ANVENDT INFORMATIKK eNytt

OPPTRYKK AV STUDENTARBEIDER VED HØGSKOLEN I MOLDE INSTITUTT FOR INFORMATIKK

DESEMBER 2005
NUMMER 3

REDAKSJON:
JUDITH MOLKA-DANIELSEN
HALVARD ARNTZEN

INNHOLD

Anmerkninger fra redaktørene:

Denne utgaven av Anvendt Informatikk eNytt inneholder fire utvalgte studentprosjekter ved institutt for informatikk i høstsemesteret 2005. Rapportene som er presentert kan være noe forkortet i forhold til originalarbeidene, men de er ellers ikke redigert eller modifisert av redaksjonen eller andre.

Stikkord for de presenterte arbeidene er: søking, indeksering, *degree distribution*, *hubs* og *reliability*.

1. SØKING/INDEKSERING

AV - ROGER OSHAUG, SIGBJØRN VAKSVIK OG KRISTIAN JOHANNESSEN

Del I. Innledning

Informasjon er ikke mye verdt hvis den ikke ligger lett tilgjengelig for de som skal bruke den. Informasjonsarkitekter har en formidabel oppgave fremover, når de skal sørge for at nettsteder er konsistente og behagelig å bruke. På et større nettsted er det umulig å vise frem all informasjon som er tilgjengelig på ett og samme sted. Hvordan blir informasjonen man publiserer tilgjengelig for målgruppen?

Det må sannsynligvis legges til rette for en eller flere muligheter for å søke. De fleste som er kjent med bruk av internett har vel utført en eller flere former for søk. Det finnes mange forskjellige måter å søke på. Noen vet akkurat hva de er ute etter. Noen vet sånn omtrent hva det de er ute etter handler om, og andre igjen er rett og slett bare ute og surfer for å se hva som dukker opp. Hvor trenger man søkemuligheter, og hvor trenger man det ikke?

Det kreves ressurser for å tilrettelegge for søk på et nettsted, og mindre nettsteder trenger kanskje ikke søk i det hele tatt. Andre større nettsteder igjen, legger kanskje ned flere årsverk for at nettstedet deres skal være godt søkbart. For at et større nettsted skal kunne lykkes, må det sannsynligvis legges til rette for mange typer av søk.

Hvordan gjør man det behagelig å søke etter informasjon? Noe av det viktigste er vel at det må gå ganske raskt å finne det man leter etter. For at et søk skal gå raskest mulig pleier man å indeksere de data man har tilgjengelig. En indeks er en datastruktur som kan brukes for å finne informasjon raskere. I denne artikkelen ser vi på litt av både det å søke etter, og det å indeksere data. Vi tar i tillegg for oss litt om søkemotoren Google, som de fleste av oss kjenner til.

Del II. Søk

1. Når er søking en hensiktsmessig form for navigasjon?

• Når det er så mye innhold at man ikke kan bla mellom sidene som finnes. Dersom brukerne kommer til sidene for å lete etter informasjon, og det er så mye innhold at det er for tungvint å bla mellom sidene for å finne det man er ute etter, er søk en nødvendighet[1].
• Når innholdet på en side er svært fragmentert. Mange nettsteder inneholder mye forskjellig informasjon som har vært opparbeidet fra forskjellige kilder innenfor en organisasjon på en lite planmessig måte. Her er ikke søk løsningen på alle problemer, men det kan være en måte å tilby brukerne informasjon uansett hvilket område i organisasjonen den kommer fra. Søkesystemet kan gi informasjonsarkitekten oversikt over hva som finnes i systemet, og analyse av logg fra søkesystemet kan gi en pekepinn om hvordan innholdet bør organiseres.
• Når det finnes tid og teknisk kunnskap til å optimere søkesystemet. Det er rimelig enkelt å sette opp en søkemotor, men det er vanskelig å sette opp en søkemotor med høy kvalitet. Mange sider har uforståelige grensesnitt for søking, og søkene returnerer irrelevante resultater. Hvis det ikke er vilje eller evne til å konfigurere søkemotoren ordentlig, bør beslutningen om å ha en søkefunksjon revurderes.
• Finnes det bedre alternativer? Søk kan være en bra måte å tilby brukerne informasjon, men ikke nødvendigvis den beste i alle situasjoner. Hvis det ikke finnes teknisk ekspertise eller penger til å konfigurere en søkemotor, kan et innholdsregister være et alternativ.

• Nå brukerne er interessert i å søke. Det kan hende det er innlysende at brukerne heller vil bla gjennom sidene enn å søke. Hvis du tilbyr bilder, er kanskje muligheten til å bla en bedre form for navigasjon for brukerne, eller kanskje søk er mindre viktig for brukerne. Da er det kanskje bedre å bruke utviklingsbudsjettet for informasjonsarkitekturen på en annen måte.

2. Hva bør det søkes i?

2.1. Søkesoner

En søkesone er en del av et nettsted som har blitt indeksert separat fra resten av innholdet på nettstedet. Når en bruker søker i en søkesone har han allerede sagt seg interessert i denne typen informasjon. Ideelt sett bør søkesonen på nettstedet passe til hans behov, og resultatet er at det er lettere å finne relevant informasjon, siden informasjon som er irrelevant for brukeren ikke er med i søket. Søkesoner kan organiseres etter blant annet:

• Innholdstype
• Målgruppe
• Rolle
• Emne
• Lokasjon
• Kronologi
• Forfatter
• Del av organisasjon

2.2. Navigasjon og destinasjon

De fleste nettsteder inneholder minst to typer sider: navigasjonssider og destinasjonssider. Destinasjonssidene inneholder det faktiske innholdet på nettstedet: sportsresultater, bokanmeldelser, programvaredokumentasjon osv. Navigasjonssider omfatter hovedsider, menysider, søkesider og sider som hjelper deg med å bla rundt i innholdet på et nettsted. Formålet med navigasjonssidene er å finne destinasjonssider. Når en bruker søker på et nettsted, er det rimelig å tro at den er ute etter destinasjonssider. Hvis navigasjonssidene er med i søkeresultatet, vil dette bare gjøre treffene i søket mindre relevant for brukeren.

2.3. Utvelgelse av innholdskomponenter for indeksering

På samme måte som det ofte er nyttig å tilby tilgang til deler av et nettsteds innhold, er det verdifullt å tilby brukerne å søke i deler av et dokument. På denne måten kan brukerne få mer spesifikke, presise resultat. Hvis dokumentene har administrative eller andre deler som ikke er særlig relevant for brukerne, kan disse utelates fra søket.
Deler det kan søkes i kan være:

• Artikkeltekst
• Dato
• Tittel
• URL
• Navn på nettsted
• Lenke
• Beskrivelse
• Nøkkelord

