I prototypen skal det være mulig å høyreklikke på en kule. Hver kule representerer et konsept, dette konseptet inneholder en url til den ressursen som representerer konseptet. Dette kan for eksempel være et dokument, link til en internettside eller en video snutt.

Når en bruker høyreklikker på en kule i programmet så vil url-en til ressursen som representerer konseptet bli registrert. Det opprettes en popup meny der brukeren får et menyvalg som gjør det mulig å starte url-en fra Java 3D som en egen prosses.



Figur 8.9: skjermbilde der brukeren har høyreklikket på en node.



Figur 8.10: Skjermbilde der programmet har startet ressursen som en egen prosess

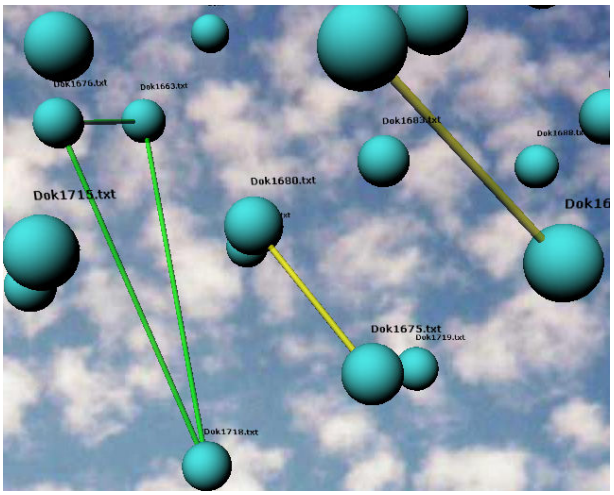
Skifting av farge på noder

Når en bruker velger en node og åpner en ressurs så forandrer fargen på noden seg. Dette gjøres for at brukeren skal kunne se hvilken node som er aktiv. Velger brukeren en annen node så vil den ”gamle” noden få tilbake originalfargen og den nye noden vil få fargen som indikerer hvilken node som er valgt. Denne teknikken hjelper brukeren til å huske hvilke node som akkurat har blitt åpnet.

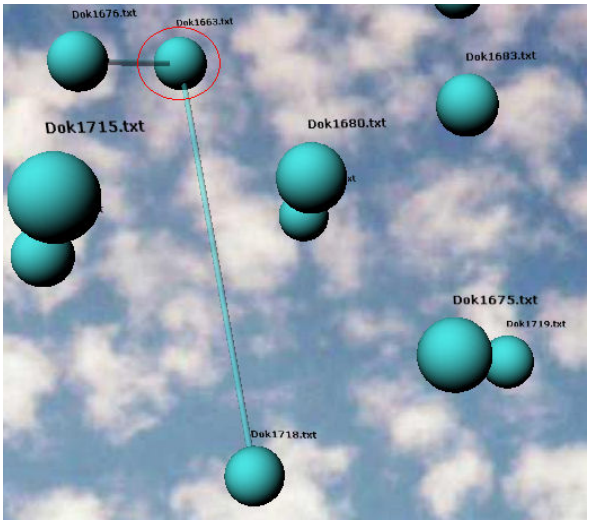
Koblingsmanipulasjon

For at brukeren skal lettere se hvilke noder som er knyttet til hverandre så er det opprettet en rekke kanter til konseptene som har likhet over en viss terskelverdi. I et stort konseptkart med mange koblinger blir det fort uoversiktlig med for mange koblinger.

For å gjøre det lettere for brukeren å fokusere uten å miste oversikten har jeg implementert en fokuseringsteknikk som jeg har valgt å kalle koblingsmanipulasjon.
Hvis brukeren venstreklikker på en node i konseptkartet så vises kun koblingene fra denne noden i konseptkartet, alle de andre nodene blir skjult for brukeren.
Jeg har her lagt til to skjermbilder som illustrer dette.



Figur 8.11: Når brukeren ikke fokuserer på et konsept i konseptkartet så vises alle koblingene.



Figur 8.12: Når en bruker velger å fokusere på et konsept(venstre klikker på en kule) får man kun se de koblingene som går fra dette konseptet. Alle andre koblinger fjernes.

Denne manipulasjonen gjør det lettere for brukeren å finne de konseptene som er mest likt det valgte konseptet. Det gjør grafen mer oversiktlig ved at man kan filtrere vekk unødvendig informasjon slik som uinteressante koblinger mellom andre konsepter.

Fokuseringsteknikken bevarer også konteksten siden. Konteksten er identisk lik med slik den var før man valgte å fokusere.

Farger på koblinger

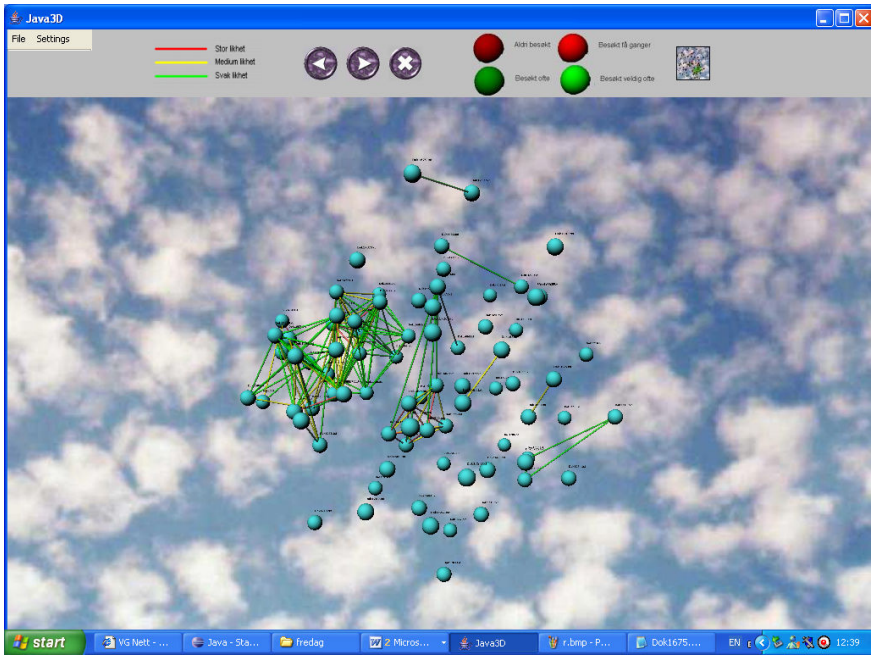
For å vise brukeren forskjellen i koblingene visuelt så har jeg valgt å ha forskjellig farge på koblingene. Koblingens farger blir styrt av likhetsverdien mellom nodene koblingene er tegnet mellom. Jeg har prøvd forskjellige farger på koblingene men satt igjen med fargene rødt, gult og grønn til slutt. Rødt indikerer at det er stor likhet mellom to noder, en gul farge indikerer en medium likhet mellom nodene, mens en grønn farge indikerer svak likhet mellom nodene.

Merking av besøkte noder

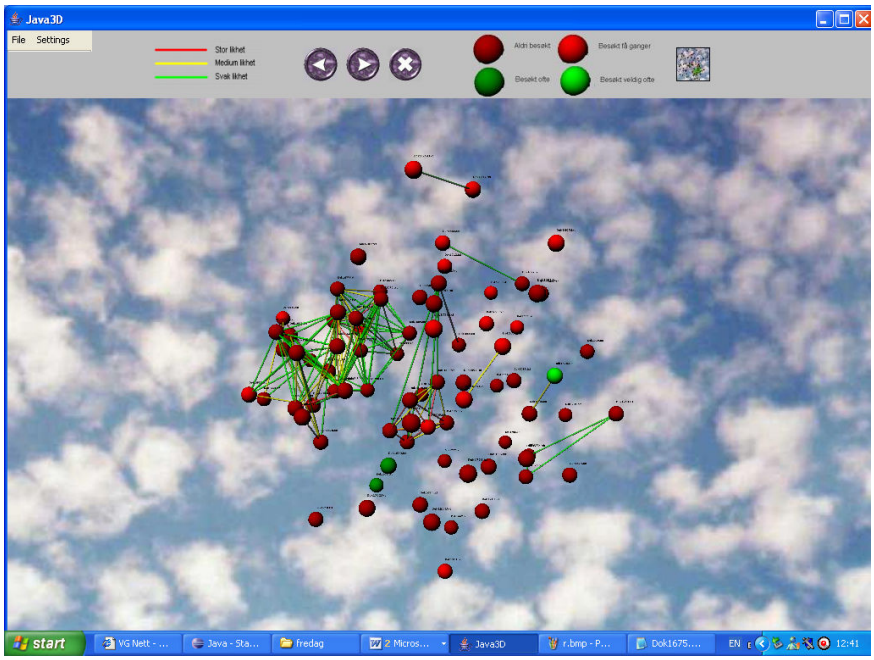
For at brukeren skal kunne se hvilke noder som har blitt besøkt så har jeg implementert en teknikk kalt frekvensbilde. Denne teknikken går ut på å vise et oversiktsbilde der det vises hvilke noder som er besøkt og hvilke noder som ikke er besøkt.