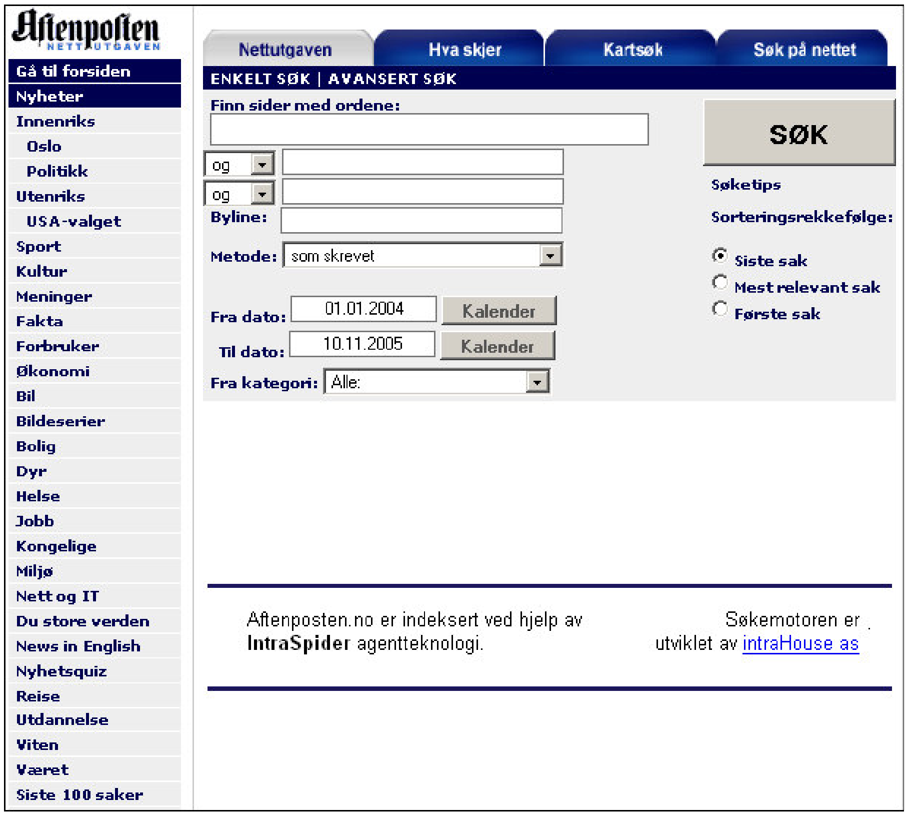
Disse kan brukes i søkegrensesnittet, der brukerne kan velge hvilke parametere de vil ta med i søket. Et problem kan være at selv om dette ville gitt brukerne mer relevante søkeresultater, er det ikke sikkert at brukerne er interessert i å bruke slike" avansert søktype funksjoner, eller ikke først år nytten de kan ha av det.
Avansert søk hos aftenposten.no (figur 1 på neste side) tilbyr brukerne søk i sidetekst, men også søk i byline, og muligheter for innsnevring av søkeresultater gjennom å velge en tidsperiode eller en kategori.

3. Søkealgoritmer

Søkemotorer blir brukt i mange forskjellige omgivelser, alt fra søkemotorer som kun søker etter informasjon på din egen PC, motorer som søker kun på små begrensede webområder og større søkemotorer som prøver å søke gjennom så store deler av internett som mulig. Felles for søkemotorene er at de bruker en søkealgoritme for å finne informasjonen det ble søkt etter.

Hva som gjør at f.eks. en side eller et dokument blir regnet som et treff varierer ut fra instillingene på søkemotoren. Noen algoritmer returnerer alle websider og dokumenter som et eller annet sted hadde samme tekststreng som det ble søkt etter. Dette er fulltekstsøk der alt innholdet i dokumentene er indeksert. Andre algoritmer kan begrense søket til f.eks. tittelen, filnavnet og/eller innholdet i metadatafeltet, alt avhengig av hvordan en eventuell indeks er oppbygd og hvilke data indeksen er basert på.

Mens enkle algoritmer ofte returnerer alle treff de kommer over og dermed får et stort antall treff, kan mer avanserte algoritmer begrense antall treff ved kun å returnere treff som de mener er mest mulig relevante for søket. De kan også utvide søket og søke etter lignende ord, synonymer, og/eller andre bøyninger av søkeordet og dermed få et betydelig antall ekstra treff. Om en søkemotor vil få flest mulig treff, eller færre, mer presise treff, er noe som må bli bestemt ut fra hva brukerne av søkemotoren er ute etter.



Finn sider med ordene:

og

og

Byline:

Metode: som skrevet

Fra dato: 01.01.2004

Til dato: 10.11.2005

Fra kategori: Alle:

Sorteringsrekkefølge: [✓ Siste sak] Mest relevant sak Første sak

Aftenposten.no er indeksert ved hjelp av **IntraSpider** agentteknologi.

Søkemotoren er utviklet av intraHouse as

Figur 1: Avansert søk hos aftenposten.no

4. Tekstgjennfinning

Med bakgrunn i et dokument fra kurset til Judith Molka Danielsen i in350 fra 2002[4]. Dette dokumentet er sterkt relatert til databaser, men vi finner at det som beskrives her er relevant for IA (Information Architecture) faget også.

Når en bruker søker etter tekst, bruker han flere metoder, selv om han kanskje ikke tenker over det selv.
• Begrenset søking
• Utforskende søking
• Tilfeldig søking.

4.1. Begrenset søking

Du vet at informasjonen du søker finnes, og du vet også hvordan denne informasjonen ser ut, før du faktisk finner den. Dette gjør at du vet at du har truffet riktig når søkeresultatet foreligger. Et eksempel kan være en søkestreng: "Når solen daler i vest, arbeider den late best". Dette søket vil returnere et dokument som inneholder denne tekst strengen. 100% presisjon krever at man beskriver det man vil ha 100% riktig. Dette krever bare et forholdsvis enkelt gjennfinningsverktøy.
Hvis man vil ha et smartere søk, der det returneres ord som har sammenheng med søkestrengen, må man bruke et mer avansert verktøy.

4.2. Utforskende søking

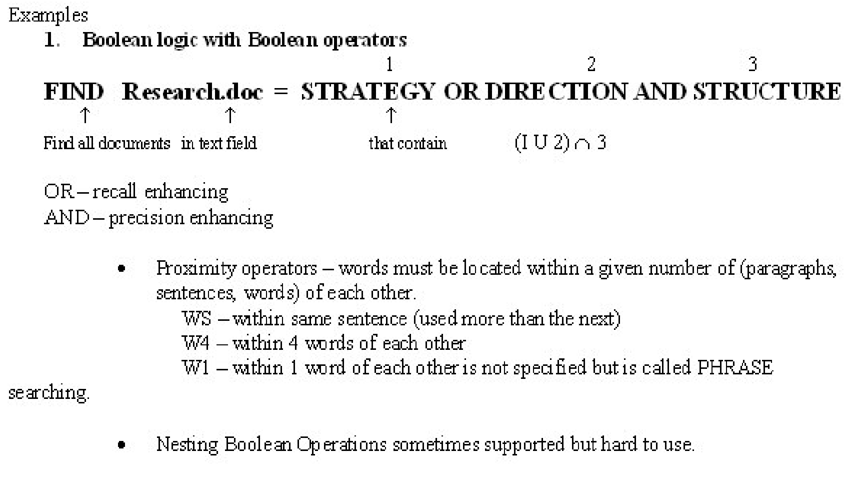
Brukeren har ingen kunnskap om dokumentene det skal søkes i. Han må finne ut om det kan finnes noe som er relevant for det han er ute etter. Dette blir en gjentagende prosess.
• Brukeren starter med en kategori (nøkkelord) f.eks. "Volleyball"
• Systemet returnerer emner. f.eks. "Norges Volleyballforbund", "USA Volleyball", "Volleyball Hall of Fame"
• Brukeren velger et emne. f.eks. "Volleyball Hall of Fame"
• Systemet returnerer en liste over dokumenter. f.eks. Dokumenter der man kan lese biografier til personer som er opplistet i "The Volleyball Hall of Fame"
Her er brukeren fortsatt ute etter en bestemt bit med informasjon. Brukeren har et bestemt mål når han starter søket, og søkeprosessen hjelper ham å finne det han er ute etter.

4.3. Tilfeldig søking

Brukeren har ikke noe spesielt mål med søkingen. Han bare klikker på lenker og hopper hit og dit mellom informasjon som tilsynelatende ikke har noe med hverandre å gjøre.

4.4. Spørringer for gjenfinning av tekst

• Boolean basert



Examples

1. **Boolean logic with Boolean operators**

```
                                  1                  2              3
FIND  Research.doc  =  STRATEGY OR DIRECTION AND STRUCTURE
  ↑              ↑              ↑
Find all documents  in text field          that contain        (I U 2) ∩ 3
```

OR – recall enhancing
AND – precision enhancing

• Proximity operators – words must be located within a given number of (paragraphs, sentences, words) of each other.
  WS – within same sentence (used more than the next)
  W4 – within 4 words of each other
  W1 – within 1 word of each other is not specified but is called PHRASE searching.

• Nesting Boolean Operations sometimes supported but hard to use.

Figur 2: Eksempel på boolsk spørring. Kilde: [4]

• Spørring basert på utfylling av skjema
– Sette søkeord inn i tekstbokser i et skjema. Søkeordene representerer felter i databasen. Et eksempel kan være at du designer et skjema og kobler dette til en tabell. Tekstboksene i skjemaet tilsvarer kolonnene i tabellen, og du f år treff når du skriver ord som finnes i kolonnene inn i tekstboksen.

– Sette spørringer i tekstbokser i et skjema. For eksempel SQL[9]. Et skjema kan kobles til en database, og du skriver SQL direkte i tekstboksen for å hente ut data fra databasen.
– Konvertere ordene i et skjema om til en spørring. Dette ligner på det første eksempelet, men man kan se for seg at skjemaet er koblet til en database, ikke bare en tabell, at spørringen er låst og at ordene som skrives inn behandles som variabler.
• Spørring basert på eksempel
– Du vet ikke hva du er ute etter, men du har et emne du ønsker å finne mer om.
– Det kan betegnes som interaktiv søking.
– En spørring gir et svar som fører til at du endrer ditt mål for søket.
– Eksempel: Man kan se for seg at man jobber med utvikling og vil implementere en form for meldingsbasert kommunikasjon. Man jobber i sin favoritt utviklingsmiljø og bruker Google og søker på "developing a message based system". Noe av det som returneres viser seg å være informasjon om hvordan man implementerer et meldings basert system i et annet utviklingsmiljø enn det du jobber i nå. Neste søk kan da bli etter informasjon om hvordan man setter opp dette andre utviklingsmiljøet.
• Spørringer med naturlig språk
– For eksempel: "Gullmedalje vinnere fra OL i 1992".
– Du skriver en spørring som om du snakket til en annen person.
– Kan gi uforutsigbare resultater.
– Finner likevel som oftest mer relevant informasjon på kortere tid, en for eksempel en boolsk spørring, som er presis, men vil utelate andre resultater som kan være relevant for søket.

4.5. Teknologier for tekst gjenfinning

• Applikasjonen brukes på en maskin
• Applikasjonen kjøres på stormaskin, feks AS400 [10]
• Nettverk. LAN basert klient-server arkitektur. Prosessering og indeksering gjøres på server. Klient viser resultat av søk.

5. Stopwords

I alle språk er det flere vanlige ord og uttrykk som går igjen og som gir lite informasjon om innholdet i dokumentet eller websiden. Disse vanlige ordene kalles stopwords. Et lite utdrag med eksempler på stopwords er and, or, the, eller, i, du, det, på og ikke. Hvilke ord som defineres som stopwords er det opp til søkemotoren å bestemme. Standarden hos søkemotorene når slike ord er med i en søkestreng er at de ikke tas med i søket. Mange søkemotorer velger å la være å indeksere stopwords fordi disse ordene krever veldig mye plass og ressurser. F.eks. i et engelsk dokument på ca. 170'000 ord, som inneholdt 15'370 unike ord, utgjorde de 26 vanligste ordene hele 33% av det totale antall ord i dokumentet! Ordet "the" utgjorde alene hele 5% av alle ordene i dokumentet! [6]

Større søkemotorer som Google har valgt å indeksere også disse ordene, men som standard utelater også Google disse ordene fra vanlige søkestrenger. Bruker du derimot hermetegn for å markere at søkestrengen er f.eks. et uttrykk, så vil søket også inkludere stopwords.

6. Måling av kvaliteten på søkealgoritmer

Kvaliteten på en søkealgoritme kan måles i presisjon og recall. For å finne presisjonen til søket har du formelen:
Presisjon = Antall relevante dokumenter returnert / Antall returnerte dokument
For å finne recall til søket er formelen:
Recall = Antall relevante dokumenter returnert / Totalt antall relevante dokumenter (kilde: [11])
Recall måler hvor bra søkesystemet finner det du vil ha, og presisjon måler hvor bra systemet siler ut det du ikke vil ha. Et eksempel: du skal søke etter dokumenter om BibTeX blant 20 millioner dokumenter. Blant disse dokumentene finnes det 2000 dokumenter som er relevant for deg. Søket ditt returnerer 1600 dokumenter, 1000 av disse er relevant for deg. Du har funnet 1000/2000 av de relevante dokumentene, så recall er 50%. 1000 av de 1600 dokumentene var relevante, så presisjonen er 62,5%. Disse størrelsene kan være vanskelige å måle – i en sammenheng der en søkemotor er

nødvendig, kan det nok være vanskelig å telle treffene, samtidig er relevans et svært subjektivt begrep[2].

7. Utfordringer ved søk: hvordan finne relevante treff?

De fleste søkemotorer tilbyr avansert søk, men i praksis viser det seg at f å brukere faktisk benytter seg av de mulighetene avanserte søk tilbyr (mindre enn 0,5 prosent av brukerne[2]). Her presenterer vi kort to scenarier der brukeren mister relevante treff, og muligheter for forbedringer av søket i disse tilfellene. Vi ser også at løsningsforslagene nok i mange tilfeller vil føre til at recall forbedres (vi får flere treff), mens presisjonen forverres (vi får flere irrelevante treff).

7.1. Metadata

Metadata blir ofte beskrevet som data om data, noe som ikke kan sies å være en særlig god beskrivelse. En bedre beskrivelse er at det er informasjon om, eller dokumentasjon om annen data[1]. Forfatter, størrelse, plassering og nøkkelord om innholdet til et dokument er eksempler på metadata. Brukt rett kan metadata være til stor hjelp innen søking, men for websøking har det vist seg at all tekst, som ikke er synlig, som eksisterer på en webside, f.eks. innholdet i metadata-feltet, blir brukt for å manipulere søkemotorer. For å bekjempe dette setter søkemotorene en grense på ca. 25 nøkkelord i metadata-feltet. Dersom det er overdreven bruk av nøkkelord, eller enkeltord blir gjentatt for mange ganger, kan søkemotoren ignorere metadataen, eller til og med hele siden[5].

7.2. Brukeren bruker andre begreper enn de som finnes i teksten

Ofte vil en bruker spørre om et begrep, mens et annet begrep er brukt i teksten. Et mulig tiltak for å finne flere relevante treff kan være å ha en thesaurus eller synonymringer knyttet til søkemotoren. På denne måten kan søk også vise resultater for ord med samme eller nærliggende betydning. På den andre siden har mange synonymer en lignende, men distinkt annerledes betydning, og fullstendig irrelevante treff kan dukke opp blant søkeresultatene[20].