Jeg har valgt å legge til frekvensbilde knapp i prototypen. Når denne knappen trykkes så forandres fargene på kulene i grafen seg i forhold til hvor mange ganger de har blitt besøkt. Det er lagt til forskjellige fargenyanser her. Kulene viser sterk rød farge hvis noden aldri har blitt besøkt. Når nodene har blitt besøkt et viss antall ganger så vil nodene forandre fargen til en lavere intensitet av rødt. Når nodene er besøkt mange ganger så vises de med en grønnfarge. Når nodene har kommet over et vist punkt vil node fargen være lysegrønn.

I prototypen lagres antall ganger en node er besøkt til fil når programmet avsluttes, når prototypen åpnes neste gang så leses disse dataene inn og oppdaterer frekvensbildedataene. Frekvensbildedataene blir oppdatert underveis under kjøring også men blir ikke lagret til fil før brukeren velger å avslutte programmet.



Figur 8.13: Skjermbilde der nodene vises normalt.



Figur 8.14: Skjermbilde der frekvensbilde er valgt.

Prototypen viser hva de forskjellige fargene på kulene representerer i frekvenskartet. Figur 8.15 viser denne informasjonen om hva de forskjellige fargene til nodene representerer i frekvensbilde. Den viser også knappen som setter frekvensblide på og av.

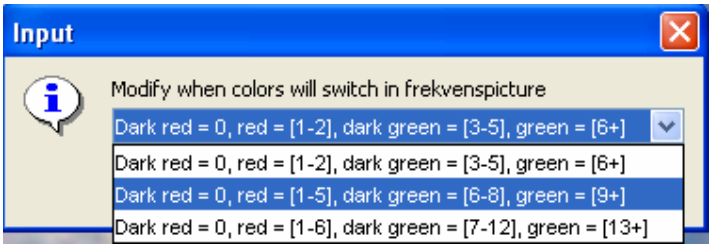
Figur 8.15: forklaring på fargenoder til frekvensblide knappen til høyre setter frekvensbilde på og av.

I prototypen kan brukeren velge mellom tre innstillinger som styrer når noden skal forandre farge. Jeg har valgt å ta med tre forskjellige innstillinger. Disse verdiene er vist i tabell 8.1

	Mørk rød	**Rød**	**Mørk grønn**	**Lys grønn**
Innstiling 1	Aldri besøkt	Besøkt 1-2 ganger	Besøkt 3-5 ganger	Besøkt mer en 6 ganger
Innstiling 2	Aldri besøkt	Besøkt 1-5 ganger	Besøkt 6-8 ganger	Besøkt mer en 9 ganger
Innstiling 3	Aldri besøkt	Besøkt 1-6 ganger	Besøkt 7-12 ganger	Besøkt mer en 13 ganger

Tabell 8.1: De tre forskjellige innstillingene som bestemmer når fargene skal forandre seg i frekvensbilde

Brukeren kan stille inn dette fra menybaren ved å velge menyvalget frequens data fra settingsmenyen. Brukeren velger fra en popupliste hvilke av disse innstillingene som skal brukes.



Figur 8.16: Liste der brukeren kan velge hvilke innstilling som skal gjelde for frekvensbilde.

Personlige stier

Jeg har lagt til mulighet for brukere til å lagre egen personlige stier. Dette gjøres ved at brukeren velger en sekvens av noder i konseptkartet.
Rekkefølgen disse noden blir valgt kan lagres til fil som en personlig sti. Denne stien kan hentes opp igjen når brukeren ønsker det. Det er mulig å lagre en uendelig mengde med stier siden brukeren selv velger navnet på filen stien lagres i.

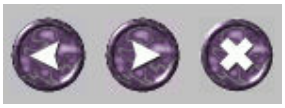
Slike stier gir mulighet til å lagre en sekvens av noder som inneholder informasjon som ofte er benyttet av den enkelte bruker.
Man har også stier som kalles ledede turer. Dette er stier som en forfatter eller en veileder har konstruert fra node til node. En bruker som vil følge en ledet tur kan følge nodenes forutbestemte rekkefølge. Slike ledede turer kan benyttes for opplæring av brukere og gir mulighet til å generere en sekvensiell sekvens selv om man representerer dataene i et konseptkart. Man kan si at det er lagt til en metode for sekvensiell framstilling i konseptkartet. Disse kan opprettes på samme måte som personlige stier, men her benyttes de personlige stiene som ledede turer i prototypen.

I prototypen åpnes og lagres stiene eller de ledede turene fra fil via en meny. Når brukeren har lastet inn en sti så benyttes framover pilen til å følge sekvensen av noder. Det er også mulig å gå tilbake til noder man har besøkt med tilbake pilen.

Gå tilbake mulighet

Jeg har i prototypen lagt til mulighet til å navigere mellom forskjellig noder som har blitt besøkt. Prototypen følger her brukerens bevegelser og lagrer hvilke noder brukeren har besøkt. Brukeren kan da bevege seg bakover til noder som er besøkt før, dette er en god teknikk hvis brukeren roter seg bort. Brukeren får med denne teknikken mulighet til å gå tilbake et par noder isteden for å starte helt på nytt.

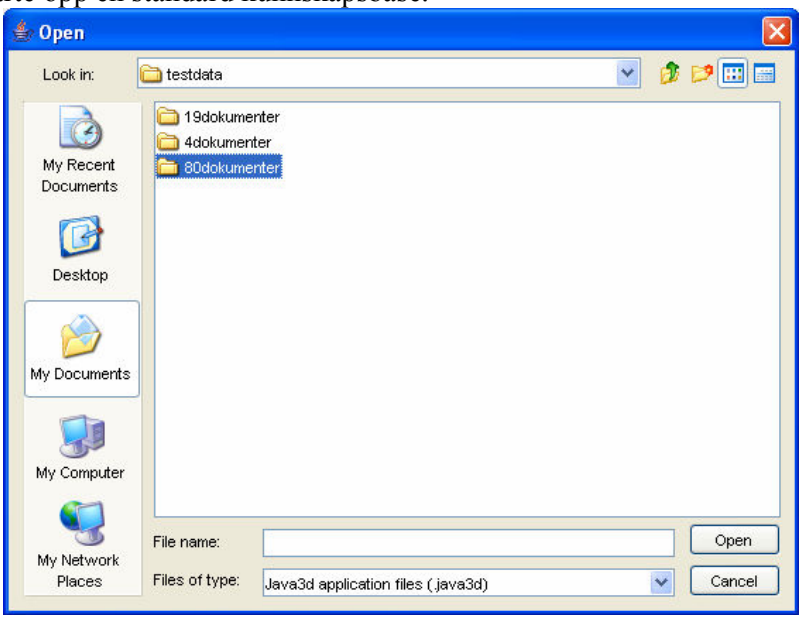
Jeg har også implementert en "gå fram" knapp i tilfelle brukeren går et skritt for langt tilbake. Dette er implementert ved å programmere 2 staker som oppdaterer hverandre i forhold til brukerinnputt. Figur 8.17 viser navigasjonsknappene som brukes for å gå fram og tilbake til noder:



Figur 8.17: knapper som brukes til navigasjon mellom noder. Det er også lagt til knapp som avslutter programmet.

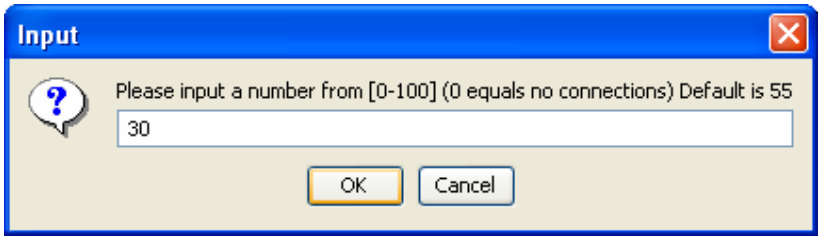
8.2.6 Åpning av kunnskapsbase i prototypen

Før programmet kan visualisere en kunnskapsbase så må brukeren velge den ”basen” som skal visualiseres. Dette gjøres ved at brukeren spesifiserer mappen dataene ligger i og velger filen med filtypen java3d, programmet vil da laste inn denne kunnskapsbasen. Velger brukeren en ukjent filtype eller en fil som ikke har noe med prototypen å gjøre så vil prototypen starte opp en standard kunnskapsbase.