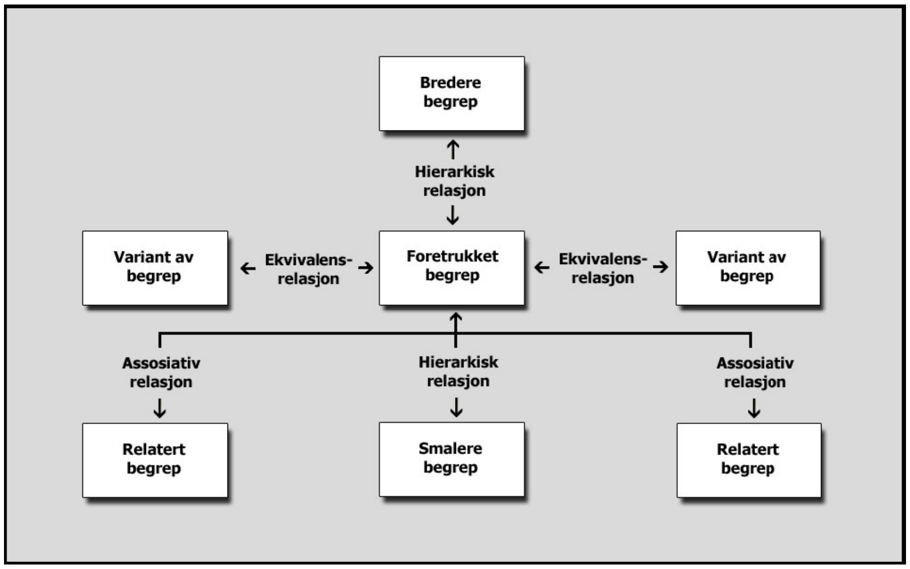
7.2.1. Thesaurus

En thesaurus er en synonymordbok som også inneholder assosierte begreper, og kanskje til ord med akkurat motsatt betydning[1]. I en thesaurus kan vi definere emantiske relasjoner mellom ulike ord og begreper (se fig. 3 på neste side). Hvert foretrukne begrep blir senter for et semantisk nettverk. Ekvivalensrelasjoner tilsvarer synonymer. Hierarkiske relasjoner gjør det mulig å dele begreper inn i under- og overkategorier. Assosiative relasjoner tilbyr andre forbindelser mellom ord.

Det er vanskelig å finne eksempler på bruk av thesauri på nettsteder – dette kan ha sammenheng med at det er noe som skjer "bak kulissene", og som vi ikke legger merke til. Noen steder kan vi se at det gis forslag til nærliggende begreper, noe slikt som "se også", forslag til andre begreper man kan søke etter. Det er forventet at thesauri vil bli viktigere etter hvert som flere nettsteder blir større og viktigere[1].

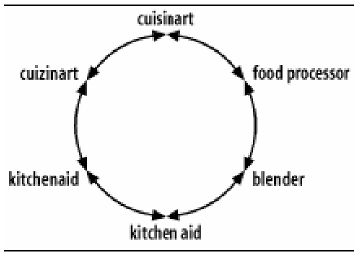
7.2.2. Synonymringer

Synonymringer (se figur 4 på neste side) forbinder et sett med ord med ekvivalent betydning - synonymer. I praksis har ofte synonymer større eller mindre forskjeller. Dette kan gi søkeresultater som forvirrer brukeren, og som forbedrer recall for søket.



Figur 3: Semantiske relasjoner i en thesaurus. Kilde: [1].

Dersom synonymringer brukes, bør dette taes hensyn til når brukergrensesnittet designes. Søk med treff for eksakt det ordet brukeren skrev bør komme først, eller synonymringer bør i utgangspunktet ikke brukes, men brukerne kan ha muligheten til å utvide søket til å også ta med relaterte begreper.



Figur 4: Synonymring. Kilde: [1].

7.3. Brukeren bruker begreper som finnes i teksten, men i en annen form

Dersom brukeren søker etter "Computer", vil treff som "Compute" og "Computational" forsvinne. Dette kan motvirkes ved "stemming", det å kappe ordet ned til en "kjerne", og søke etter denne. Dette krever en viss språklig (morfologisk) forståelse i implementeringen av algoritmen, noe som er vanskelig å utføre i praksis, og kan føre til forverring av presisjonen uten noen vesentlig forbedring av recall.

Del III. Indeksering

8. Hva er indeksering?

Indeksering brukes i mange sammenhenger. Det kan for eksempel brukes for å tilrettelegge for søk etter dokumenter på et nettverk. Indeks benyttes også i programmeringsspråk, blant annet når man skal velge ut et objekt i et array av objekter. I et array finner man ofte det første objektet på indeks nr. 0, det neste på indeks nr. 1 osv... Hvorfor bør man indeksere? Man indekserer for raskere å kunne finne den informasjonen man søker etter. En indeks[16] er en datastruktur som forbedrer ytelsen ved søking. Analysering av datastrukturer for indeksering er en stor del av hverdagen til mange informatikere.

Ting å tenke på her er søkeytelse, størrelsen på indeksen og hvor mye maskinressurser som brukes på å oppdatere indeksen. Hvis for eksempel et datalager inneholder (N) dataobjekter, vil en dårlig indeks, eller ingen indeks gjøre at man måtte ta for seg hvert objekt i rekkefølge, helt til man finner det man er ute etter. Dette fører til at man i verste fall må søke gjennom (N) objekter (hvis det er det siste objektet man er ute etter), og at man gjennomsnittlig må søke gjennom (N/2) objekter, alts å et søk av orden O(N). Ettersom datalager ofte inneholder mange objekter, er det ønskelig å gjøre denne søkingen mer effektiv. Relasjonsdatabaser inneholder indekseringsteknologi for å øke ytelsen ved oppslag i tabeller i databasen.

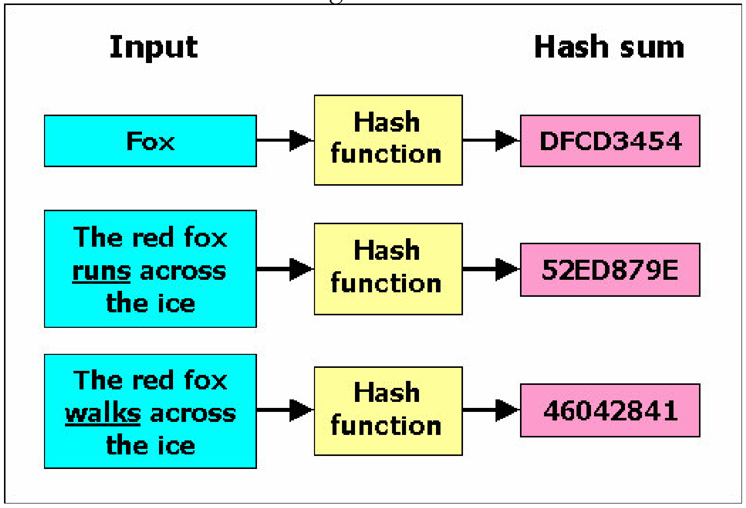
En spesifikk og veldig vanlig bruk av indekserings teknologi er i informasjonsgjennfinnings systemer. Der brukes ofte en full-tekst indeks for å raskt kunne identifisere dokumenter basert på deres tekstlige innhold.

9. Implementering av indekser

Vi skal her ta kort for oss mekanismene bak noen av de vanligste formene for indekser.

9.1. Hash-basert indeks

Hash-tabeller burde være kjent for de som har tatt videregående programmeringskurs. Ved hashing regnes det ut en numerisk kode basert på verdien man hasher på. Denne verdien bør ha så stor spredning som mulig, det vil si at færrest mulig input-verdier skal gi samme hashkode. I figur 5 på neste side ser vi hvordan strenger omdannes til en hashkode vha. hashfunksjonen. Legg merke til at hashfunksjonen forandres helt selv om strengen bare forandres litt.