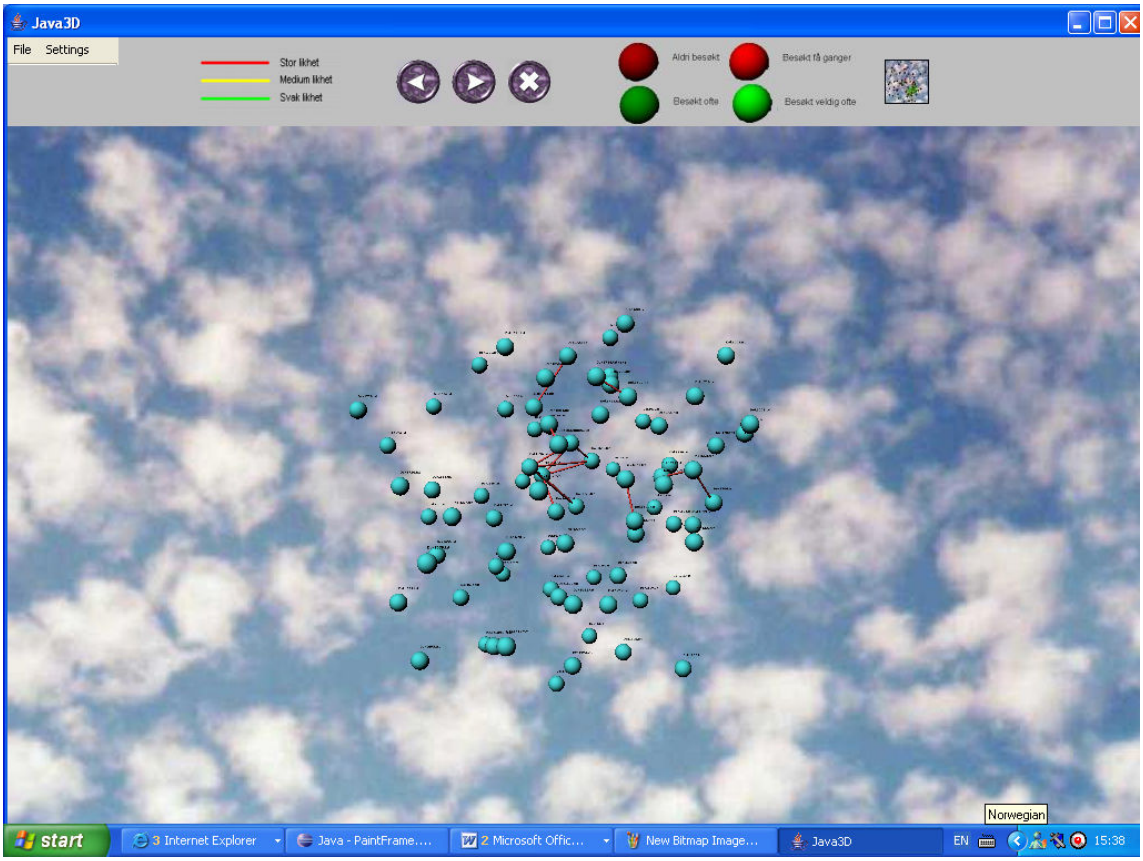
Figur 8.18: valg av kunnskapsbase

Når brukeren har valgt den kunnskapsbasen som skal visualiseres så må brukeren velge en terskelverdi for koblingene til konseptkartet. Hvis brukeren ikke skriver inn en terskelverdi vil programmet bruke terskelverdien 55. 55 omgjøres til reell terskelverdi i programmet som vil tilsvare 0,45. Alle noder med større likhet en 0,45 vil det da dannes kobling mellom i kunnskapsbasen. Likhet 0,00 tisvarer ingen likhet mens 1,00 tilsvarer 100 % likhet. Grunnen til at jeg har innputt verdi 0-100 med 0 som ingen koblinger er at brukeren ikke skal bli forvirret med at et lavt tall gir mange koblinger. Det er derfor i innputt verdiene et høyt tall som gir et stor mengde med koblinger, mens et lavt tall gir få eller ingen koblinger.



Figur 8.19: Innputt til terskelverdi i programmet som styrer antall koblinger som skal vises

Når kunnskapsbasen er valgt og terskelverdien er satt så vil prototypen visualisere kunnskapsbasen slik som i figur 8.20:



Figur 8.20x: Kunnskapsbase med 80 dokumenter og terskelverdi 0,70 (brukerinput 30)

8.2.7 Lagring av data til kunnskapsbasene

All lagring av data til en kunnskapsbase skjer i en egen mappe som tilhører kunnskapsbasen. For eksempel så vil de ledede turene og stiene være lagret i en egen path mappe i den kunnskapsbase mappen de tilhører. Når brukerne vil åpne en sti vil de få fram path mappen som tilhører den nåværende visualiserte kunnskapsbasen. De sliper å lete rundt i filsystemet etter disse stiene. Alle data slik som frekvensbildeinformasjon og nodenes plassering vil også bli lagret i mappen som tilhører nåværende kunnskapsbase. Prototypen laster inn riktig data ut fra hvilken kunnskapsbase som brukeren velger ved oppstart.

8.2.8 Automatisk generert konseptkart

Jeg har i avsnittene ovenfor forklart hvordan jeg har omdannet en likhetsmatrise fil og en liste over ressurser om til et automatisk generert konseptkart. Selve konseptkartet har en tredimensjonal form som gir den er bedre visuelt utrykk Jeg har også beskriv de metodene jeg har lagt til for å løse eventuelle desorienterings problemer Jeg har i løsningen lagt vekt på de estetiske og funksjonelle siden dette er viktig i en læringssammenheng.

I det neste kappitlet vil jeg teste ut konseptkartet både visuelt sett og brukbarheten til løsningene jeg har valgt.

9 Testing og diskusjon

9.1 Testgrunnlag

Jeg vil i dette kapittelet utføre en rekke forsøk mot det tredimensjonale konseptkartet jeg har laget. Prototypen kan være krevende for prosessoren, jeg har i min evaluering av konseptkartene brukt følgende system:

Skjermkort:	RADEON X300 SE 128 MB
Prosessor:	Intel Pentium 4 3.2 GHz
Minne:	1 GB RAM
Operativsystem:	Microsoft Windows XP Professional SP 2

Skjermstørrelse og prosessor som brukes til å vise konseptkartet har stor betydning. En stor skjermstørrelse vil bestemme hvor stort man kan visualisere konseptkartene, Det vil for eksempel være plass til flere konsepter på en større skjerm. Beregningshastigheten til prosessoren har innvirkning på hvor god interaktiviteten til konseptkartet blir, siden manøvrering av konseptkartet kan være svært krevende for prosessoren ved visualisering av store grafer.

9.1.1 Kriterier for evaluering

For å vurdere hvor vellykket min prototyp er for å visualisere konseptkart så vil jeg legge vekt på følgende:

Visualisering: Konseptene må visualiseres på en god og oversiktlig måte, for å vurdere dette så vil jeg komme tilbake til aspektene jeg kom innom i kapitlet om visualisering. Størrelse, forutsigbarhet, fokusering og desorienteringsproblemer.

Tidsforbruk: Jeg vil komme innom en rekke tester for å sjekke hvor lang tid prototypen bruker på å visualisere konseptkartet, jeg vil her prøve å finne begrensingen for konseptkartets antall objekter som kan visualiseres.

9.2 Størrelse på konseptkartet

Størrelsen på konseptkartet er viktig i forhold til utseende på presentasjonen, det er som sagt begrenset hvor mange noder som kan visualiseres samtidig.

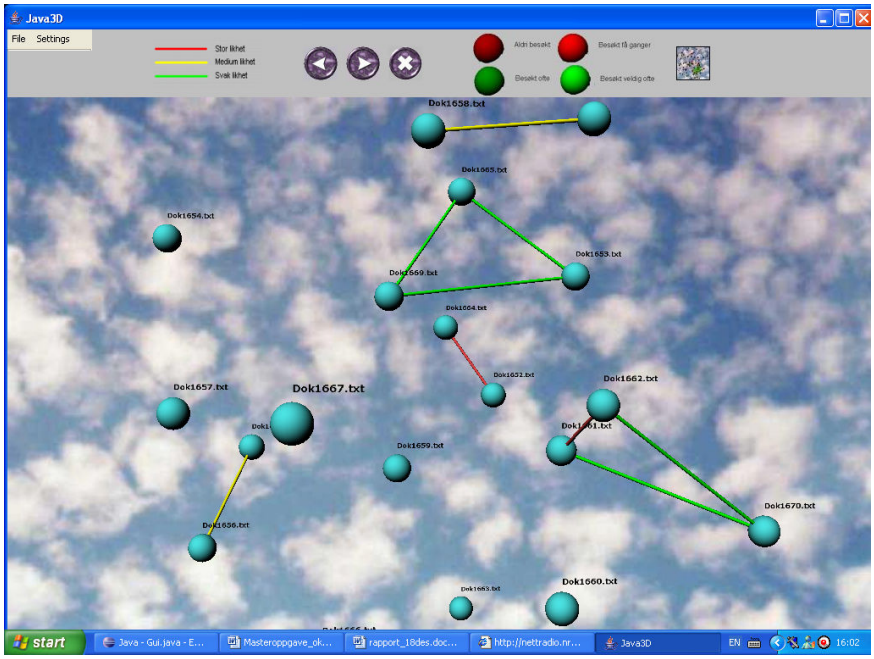
Jeg vil her teste hvordan konseptkartet oppfører seg rent visuelt sett, i forhold til hvor mange konsepter som blir presenteret i konseptkartet. Jeg har valgt å dele opp konseptkartet i tre kategorier. De små konseptkartene er konseptkart på størrelse opp til 20 konsepter, de mellomstore konseptkartene vil inneholde 20-60 konsepter, mens de store konseptkartene vil inneholde 60-220 konsepter.

For å vurdere de forskjellige konseptkartkategoriene så vil jeg kvalitativt teste en rekke konseptkart fra hver konseptkategori. Jeg vil legge vekt på følgende ved min vurdering:

Oversiktlighet: Hvor god er oversikten i konseptkartet og hvor lett ser man sammenhengene mellom konseptene.
Tredimensjonaliteten: Hvordan fungerer de tre dimensjonene i forhold til konseptkartets størrelse, jeg vil her spesielt se på dybdeføleIsen.
Detaljer: Hvor godt er detaljene bevart i konseptkartet ved ulik størrelse, jeg vil her komme inn på detaljer som hvor enkel det er å lese konseptitlene.
Fordeling av noder: Hvor godt er nodene fordelt i konseptkartet?

Små konseptkart

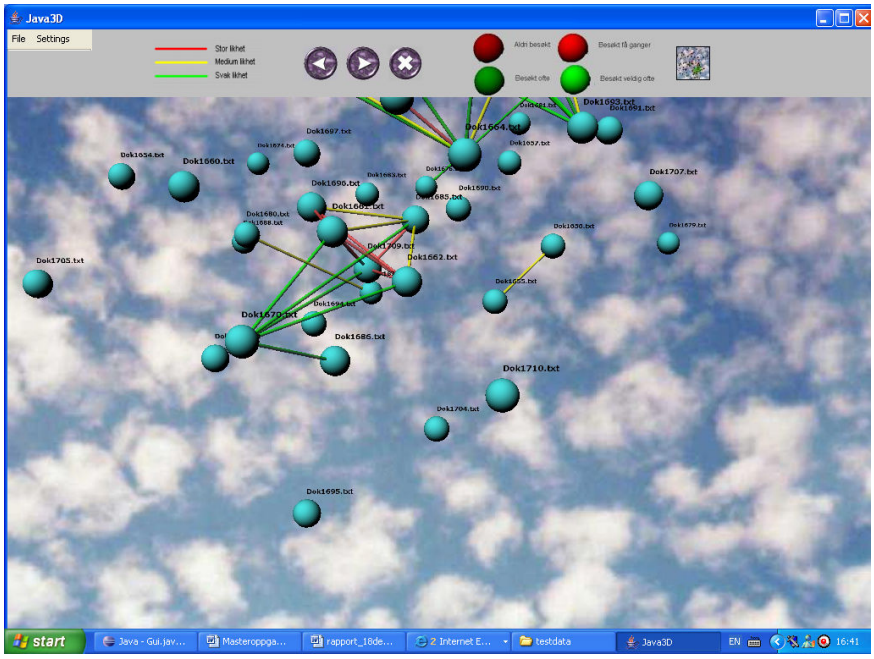
I evaluering av de små konseptkartene har jeg sett på konseptkart med størrelse på antall konsepter opp til 20. Oversikten i disse konseptkartene er god siden det er enkelt å få oversikt når det er få konsepter i konseptkartet. Det tredimensjonale fungerer også godt. Avstanden mellom konseptene er enkel å se, man ser helt klart hvilke konsepter som er plassert framme i konseptkartet i forhold til de som ligger lenger bak, dybdeføleIsen er derfor veldig god. Denne avstanden blir sterkere visualisert ved at brukeren translerer grafen.
Detaljene kommer også godt frem i de små konseptkartene. Konseptnavnene vises klart og tydelig. Fordelingen av nodene er svært god i de små konseptkartene. De plasseres godt i forhold til likhetsverdiene.



Figur 9.1: Testsett med 19 konsepter

Mellomstore konseptkart

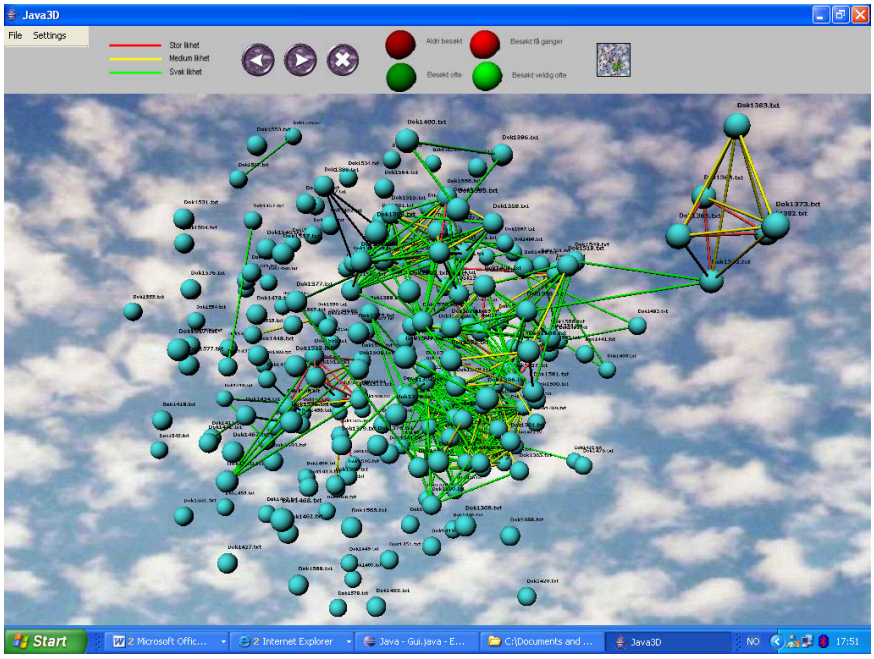
I evaluering av de mellomstore konseptkartene har jeg sett på konseptkart med størrelse fra 20 til og med 60 konsepter. Det er i disse konseptkartene vanskeligere å få oversikt en i de små konseptkartene men oversikten er enda tilfredsstillende.
Den tredimensjonale effekten kommer også her godt frem, det er også mulig å lese konsepttitlene relativ greit, man må derimot zoome nærmere inn på grafen for å lese dem. Fordeling av nodene er jeg fortsatt fornøyd med. Man ser lett i konseptkart hvilke konsepter som er relativt uinteressante i sammenhengen, ved at de er plassert alene og langt ut til sidene av konseptkartet. Man ser også her antyding til klyngedannelser. Man får nå egenskaper i konseptkartet som gjør at man kan tolke på makronivå i store nok konseptkart. Man ser ikke hvert enkelt konsept men grupper av konsepter.



Figur 9.2: Testsett med 60 konsepter

Store konseptkart

I evalueringen av de store konseptkartene har jeg sett på konseptkart med størrelse på 60 til 220 konsepter. Oversikten i disse konseptkartene er mye dårligere en i de andre siden det blir alt for mange konsepter brukeren må forholde seg til. Den tredimensjonale effekten er ikke like god nå siden det er for mange konsepter. Det er her intuitivt vanskeligere å se hvilke konsepter som er nær og lenger unna hverandre enn i de mindre konseptkartene. Det er også vanskelig å lese konsepttitlene siden fordelingen av nodene ikke er god nok. Nodene er nå fordelt i en oppklumping mot midten. Dette var ikke et problem i de mindre konseptkartene men når antall konsepter som skal visualiseres overstiger 100 så er ikke spring embedderen like god til å spre nodene i konseptkartet Jeg har i tabell 9.1 prøvd å oppsummere de forskjellige størrelse kategoriene i forhold til kriteriene jeg har testet etter.