Figur 5: Hash-funksjon i bruk, en streng blir til et 32-bits tall. Kilde: [15]

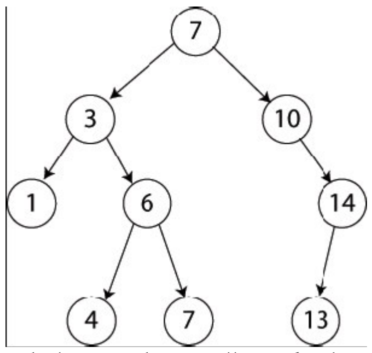
Ved innsetting av et element i en tabell, kan vi bruke hashkoden for å bestemme hvilken plass elementet skal settes på. Senere kan elementet raskt finnes igjen ved hjelp av å regne ut hashkoden. Dette gjør at både innsetting og gjennfinning av elementet kan foregå med bare en operasjon - O(1), vi slipper å gjøre noen søking i det hele tatt. Men det er problemer med hashtabeller. Flere element kan bli tilordnet samme hashkode, da må vi søke for å finne en ledig plass til elementet ved innsetting, og søke for å finne igjen elementet ved henting av det. For å unngå at dette skjer i for stor grad, må hashtabellen ikke ha for stor lastfaktor[13], dette vil si at det må tilordnes mer plass til tabellen enn det den egentlig bruker - i praksis minst dobbelt så stor plass (!). Så hashtabeller kan være svært effektive hvis vi har nok plass i primærlager til alle elementene vi har tenkt å ha i hashtabellen, men hvis datamengden er så stor at diskaksesser kan være nødvendig, trengs det en annen datastruktur.

9.2. Tre-basert indeks

Et tre (se figur 6 på neste side) er en datastruktur som etterligner en trestruktur vha. noder med lenker til hverandre. Hver node har null eller flere barnenoder nedenfor seg i treet[17].

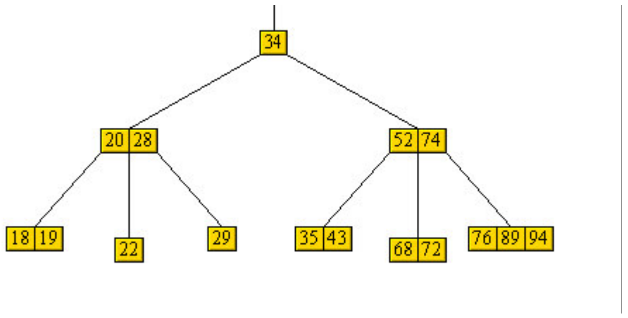
9.2.1. Balanserte trær

Den vanligste formen for indekser i database management system (DBMS) er balanserte trær. I en slik datastruktur sørges det for at det er et jevnt antall noder på hver side av (rot)-nodene i treet. Dette krever en viss bruk av ressurser



Figur 6: Binærtre, en enkel trestruktur - alle noder har lavere verdier enn sin egen verdi p°a venstre side og høyere verdier p°a høyre side. Kilde: [17].

til vedlikeholding av indeksen for å holde treet balansert, men gjør at tiden for innsetting og uthenting holdes noenlunde konstant. Eksempler på slike trær er B-trær (se figur 7). I et B-tre inneholder hver node et gitt antall elementer – dette gjør B-trær til gunstige datastrukturer når data må hentes fra sekundært lager (disk), siden antall elementer kan tilpasses til så mange som det er plass til i en en diskblokk [14].



Figur 7: B-tre, her vist med maksimum fire verdier pr. node. Kilde: [14].

10. Strategier for bruk av indekser

10.1. Lager og indeksering

• BLOB (Binary Large Object) – Lagrer hele dokumentet binært i databasen [7]

• Lagre bare en peker til dokumentet på disk.

• Bruke SGML[8] til å opprette et hybrid system for dokumentbehandling som er både strukturert og ustrukturert. I systemet brukes tagger for å tillate full-tekst søk til deler av dokumentet. For eksempel til overskrifter.

10.2. Indekserings metoder

Eksempel på en invertert indeks:

• Invertert Indeks

Alle ord i alle dokumenter legges sortert inn i indeksen. Stopwords[6] tas ikke med. Indeksen holder rede på forekomst og lokasjon til ordet

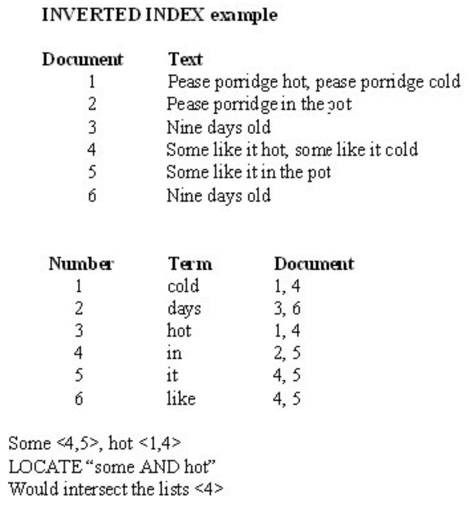
• Posisjonell invertert indeks

Den eksakte lokasjonen til hvert eneste ord holdes rede på. Dette gir raske spørringer, men størrelsen på indeksen kan bli veldig stor og den må bygges på nytt hver gang teksten redigeres.

• Ikke posisjonell invertert indeks.

Peker på den første forekomsten av et ord i et dokument. Vet antall forekomster av ordet. Vet ikke om lokasjonen til ordet. Dette fører til en mindre indeks, men også tregere spørringer.

• Mønster gjenkjenning
Ordene konverteres til en binær representasjon og algoritmer brukes for å beregne et mønster i ordet. Disse mønstrene legges i indeksen sammen med pekere til dette mønsteret sin lokasjon i dokumentet. Dette systemet støtter spørringer med naturlig språk. Denne metoden har egenskaper som raske svar på spørringer og en liten indeks. Metoden gir også treff på lignende mønster, slik at eventuelle stavefeil i søket likevel kan gi treff. Bruker fritt søk i teksten som neste alternativ for å oppnå eksakte treff. Siden denne søkemetoden jobber binært kan den også brukes for å søke etter lyd, bilde og video.
• Fritt søk i teksten
Sekundær søkemetode som brukes sammen med ikke posisjonell invertert indeks og mønster gjenkjenning. Denne metoden har ikke indeks.


INVERTED INDEX example

Document	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

Number	Term	Document
1	cold	1, 4
2	days	3, 6
3	hot	1, 4
4	in	2, 5
5	it	4, 5
6	like	4, 5

Some <4,5>, hot <1,4>
LOCATE "some AND hot"
Would intersect the lists <4>

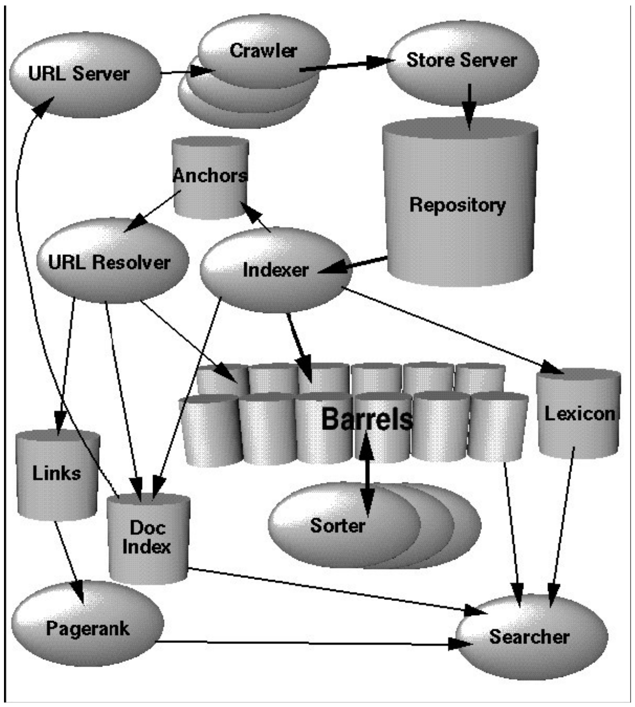
Figur 8: Eksempel på en invertert indeks, Kilde: [4].

Det utføres bare et sekvensielt søk. To positive egenskaper er at det man ser alltid er oppdatert, og at det ikke finnes en indeks å vedlikeholde. Søkemetoden taper seg i ytelse når antall dokumenter og samtidige brukere øker. Brukes når man vil ha eksakte treff på ord.
• Hypertekst
Ikke basert på spørringer. Dokumenter er lenket sammen. Dette gjør at man kan bla gjennom en samling av dokumenter. Et hypertekst system benytter seg ofte av søkemetodene som er beskrevet ovenfor.