Figur 9.3: testsett med 220 konsepter

Kategori	Oversiktlighet	Tredimensjonalitet	Detaljer	Fordeling av noder
Små	God	God	God	God
Mellomstore	God	God	Tilfredsstillende	God
Store	Dårlig	Dårlig	Dårlig	Dårlig

Tabell 9.1: Sammenligning av størrelse kategoriene

Størrelse

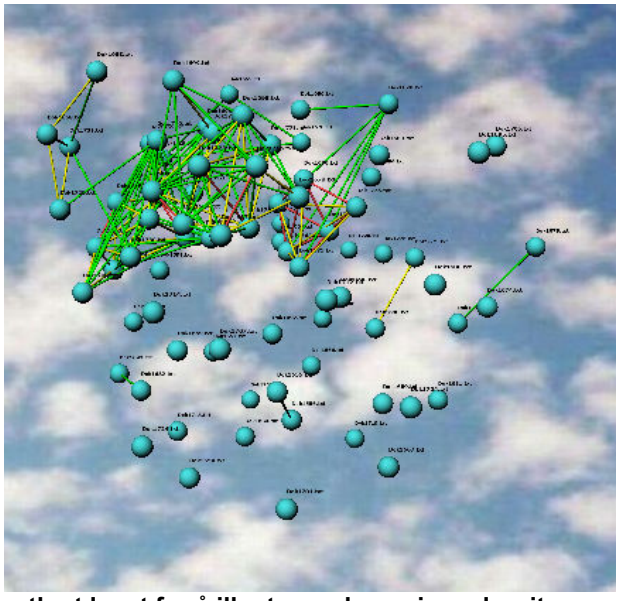
Testresultatene for størrelsen til konseptkarten var klare. De små og mellomstore konseptkartene fungerer meget godt. De store konseptkarten har mistet den tredimensjonale følelsen og detaljene er blitt borte. Fordelingen av nodene er også dårligere i de store konseptkartene siden det dannes en oppklumping mot midten.
De små og mellomstore konseptkartene gir en mer detaljert oversikt. De store konseptkartene fungerer litt annerledes ved at man først gir en oversikt over en stor mengde konsepter uten

detaljer, brukeren kan deretter zoome inn på områder som er av interesse. Jeg mener at rundt 100 konsepter så blir konseptkartet så stort at det er vanskelig å holde oversikten. Det er ikke antall konsepter alene som setter en begrensing, det er antall koblinger pluss antall konsepter som har innvirkning på hvor oversiktlig konseptkartet er. Et konseptkart med mange konsepter med få koblinger kan være mer oversiktlig en et konseptkart med færre konsepter og mange koblinger. Blir konseptkartet for uoversiktlig så kan brukeren sette ned terskelverdien slik at det blir vist færre koblinger for å få bedre oversiktlighet.

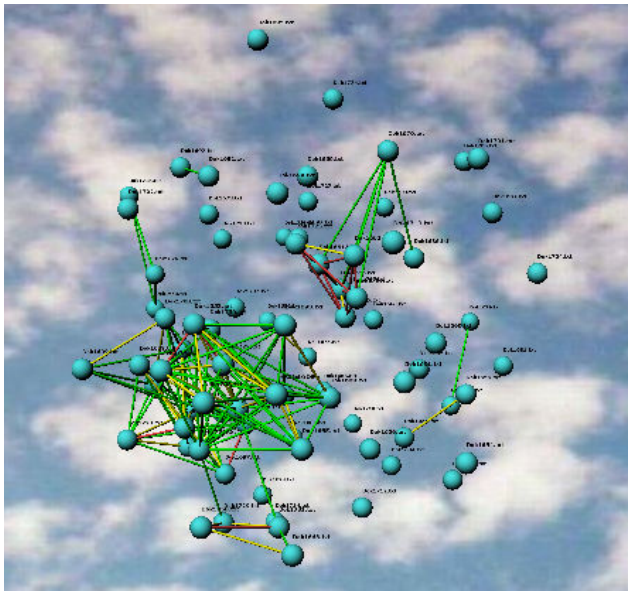
9.3 Forutsigbarhet

Jeg benytter i oppgaven en plasseringsalgoritme kalt spring embedder, denne har en svakhet at den er lite forutsigbar, forutsigbarheten er helt grunnlegende for å forhindre at visualiseringen forvirrer. Jeg forklarte kapittelet om prototyp hvordan jeg har løst dette ved å lagre de beregnede punktene til fil.

Siden jeg har valgt å kjøre plasseringsalgoritmen kun første gangen man starter kunnskapsbasen så er forutsigbarhetene bevart i min prototyp. Hver gang brukeren starter konseptkartet så vil den se lik ut hver gang. Hadde prototypen beregnet ny plassering for nodene hver gang ville konseptkartet forandret seg hver gang slik som illustrert i figurene nedenfor. Man kan se at konseptkartene ser svært forskjellig ut.



Figur 9.4: Konseptkart laget for å illustrere plasseringsalgoritmens uforutsigbarhet



Figur 9.5: Konseptkart laget for å illustrere plasseringsalgoritmens uforutsigbarhet

9.4 Fokusering i konseptkartet

Fokusering er viktig for å gi brukeren mulighet til å utforske forskjellige deler av konseptkartet. Dette er spesielt viktig i de store konseptkartene.

Jeg har lagt til 2 fokuseringsteknikker i min prototyp, zooming og koplingsmanipulasjon.

Jeg vil her prøve å teste hvor vellykket disse fokuseringsteknikkene er ved å måle:

Fokusering: Teknikkens egenskaper til å sette konseptet eller området som brukeren ønsker å sette fokus på.
Kontekst: Det er viktig at man ikke mister oversikten over helheten når man skal fokusere på et område. Jeg vil derfor se på hvordan konteksten holdes i fokuseringsteknikkene.

Zooming

Det er i konseptkartet mulig å manøvrere konseptkartet både ved å zoome inn og ut, flytte sidelengs eller rotere konseptkartet. Dette gir brukeren mulighet til å se konseptkartet fra forskjellige synsvinkler. Fokuseringen her er god siden man kan zoome inn og ut på området for å gjøre det så stort man ønsker. Man kan derimot miste oversikten over konteksten hvis man fokuserer på deler av konseptkartet på denne måten.

Koplingsmanipulasjon

Denne fokuseringsteknikken går ut på å vise kun de koblingene som er koblet til det konseptet man klikker på, alle de andre koblingene i konseptkartet fjernes. Man har her ikke noen form for fysisk forstørring, men man får fjernet uinteressant informasjon fra andre deler av konseptkartet for å sette fokus på den delen av konseptkartet som brukeren er interessert i.

Bruk av denne teknikken gjør konseptkartet mer oversiktlig, metoden er svært brukbar i store og uoversiktlige konseptkart med mange koblinger. Selve konteksten bevares i denne fokuseringsteknikken. Siden konteksten er lik den man hadde før man begynte å fokusere.

Fokuseringsteknikk	Fokusering	Kontekst
Zoom	Meget god	Dårlig
Koplingsmanipulasjon	God (Ikke i store konseptkart)	Perfekt bevart

Tabell 9.2: Sammenligning av fokuseringsteknikkene.

Fokuseringsteknikkenes hastighet

Jeg nevnte i kappitlet om visualisering at det er viktig for brukeren at systemet reagerer i forhold til brukerinput i sanntid. For å gjøre et system interaktivt så stilles det store krav til ytelse, jeg vil her finne ut hvor mye prototypen kan håndtere før det går utover den interaktive følelsen til brukeren.

Jeg har utført en rekke testkjøringer med forskjellig antall konsepter i konseptkartet der terskelverdien varieres for å finne ut hvor mange objekter som en graf kan inneholde før den blir treg.

Jeg har delt opp manipulasjonen mot grafen i to grupper:

- Manipulasjon 1: Rotasjonshastighet, translering og zooming av grafen.
- Manipulasjon 2: Koblingsmanipulasjon ved valg av konsept.

Verdiene jeg har gitt de forskjellige manipulasjonene er:

- Rask: Ingen tegn til treghet ved brukerinput mot grafen
- Litt treg: Litt treghet men grafen er så rask at det ikke er irriterende
- Treg: Her er tregheten så stor at jeg mener at grafen ikke bør brukes

Antall konsepter	Antall koblinger	Antall objekter	Terskelverdi	Manipulasjon 1	Manipulasjon 2
19	9	28	0,45	Rask	Rask
19	54	73	0,3	Rask	Rask
60	72	132	0,45	Rask	Rask
60	477	537	0,3	Rask	Rask
80	215	135	0,45	Rask	Rask
80	913	833	0,3	Litt treg	Litt treg
220	398	610	0,45	Rask	Rask
220	610	830	0,4	Litt treg	Litt treg
220	1156	1376	0,35	Treg	Treg

Figur 9.3: Tabell over testverdier

Ser man på testdataene så er det ikke antall konsepter som bestemmer om grafen oppdateres raskt. Det er antall koblinger pluss antall konsepter (kuler) som bestemmer om interaksjonen mot grafen er rask eller ikke.