Del IV. Google - Søking, sortering og indeksering i praksis

Google er en mye brukt og populær tjeneste for å finne informasjon på nettet. Man kan bruke google for å søke i forskjellige kategorier, som for eksempel vanlig søk etter dokumenter på Internett, søk etter bilder, søk i nyhetsgrupper og så videre. Utviklerene av Google har valgt navnet fordi det henspeiler på "googol", en betegnelse på tallet 10100, som passer med målet deres – å bygge søkemotorer med stor målestokk.

I den daglige bruken av søkemotorer er det vel de færreste som tenker over kompleksiteten i systemet som presenterer resultater for søkeren. Her skal vi forsøke å forklare noe av det som er implementert i Google, med bakgrunn i deres egen artikkel[3] fra noen år tilbake.



Figur 9: Oversikt over Googles arkitektur: [3].

11. Arkitekturen

Når Google skal laste ned web-sider bruker de flere "Crawlers"[18]. Disse Crawlerne f år tilsendt en liste med Uniform Recource Locator (URL)'er fra en URL server. Crawlers, eller roboter som det også kalles, får ofte regler å forholde seg til i form av en tekstfil, f.eks. "robots.txt", på nettstedene de besøker. Disse reglene setter en ramme for hva robotene får lov å berøre på nettstedene. De sidene som blir lastet ned sendes til en annen server for lagring. De kaller denne serveren for "Store Server". Store Server komprimerer og lagrer sidene i "Repository". Repository kan oversettes med oppbevaringssted. Hver eneste side har en id assosiert med seg, og denne id'en kalles docId. To andre deler av arkitekturen er "Indexer" og "Sorter". Google sier ikke hvordan Indexer og Sorter er implementert, men vi går ut i fra at det er tjenester som kjører hver for seg på en eller flere servere. Vi først år ut fra navnet at Indexer utfører indeksering, og at Sorter sorterer den ferdige indeksen. Indexer gjør flere operasjoner når den indekserer. Den leser oppbevaringsstedet, pakker opp dokumentene fra komprimert tilstand og tolker dem. Tolke vil her si å dra ut alle ordene som dokumentet består av. Hvert dokument konverteres på denne måten til et sett, eller en samling med ord. Denne samlingen kalles "hits". Denne samlingen av ord tar selv vare på en god del informasjon:

• Selve ordet
• Ordets plassering i dokumentet
• Informasjon om skriftstørrelse
• Registrerer om ordet er skrevet i store eller små bokstaver

Dette tas vare på for at det skal brukes i rangering av sider (PageRank[12]) I tillegg til PageRank rangerer Google sider basert på hvor ord befinner seg i dokumentet og hvordan det er formatert.
12. Internett blir større... ...og det blir også antallet brukere som mangler erfaring med å utforske www[19].

Bruken av automatiserte søkemotorer som baserer seg på nøkkelord alene fører ofte til at det returneres mange svar som ikke er relevant for det brukeren egentlig er ute etter.

Utviklerne av Google hadde som mål å adressere mange av problemene som oppstår ved å utvide søkemotorenes målestokk, slik at de kan takle det økende antallet av tilgjengelige dokumenter, og også det økende antallet forespørsler. Å skalere i takt med www er en utfordring. Lagringsplass må utnyttes effektivt for å kunne lagre store indekser og mange dokumenter. Indekserings systemet må indeksere store mengder data på en effektiv måte. Spørringer må også håndteres rasket mulig, helst i størrelses orden 103/sec.

Disse oppgavene har vist seg vanskelig å overkomme, men økt maskinvare ytelse og lavere maskinvare kostnad har gjort at noen av problemene ikke har blitt uoverkommelig store, like raskt som forventet. To områder der økt ytelse ikke har vokst i takt med andre områder er harddiskenes søketid, og operativ systemer som er robuste nok.

13. Kvantitet og kvalitet på treff

Et stort problem med at det returneres mange treff på et søk, er at det du egentlig er ute etter forsvinner i en mengde av ikke-relevante treff. I november 1997 var det bare en av de fire største søkemotorene som greide å returnere seg selv blant de ti første treffene på søket. Det viser seg at en vanlig bruker går lei etter at han har prøvd de ti første treffene som returneres som svar på søket. Etter det kjører han et nytt søk, og prøver de ti første treffene der, og så videre. Dette gjør at Google har brukt ressurser på å øke presisjonen til søkemotoren sin. De beregner presisjonen ut fra antall relevante dokumenter returnert blant de 10 første treffene på søket. De mener at denne presisjonen er viktigere enn recall. De ser videre muligheter for å utnytte lenke strukturen i www, og teksten i lenken, for å kunne øke kvaliteten til søkemotoren. Google benytter ikke synonymringer eller stemming for å øke antall treff. Dette betyr i praksis at et søk etter f.eks. "bil", vil returnere kun sider som inneholder ordet "bil", og ikke "bilen" eller "biler". Synonymer kan selvfølgelig forekomme i sidene som returneres, men dette er ikke søkemotorens fortjeneste. Det er bare dokumentet som inneholder ordet "bil" som er skrevet slik.

14. Siderangering

Google bestemte seg for å legge stor vekt på lenker for å finne ut hvor relevant en side er. Dette har vært lite, eller ikke brukt i hele tatt, av tidligere søkemotorer. Google teller for eksempel antall lenker som refererer til en spesiell side. Dette gir en tilnærming til hvor viktig, eller av hvor god kvalitet, siden er. Google sin PageRank tar rangeringen ett steg videre ved å ikke verdsette lenkene fra alle sider likt. Hvis en side med høy PageRank lenker til en gitt side, blir denne lenkingen verdsatt mer enn hvis en side med lav PageRank lenker til den samme siden.

15. Lenketeksten

Google assosierer lenketeksten med siden lenken er på, og i tilleg med siden lenken peker til. De mener også at anker kan ha en mer nøyaktig beskrivelse av en side, enn siden selv har. De ser verdi i at anker kan brukes på annet en tekst, slik som bilder, programmer og databaser. Dette gjør at de kan returnere sider som ikke har blitt prosessert av google sine web-crawlere[18], og dermed øke antall treff uten større belastning på systemet. Andre ting som Google benytter i utviklingen av sin søkemotor er blant annet:

• Lokaliserings informasjon for alle treff
• De tar hensyn til font-størrelsen som er brukt i ord
• Ord som er skrevet i fetere eller større font verdsettes mer enn andre ord

Det siste av punktene ovenfor gjelder pr. dokument, så et dokument som er skrevet med stor skrift blir ikke verdsatt mer enn et dokument som er skrevet med liten skrift.

16. Konklusjon

Vi har forsøkt å trekke opp noen retningslinjer for når det er hensiktsmessig med søking og hvordan dette kan implementeres på en måte som gjør brukeren i stand til å finne data som er relevant for denne. Så har vi sett på forskjellige former for søk, og størrelser for å måle kvaliteten på et søkeresultat - de negativt korrelerte måltallene presisjon og recall. Disse er vanskelige å måle, men kan være nyttige for å diskutere kvaliteten på en søkealgoritme. Vi har også sett på metoder for å

forbedre søkeresultater, samt fordeler og ulemper ved disse. Vi har prøvd å vise hvorfor indekser er nødvendige, og sett litt overfladisk på to vanlige typer indekser.
Til slutt har vi sett på Google, og noe av teknologien som ligger bak en søkemotor i stor skala.

Referanser

[1] Information Architecture For The World Wide Web - Designing Large-Scale Web Sites, 2nd Ed. O'Reilly, 2002.
[2] Tim Bray. On search (essayserie). Technical report, http://www.tbray.org/ongoing/When/200x/2003/07/30/OnSearchTOC , 2003.
[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. Technical report, Stanford University - http://www.db.stanford.edu/backrub/google.html , 2000.
[4] Judith Molka Danielsen. Types of indicies. Technical report, http://home.himolde.no/~molka/in350/class5b.doc , 2002.
[5] Diverse/ukjent. Search engines. Technical report, http://www.webreference.com/content/search/how.html , 1998.
[6] Diverse/ukjent. On search: Stopwords. Technical report, http://www.tbray.org/ongoing/When/200x/2003/07/11/Stopwords , 2003.
[7] Diverse/ukjent. Wiki: Blob. Technical report, http://en.wikipedia.org/wiki/ Binary large object, ukjent.
[8] Diverse/ukjent. Wiki: Sgml. Technical report, http://en.wikipedia.org/wiki/SGML , ukjent.
[9] Diverse/ukjent. Wiki: Sql. Technical report, http://en.wikipedia.org/wiki/SQL , ukjent.
[10] Diverse/ukjent. Wikipedia: As400. Technical report, http://en.wikipedia.org/wiki/ISeries , ukjent.
[11] The Regents of the University of California. Relevance, recall, precision, fallout. Technical report, http://www.law.berkeley.edu/library/classes/alr/boolean.html , 2000.
[12] Sergey; Motwani Rajeev; Winograd Terry Page, Lawrence; Brin. The pagerank citation ranking: Bringing order to the web. Technical report, http://dbpubs.stanford.edu:8090/pub/showDoc.Fulltext?lang=en&doc=1999-66&format=pdf&compression=&name=1999-66.pdf, 1999.
[13] Mark Allen Weiss. Data structures and algorithm analysis in java. Technical report, Addison Wesley Longman, 1999.
[14] Diverse/ukjent (wiki). Wikipedia: B-tree. Technical report, http://en.wikipedia.org/wiki/B-tree , Ukjent.
[15] Diverse/ukjent (wiki). Wikipedia: Hash function. Technical report, http://en.wikipedia.org/wiki/Hash , Ukjent.
[16] Diverse/ukjent (wiki). Wikipedia: Information technology. Technical report, http://en.wikipedia.org/wiki/Index , Ukjent.
[17] Diverse/ukjent (wiki). Wikipedia: Tree data structure. Technical report, http://en.wikipedia.org/wiki/Tree data structure , Ukjent.
[18] Diverse/ukjent (wiki). Wikipedia: www. Technical report, http://en.wikipedia.org/wiki/Web crawler, ukjent.
[19] Diverse/ukjent (wiki). Wikipedia: www. Technical report, http://en.wikipedia.org/wiki/Www , ukjent.
[20] W.A. Woods. Searching versus finding: Why systems need knowledge to find what you really want. Technical report, Sun Microsystems Laboratories, ukjent.

2. INF770: RELIABILITY FROM A USER PERSPECTIVE

AV ARTHUR SUND

1. On Reliability Issues

The purpose of this exercise is to discuss some reliability issues related to e-commerce applications such as amazon.com and ryanair.com. These services are primarily business to consumer (B2C) and offers standard products to a substantial discount compared to the traditional sales channels due to reduced unit costs and economies of scale. Our objective is to base these discussions on a user perspective.

There is a great difference between the structure of the Internet and the more reliable telecom systems. Telecom networks have implemented the logic into the network, but have only dumb terminals at the ends. The Internet, on the other hand, has intelligent terminals at the ends, but has very little logic in-between. All Internet services are thus based on an end to end argument. Performance issues also justify not handling reliability at every hop, but favor the concept of building such mechanism in the ends of the communication links. TCP is the main control mechanism that ensures retransmission of lost packets from end to end. However, it does not provide any form of reliability guarantees. TCP cannot set up multiple (backup) routes between source and destination, making it a serial system. Routers may fail, and it is neither a swift or reliable process for TCP to detect and repair failures. This is based on timeouts. In the mean time the service will be unavailable and the problem is sometimes un-repairable by rerouting.

Since online services must communicate over this relatively unreliable and uncontrollable communication link, it is impossible to build critical systems to communicate over the Internet. Amazon and RyanAir are however not critical systems. There are no potential for catastrophically natured incidents if one cannot buy a book or a plane ticket within a timely fashion. Thus, unavailability for these services has limited consequences from a user perspective (though there may be severe economical consequences for the companies).

A customer at a web shop has very little basis for assessing the situation if the service is to appear unavailable. If the problem does not lie on the Internet, the problem could be local on the customer's computer. This may be virus, browser incompatibility, lack of a plug-in (java, flash etc.), firewall, IDS/IPS software, bad Internet settings or perhaps problems with NIC or drivers. The problems may also lie in the server end. Applications or databases could fail.

This part is what system developers focus on avoiding with the use of reliability and fault tolerance techniques like WS_RELIABILITY, replication and fail-over solutions. When it comes to performance, a user on a web service of that he is not dependant will become inattentive and annoyed when experiencing response times of more than a few seconds. If the response times for each click exceed 10 seconds, he may navigate to a competing service or drop the order altogether. Related to this matter is the graphical interface of the web service. What is noticeable on RyanAir and Amazon is the simple layout and limited unnecessary graphics. This substantiate fast response times independent of Internet connections, makes the service less dependent on Internet backbone capacity and limits server loads. Amazon.com is a world-wide service. Therefore it is vital that the user experience is somehow equivalent on a Norwegian broadband connection and a Malaysian modem. In addition, they run a service with user groups in all time zones. With regards to distribution of load, this could be an advantage. Nevertheless, problems may very well occur outside the limits of American business hours, making it more difficult and expensive to repair. From a user perspective, you should be able to expect the same service level all over the world at all times.

The purchase process on web consists roughly speaking of three stages. This is selection of products (and vendor), the actual purchase and delivery. The web shop must facilitate availability and performance in all these stages. For the selection stage, this means that one should be able to browse a catalogue of available products at all times with a limited latency. For the purchase and delivery stage, an important element of security also comes into consideration. Regarding the purchase, there should be some form of concealment throughout the transaction. RyanAir uses for this matter the HTTPS protocol with 128 bits encryption. This also involves a third-party issued certificate to authenticate the web server.

Authentication is an important subject in this context, since IP itself does not identify transmitter or receiver. To establish some form of increased trust, the web server therefore presents a certificate issued by a third-party that both parties trust.

Privacy is another security related topic. When you hand over personal information you must have some guarantee that this information will be protected and handled confidentially. In most civilized countries the customers' interests in this area is legally protected. Security of delivery means that the customers get what they ordered, according to the specifications, within a reasonable timeframe. RyanAir uses electronic tickets to considerably reduce distribution and handling costs and makes it possible for instant delivery of the ticket. The transaction is however not completed until you have made use of the ticket for the actual flight service and got where you wished to go on time. For Amazon, most products are shipped by mail or some other parcel service. The delivery is therefore made by a third-party outside Amazon's control, making it as unreliable as the Internet. Still, most packages reach its destination.

In general, an important security issue is security against intentional threats. This includes protecting the server application and databases from intruders. That is, data integrity must be protected and theft of personal or business critical information must be avoided.