De små konseptkartene hadde ingen problemer med ytelse, men når antall konsepter + koblinger nærmet seg 800 så begynte grafenes oppdatering å bli litt treg, denne treghetene var så liten at den ikke var irriterende. Men når antall objekter nærmet seg 1000 begynte manipulasjonene å bli tregere og tregere, rundt 1400 objekter er oppdateringene så trege at jeg mener de ikke bør brukes.
Med et slikt antall objekter så blir konseptkartene lite brukervennlige, dette fordi det blir for mange koblinger som gjør konseptkartet uoversiktlig og manipulasjonen mot den blir for treg.

9.5 Tidsforbruk

Jeg har kjørt en rekke tester på forskjellig antall konsepter i konseptkartet for å finne ut hvor krevende spring embedderen er i forhold til antall konsepter.
For å teste dette har jeg valgt å ta tiden programmet bruker på å plassere konseptene med algoritmen.
Jeg har også i disse testene registrert totaltiden som Java bruker på framstillingen av konseptkartet.

9.6 Beregning av tid

For å beregne tiden så benytter jeg meg av system metoden System.currentTimeMillis(). Denne metoden returnerer differansen målt i millisekunder mellom nåværende tid og midnatt 1. januar 1970.

Ved å bruke denne metoden kan man lage en slags stoppeklokke:

Når man starter stoppeklokka hentes differansen ut og lagres:

```
startvalue = System.currentTimeMillis();
```

Når stoppeklokka skal stoppes så hentes den nye differansen ut og lagres:

```
stopvalue = System.currentTimeMillis();
```

For å beregne tidsforskjellen finner man differansen mellom stopp og startpunktet:

```
Målt tid = (stopvalue-startvalue).
```

9.7 Testkjøringsverdier og grafer

Det er blitt utført tester på konseptkart med 80, 220 og 287 konsepter.

Det er i testene kun en variabel som blir forandret. Denne verdien kalles terskelverdien og er en verdi i programmet som bestemmer hvor høy likheten mellom konseptene må være for at det skal kobles en kant mellom dem. Jo lavere denne terskelverdien er jo flere kanter (koblinger) blir det for programmet å visualisere.
Det blir i programmet beregnet to tider:

- Totaltiden som det tar for Java å starte applikasjonen.
- Tiden det tar for spring embedderen å utføre algoritmen.

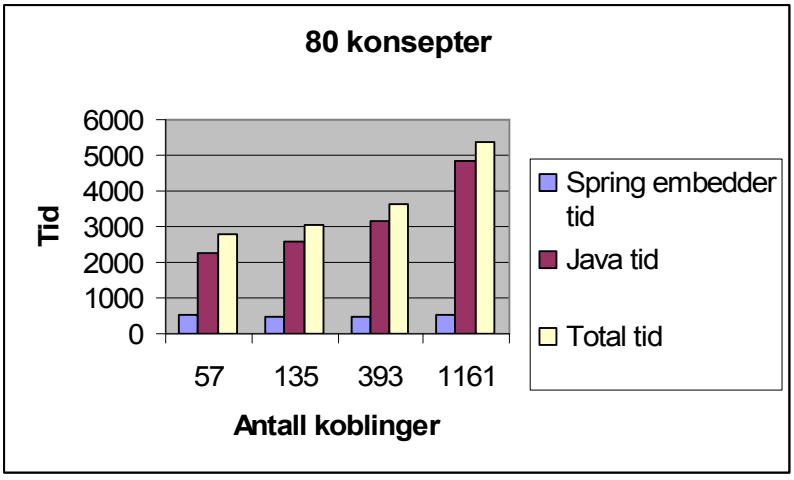
Ut fra disse tidene er det beregnet en tid som jeg har valgt å kalle Javatid. Dette er tiden det tar for Java å starte applikasjonen minus tiden spring embedderen bruker.

Dataene etter kjøringene er plassert i tre tabeller og det er også opprettet tre grafer for å lettere kunne sammenligne spring embeddertiden og Javatiden.

Grafen viser også den totale tiden det tar å få startet applikasjonen.

Antall konsepter	80	80	80	80
Terskelverdi	0,55	0,45	0,35	0,28
Antall koblinger	57	135	393	1161
Spring embeddertid	516 ms	500ms	453 ms	531 ms
Javatid	2265 ms	2578 ms	3157 ms	4859 ms
Totaltid	2781ms	3078 ms	3610 ms	5390 ms

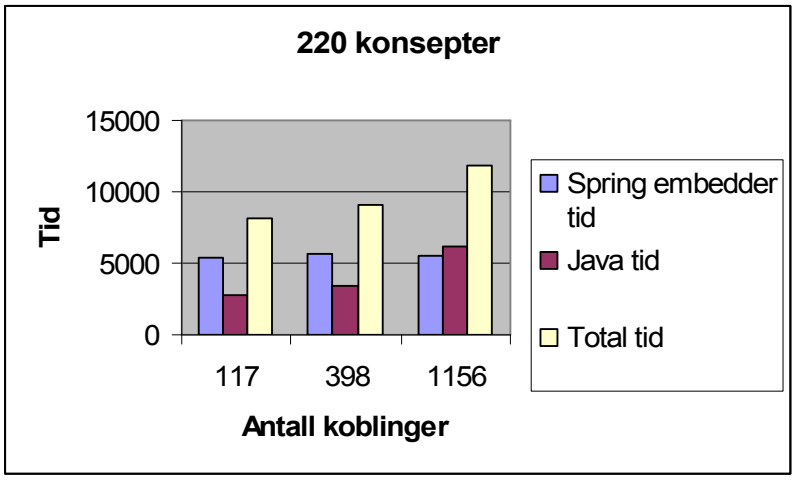
Tabell 9.4: Testverdier fra konseptkart med 80 konsepter



Figur 9.6: Tidsforbruk ved konseptkart på 80 konsepter

Antall konsepter	220	220	220
Terskelverdi	0,55	0,45	0,35
Antall koblinger	117	398	1156
Spring embedder tid	5360 ms	5609 ms	5578 ms
Javatid	2828 ms	3485 ms	6219 ms
Totaltid	8188 ms	9094 ms	11797 ms

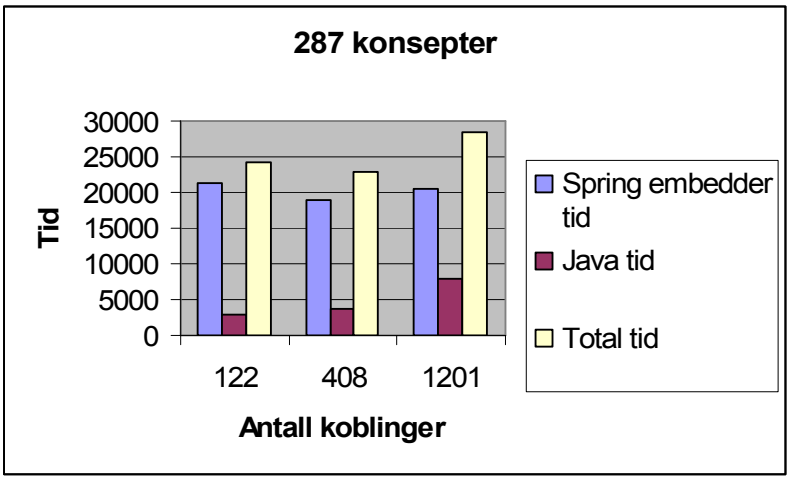
Tabell 9.5: Testverdier fra konseptkart med 220 konsepter



Figur 9.7: Tidsforbruk ved konseptkart på 220 konsepter

Antall konsepter	287	287	287
Terskelverdi	0,55	0,45	0,35
Antall koblinger	122	408	1201
Spring embeddertid	21360 ms	19047 ms	20437 ms
Javatid	2952 ms	3749 ms	8001 ms
Totaltid	24312 ms	22796 ms	28438 ms

Tabell 9.6: Testverdier fra konseptkart med 287 konsepter



Figur 9.8: Tidsforbruk ved konseptkart på 287 konsepter

9.8 Tolking av grafene og tallene

Ut fra dataene så ser man at spring embedderen bruker tilnærmet like lang tid på det samme datasettet hver gang. Dette er fornuftig siden algoritmen blir styrt av antall noder og bryr seg ikke om hvor mange koblinger det er i grafen.

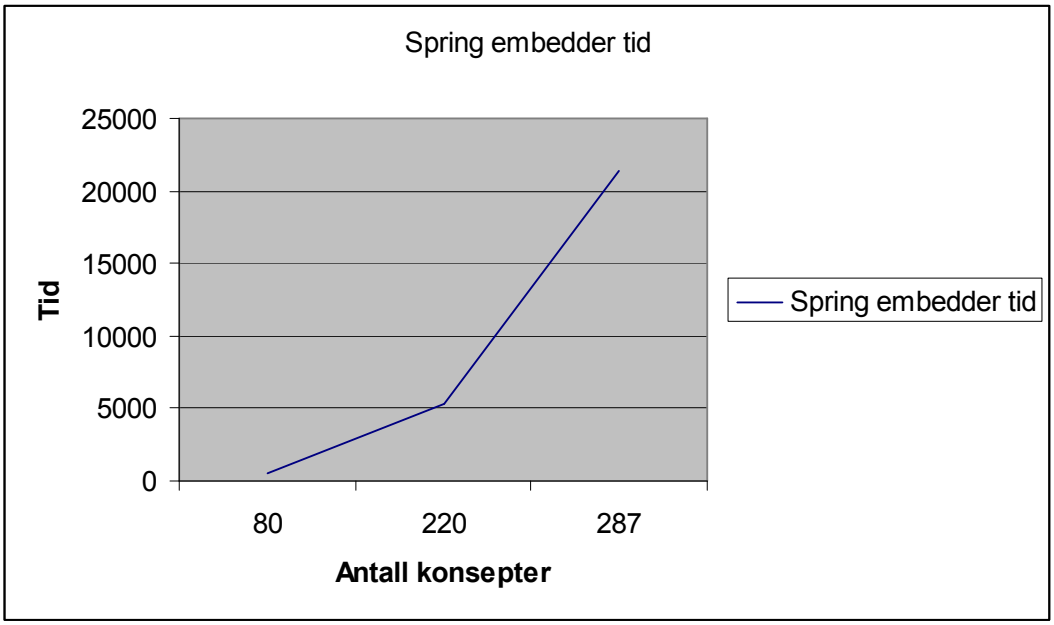
Ser man på Java tiden så øker den når terskelverdien settes ned, jo lavere terskelverdien er jo flere koblinger er det som må tegnes opp. Jo mer opptegning jo tregere blir det å få opp resten av programmet.

Ser man på grafen og tallene til datasettet med 80 dokumenter så er tiden spring embedderen bruker mye mindre en det Java bruker på å sette opp resten av programmet.
Når derimot antall konsepter øker så øker tidsforbruket til spring embedderen raskt.

- Fra 80 til 220 konsepter så øker tidsforbruket med en faktor på 10.
- Fra 80 til 287 konsepter så øker tidsforbruket med en faktor på 40.
- Fra 200 til 287 konsepter så øker tidsforbruket med en faktor på 4.

Antall konsepter	80	220	287
Spring embedder tid	516	5360	21360

Tabell 9.7: Spring embedder tid



Figur 9.9: Graf som illustrerer plasseringsalgoritmens tid

Det er tegnet opp en graf for å illustrere økningen tiden spring embedderen bruker for å spre nodene. Ut fra tallene og grafen så øker tiden til algoritmen kraftig med antall noder som behandles. Dette stemmer siden plasseringsalgoritmen er iterativ og en rekke kalkulasjoner blir beregnet flere ganger og da vil tiden den bruker øke kraftig i forhold til antall noder som blir behandlet.

Ut fra disse konseptkartene så kan man trekke en konklusjon om at spring embedderen bruker liten tid i forhold til det Java bruker for å tegne opp grafen når antall konsepter er få. Når antall konsepter blir flere så øker tiden spring embedderen bruker på å spre nodene mye raskere en den tiden som Java bruker for å få visualisert grafen.

9.9 Begrensing for antall objekter

Under testkjøringen har jeg sett på tiden spring embedderen bruker for å spre forskjellige antall konsepter i konseptkartet. Når antall konsepter overstiger 200 så begynner spring embedderen å bruke veldig lang tid på å spre konseptene i konseptkartet. Grensen for antall konsepter som bør visualiseres er relativ i forhold til hvor lang tid det er akseptabelt for brukerne å vente på at Java skal starte programmet.

Når antall konsepter var på 287 tok det hele 30 sekunder for å få fram visualiseringen, mens det tok 8 sekunder å visualisere 220 dokumenter, og kun 3-5 sekunder for å få visualisert 80 dokumenter.
Siden prototypen kun kjører plasseringsalgoritmen ved første kjøring så er ikke dette er kjempestort problem. Det vil kun ta lang tid ved første kjøring, ved andre og senere kjøringer på samme konseptkart vil det ikke være en slik tilleggs tid. Innlesning av koordinater fra fil er ubetydelig i forhold til tiden beregningen av plasseringsalgoritmen bruker. Man kan si at løsningen min sparer fra 3 sekunder og opp til 30 sekunder hvis plasseringsalgoritmen allerede har vært kjørt på de konseptene som skal visualiseres.

Siden plasseringsalgoritmen bruker mye tid ved første kjøring mener jeg det er bedre å finne en begrensing på antall objekter som kan visualiseres.

Java 3d setter en indirekte begrensning på antall konsepter som kan visualiseres.
Når terskelverdien blir satt slik at antall konsepter pluss kanter overstiger 1000 så begynner manipulasjon mot grafen å bli treg på testmaskinen. Når antallet stiger over 1400 er manipulasjonene alt for trege.
Antall konsepter + kanter som bør maksimalt visualiseres på testmaskinen ligger på rundt 1000 objekter.

Ved å bruke antall objekter som begrensing så avhenger det av konseptene og den innstilte terskelverdien. Et konseptkart med få konsepter kan håndtere en mye lavere terskelverdi en et konseptkart med mange konsepter, siden det i de små ikke vil bli dannet så mange koblinger.

Etter å ha testet med forskjellige konseptkart så mener jeg en terskelverdi rundt 0,35-0,45 passer godt for de fleste konseptkartene. En god terskelverdi avhenger selvfølgelig av antall konsepter, siden de indirekte bestemmer hvor mange koblinger som også tegnes opp. Ved små konseptkart vil en terskelverdi rundt 0,35-0,40 være et godt valg. Mens i større konseptkart så bør antall koblinger holdes lavt på grunn av oversiktlighet og hastighet i grafen. En terskelverdi på 0,45 eller høyere vil være bedre i større konseptkart. Min mening om gode terskelverdier kontra antall koblinger som bør vises kan for en annen bruker være alt for liten eller alt for stor, derfor er det lagt til mulighet for brukeren å forandre denne terskelverdien ved oppstart av konseptkartet.

10 Konklusjon

I denne oppgaven har jeg laget en modell for å organisere og presentere dokumenter. For å få til dette har jeg laget et tredimensjonalt konseptkart.

Konseptkartene jeg har testet på kunne ha vært flere og mer varierende. I testene er det rimelig like dokumenter som skal visualiseres. Alle dokumentene omhandler økonomi. Dette gjør at likhetsmatrisen som er innputt til prototypen har høye likhetsverdier. I et annet datasett med flere mindre like dokumenter ville likhetsverdiene vært lavere.

Dette har innvirkning på programmet, lavere terskelverdier i likhetsmatrisen vil føre til færre koblinger hvis man bruker en konstant terskelverdi, denne terskelverdien har jeg gitt brukeren mulighet til å stille inn før konseptkartet tegnes opp, slik at brukeren kan regulere dette i forhold til datasettene.

Men om fargene på koblingene skal være rød, gul eller grønn i prototypen bestemmes av grenser satt i programmet. I andre testsett vil det sannsynligvis være bedre med andre verdier for disse grensene som bestemmer hvilken farge koblingen skal ha. Jeg burde ha lagt til mulighet for å stille inn disse grensene i programmet, men i de testsettene jeg har testet på mener jeg at grensene er gode.

Det at dokumentene kun omhandler økonomi og gir store likhetsverdier kan være hovedårsaken til oppklumpingen mot midten når antall konsepter blir for stort. Jeg har ikke testet nok på dette til å kunne bekrefte denne antagelsen.

I testingen vektla jeg kvalitativ testing ved å studere hvert sett nøye, isteden for å gjøre en kvantitativ test. Det ville ha vært interessant å utføre flere og større kvantitative tester på prototypen, men jeg føler at resultatene fra de testene jeg utførte ga gode indikasjoner, og at resultatene ble pålitelige.

For å bestemme om prototypen er tilfredsstillende har jeg testet om konseptkartet har et godt visuelt uttrykk, og en god funksjonalitet. Dette har jeg testet i forhold til konseptkartets størrelse, forutsigbarhet og mulighetene til å fokusere i konseptkartet.

Størrelsen på konseptkartene har innflytelse på hvordan visualiseringen blir. De små og mellomstore konseptkartene ble visualisert godt, alle egenskapene i visualiseringen fungerte også godt. I de store konseptkarten mister man den tredimensjonale følelsen og detaljene er blitt borte. Man får i store konseptkart mulighet til å vise kunnskapen på et makronivå. Jeg mener at 100 konsepter er et godt estimat for maksimalt antall konsepter som bør visualiseres hvis brukeren ikke stiller inn terskelverdien. Stiller brukeren inn terskelverdien til å vise færre koblinger i større konseptkart og tar i bruk fokuseringsteknikkene så mener jeg at de store konseptkartene også fungerer tilfredsstillende.

Forutsibarheten til konseptkart er jeg svært fornøyd med, selve plasseringsalgoritmen er veldig uforutsigbar, men siden de beregnede plasseringene lagres etter første gang man konstruerer konseptkartet, så kan de hentes opp fra fil ved de neste kjøringene. Dette gjør at konseptenes plassering i konseptkartet vil være konstant fra kjøring til kjøring.

Funksjonaliteten i prototypen ble testet ved å måle prototypens evne til å fokusere. Fokuseringsteknikken er god, og hjelper til å holde oversikten i konseptkartene så lenge antall objekter som skal tegnes opp er lavt. Fokuseringsteknikkene sliter med ytelse når antall objekter i 3d grafen blir for stor. Ved rundt 1000 objekter begynner fokuseringsteknikkene å bli trege. Dette er litt synd siden det er disse teknikkene som gjør det mulig for brukeren å kunne få oversikt i store konseptkart.

11 Veien videre

Prototypen presenterer kunnskap i form av et konseptkart. Det finnes en rekke med utvidelser som ville vært interessante å jobbe videre med.

11.1 Klyngeanalyse

Konseptkartene fungerer greit opp til 100 konsepter, men hva skjer når 500 eller kanskje 1000 konsepter skal plasseres i konseptkartet?

Det vil bli en uoversiktlig presentasjon, det blir fysisk trangt på skjermen for å vise alle konseptene og det vil bli for mye informasjon i konseptkartet.

For å kunne presentere et stort antall konsepter oversiktlig nok så må man ty til andre teknikker.
Det vil være interessant her å knytte inn en klyngeanalyse i konseptkart med mange noder. Klyngeanalysen må klynge sammen de konseptene som har størst likhet og finne et fornuftig navn på klyngen. Den må også beregne en likhetsverdi mellom klyngene som opprettes. Med en slik teknikk kan man tenke seg et hierarki av klynger og konsepter.

Jeg ser for meg to måter dette kan visualiseres på i det tredimensjonale konseptkartet.

Metode 1 åpning av ressurser fra popup liste: Etter at en klyngeanalyse har blitt utført så vil klyngene spres som kuler i konseptkartet på bakgrunn av likhetsverdiene, når brukeren velger å høyreklikke på en av kulene så vil det dukke opp en popup liste. Denne popup listen vil liste opp alle ressursene til klyngen, og fra denne listen bør det være mulig å åpne hver enkelt av ressursene.
Metode 2 navigering mellom konseptkart: Når brukeren høyreklikker på en kule så vil det genereres et nytt konseptkart der ressursene til klyngen brukeren valgte vil bli framstilt. Det nye konseptkartet vil fungere på samme måte som min prototyp ved at man ved høyreklikking kan åpne ressursen som en ekstern prosess ved siden av applikasjonen. I en slik løsning så bør det være mulig å gå tilbake til oppstarte konseptkartet.

Ved en slik løsning vil det være mulig å lage konseptkart over store kunnskapsdomener, som er for store til at alle konseptene kan plasseres på skjermen samtidig.

11.2 Bruk av konseptkart innen andre områder (generalisering)

I denne oppgaven er det laget et automatisk generert konseptkart der konseptene representerer dokumenter (tradisjonelle dokumenter, læringsobjekter eller kunnskapsobjekter).
Det finnes sikkert en stor rekke med andre områder der et automatisk generert konseptkart kan brukes.

Det spesielle ved det automatisk genererte konseptkartet som er laget i min prototyp er at man ikke bruker en sekvensiell struktur for å presentere tekst. En slik måte å representere kunnskap kan være nyttig i mange sammenhenger.

For eksempel ei lærebok eller pensum i et fag, kan for eksempel presenteres i et konseptkart.

Det er også mulig å tenke seg at prototypen kobles opp mot en digital arbeidsbok der man presenterer innholdet i arbeidsboka som et konseptkart. Hvordan dette bør løses ligger langt utenfor min oppgave.

11.3 Automatisk innstilling av terskelverdi

Jeg har i prototypen latt brukeren stille inn terskelverdien som bestemmer når det skal opprettes en kobling mellom to konsepter. Det ville vært interessant å se om det fantes en objektiv terskelverdi som kan differensiere relevante og irrelevante konsepter.

Her dukker det opp en lang rekke med problemstillinger. Hvordan beregner man en slik verdi? Skal verdien være en egen verdi for hvert enkelt konseptkart? Er det mulig å finne en slik verdi?

12 Referanser og vedlegg

Referanser

[Baeza-Yates] Baeza-Yates, R.;Riberio-Neto, B. 1992. *Modern Information Retrieval*

[David Rush] Scorm Benefits
http://www.learnwise.com/downloads/David_Rush_SCORM_Intro.ppt#409,14,SCORM

[Davis] Davis 1993, What is knowledge representation
http://groups.csail.mit.edu/medg/ftp/psz/k-rep.html

[Eric, Plotnick] Concept mapping: *A graphical system for understanding the relationship between concepts.*
http://ericit.org/digest/EDO-IR-1997-05.shtml

[Gunnar Myklebost] Nettbasert læring i høgre utdanning – en samling norske erfaringer
http://www.soff.uit.no/Nettlaering_bok/kapittel_1.htm

[Hartley] Roger T Hartley Representation of Procedural knowledge
http://www.cs.nmsu.edu/~rth/publications/repProcK.pdf

[HIVE] Ulike læringsperspektiv
http://pluto.hive.no/pluto2003/espentan/Pedagogikk/Prosjekt/#hode_til_hode
[Holme] Lysark om informasjonsgjenfinning.

[Hveem.pdf] Guttorm Hveem: *Bruk av grafiske organisatorar i læringsarbeid*
http://ans.hsh.no/Biblioteket/Mastergradsoppgaver/ikt/2005/pdf/hveem.pdf

[Hypermedia] Arvid Staupe NTNU sept 2001. *Nettbaserte læringsformer Hypermedia*
http://stud.hsh.no/lu/inf/piol1/materiell/Staupe/hypermedia%20for%20kurset%20h-2001.htm

[IT 2702] Forelesnings slides
http://www.idi.ntnu.no/~agnar/it2702/2005/f6-05.pdf

[Immersion] Learning Theories and Instructional Strategies Matrix
http://www.kihd.gmu.edu/immersion/knowledgebase/

[INFO221] Slides fra forelesning fra faget INFO221
http://www.ifi.uib.no/undervisning/iv122

[Kjell A] Læring og læringsteorier
http://home.hia.no/~kjella/forelesninger/L%E6ringsteori.ppt

[Larsen] Jarle Larsen: *Private communication*

[McMillan, Bill] informatics, University of Ulster
http://ijgj229.infj.ulst.ac.uk/BillsWeb/PGCert/InfoSys/

[Nettskolen] Pedagogiske utfordringer knyttet til læring via nett
http://www.nettskolen.com/pub/artikkel.xsql?artid=106

[REN] Lær av lego
http://www2.invanor.no/upload/extranet/ren/LaeravLego.pdf

[Rijsbergen] C. J. Van, 1979. *Information Retrieval*

[Salton og McGill 1983]. Salton, Gerard and McGill, Michael J (1983) *Introduction to modern information retrieval*

[Schneiderman, Ben] The eyes have it: *A task by data type taxonomy for information visualizations*
http://citeseer.ist.psu.edu/shneiderman96eyes.html

[Simon, Steinbrückner, Lewerentz] Frank Simon, Frank Steinbrückner, Claus Lewerentz. *3D-Spring embedder for complete graphs.*
http://www-sst.informatik.tu-cottbus.de/LS-SST/Publications/2000/2000-3D-Spring_Embedder_for_Complete_Graphs.pdf

[SIMS 202] Ray Larsen og Marti Hearst SIMS 202: *Information organization and Retrieval*

[Tore Vestues] Automatisk generering av konseptkart i læring Hovedoppgave i informatikk Av Tore Vestues 2003

[UIO] Pedagogisk design
http://www.usit.uio.no/it/dlo/opplaering/forum/2005-11-10/Ped_design_1.pdf

[UIO, grafer] INF1020 − Algoritmer og datastrukturer forelesning:
http://www.uio.no/studier/emner/matnat/ifi/INF1020/h04/undervisningsmateriale/grafintroforelesning.pdf

[UIUC] Forskjellig konseptkart
http://classes.aces.uiuc.edu/ACES100/Mind/c-m2.html

[WIKIPEDIA]
Knowledge representation
Knowledge
Concept
Concept map
Mind maps
Information systems
http://no.wikipedia.org

Vedlegg

Vedlegg 1 – Klassediagram
Vedlegg 2 – Intro til Java3d
Vedlegg 3 – Ukompilert kode.zip (inneholder koden til prototypen)
Vedlegg 4 – Javadok.zip (inneholder javadok til koden)