2. On Scalability/Load Balancing Issues

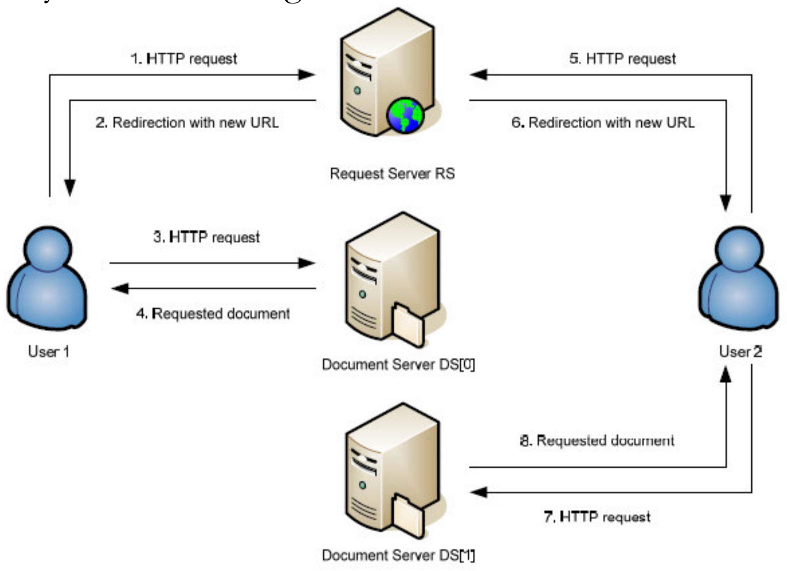


Figure 1. The System Architecture

The request server will function as follows:
1. On program start-up:
    a. An array DS of addresses for document servers and an integer k = 0 are
initiated.
2. On request:
    a. Return status code 302 with URL found in array: DS[k % DS.Size].
    b. Increment k.

The document servers will operate as standard web servers sending the requested documents or error codes. This simple solution is coded in java as a servlet. It fulfils the requirements in the assignment, distributing the HTTP requests evenly among three servers (actually two, since misi.no and itsikkerhetsportal. no is the same server). By addressing the localhost request server e.g. http://localhost:8081/Cluster/RequestServer?document=thedocument.doc I am redirected to one of the document servers. This solution is easily scalable to n servers by expanding the server array and distributing the content onto more servers.

However, some improvements could evidently be made to get a more reliable system. First of all, the servers should be pinged at start-up and at intervals to reduce the chance of redirecting to a dead server. Dead servers should be removed from the server array until they are back online. In addition, one might poll the document servers to distribute the load more evenly. Distributing the requests evenly does not necessarily implicate an evenly distributed load.

Reliability wise, the most obvious flaw is the single point of failure represented by the request server. If this is to become unavailable, the whole cluster breaks down. Such architecture would also be ideal for DoS attacks. It could be a good idea to implement some sort of failover solution for this server. When it comes to a case of distributing dynamic content the complexity increases dramatically. Dynamic behaviour causes problems for the replicas with respect to updates, failures during update, confliction updates (master, slave) and network problems (transient failures). Techniques for guaranteeing consistency must be applied. Using transactions and locking the database/documents during write or update will contribute to increased consistency on one database. If a transaction is aborted, the database content is reset to the state before the failed transaction was initiated.
Use of replicated servers and caching gives in addition major problems related to synchronization of the content. A user must be able to trust that he gets the same documents that everyone else. It is a major challenge to develop reliable systems for synchronizing replicated databases.

3. The code *RequestsServer.java*

```
package cluster;

import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestServer extends HttpServlet
{
   int k = 0;
   String[] DS = {"http://www.misi.no", "http://home.himolde.no/~010453/misi", "http://it-
sikkerhetsportal.no"};

   /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
    * @param request servlet request
```

```
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        if (request.getParameter("document") != null)
        {
            response.sendRedirect(DS[k % DS.length] + "/" + request.getParameter("document"));
            k++;
        }
        else
        {
            response.sendRedirect(DS[k % DS.length]);
            k++;
            //Redirection to index file if document is not spesified or syntax error. Or one cound give an error
code:
            //response.sendError(response.SC_NOT_FOUND, "Wrong format in request.");
        }
        out.close();
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit
the code.">
    /** Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        processRequest(request, response);
    }

    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

ABSTRACT

I decided to look into LinkedIn when the project was presented to me and wanted to see if it worked as well as people claimed. I wanted to check the %-number of introductions being forwarded, but since I lacked the relevant contacts on my profile and had agreed with a friend (Raymond Hagen) not to change anything in his profile, I decided to test how many I could get to sign up and use that service instead. The result was surprising to me, as not a single one of the ten test-subjects I decided on started using LinkedIn. This paper has given me a deeper insight into LinkedIn, and the different way it's being used. It also showed me that it takes time before the popularity of the online professional networks reach Norway and the students in higher education here.

Introduction:

In 2002 Entrepreneur Jonathan Abrams founded a social networking service called Friendster. The idea was to "make the world a smaller place by bringing the power of social networking to every aspect of life, one friend at a time. " In other words, it is a dating service where you can upload your own profile, picture and interests and start looking for a girl/boy with matching interests and build a relationship. Reid Hoffman was at this time working as Executive Vice President of PayPal and when he learned about Friendsters concept, he wanted to take that idea and remove the dating concept and instead add professional contacts to the mix. The result was launched in May 2003 and its name, LinkedIn, quickly found its way around the globe. Today, 2 ½ years later, LinkedIn has over 4 million registered professionals, a number that is four times higher then all the other online business networks combined.

How does it work?

LinkedIn claims that 97% of the people who register and use the services provided learned about the services from word of mouth. That's how it's been intended from the start. The five "founding fathers" started sending out invitations to their close friends and the ball just started rolling. They again invited their close friends and trusted contacts and it started spreading in an explosive speed. If this had been consistent, we could say that at least 97% of the 4million users were in one big cluster, all connected though the original five: Reid Hoffman, Allen Blue, Konstantin Guericke, Eric Ly and Jean-Luc Vaillant, but to prevent spam and unwanted messages from recruiters and so on, LinkedIn has limited the use of "introductions" to three degrees of separation. You can still contact everyone that is registered, but not using the "Hey, I know John and he's a good worker. You should add him to your list." method.

To explore this website I decided to use Raymond Hagen's LinkedIn profile instead of my own because he has about 50 times more contacts than I do (after completing this project, adding people I've talked to etc. my number of contacts is pretty much the same as Raymonds.), and with the above rule of stopping at 3rd degree, Raymond have access to over 650,000 professionals just by introductions. You often hear people say "I know a guy that knows a guy" and this shows how many people that can be.

The LinkedIn method can be described very easy: Register, set up a profile, invite others and search. In this section we will look deeper into each of these and give a detailed account on what to do.

Registration

Most people have never heard of LinkedIn until they get an e-email with the subject "Join my network on LinkedIn" from a friend or colleague. The mail is standardized from LinkedIn and explains in a couple of lines what the service is and gives you a link to click to get registered.

Subject: Join my network on LinkedIn
Since you are a person I trust, I wanted to invite you to join my network on LinkedIn.

I'm using it to discover inside connections I didn't know I had. It's interesting to see the level of access you can have with only a few people in your network.

It's free to join and only takes a minute to sign up and join my network.

- Name of the one doing the invites.

PS: Here is the link:
"Link to the registration site."

It is free to join and takes less than 60 seconds to sign up.

After you click the link you are taken to a page that verifies that you haven't registered at the page before and get duplicate accounts. You can have two accounts if you want to, but it's generally better to have more contacts on one profile then to spread them around. The email say that the registration wont even take a minute, and in a way that's true. To sign up and get an account on the website is quite quick, but that doesn't help you at all. You still need to fill in all your personal details, your past and current work experience, your education and classes/courses and so on. Setting up a complete profile can easily take hours if you want it done right.

To make it easy for LinkedIn to search though their database of information, it's important that all the information is stored in a standard way. As a result, you can't just upload your current CV to the website and expect it to work. There are five sections; Basic, Summary, Experience, Education and Additional information. Each section have several text fields where you put in the information they ask for. Experience is the one that will usually take the longest and it's important that the information is accurate. This is what most people will look at when considering you for a position, and if the information doesn't "add up" they may think you are lying about your work experience.

Another time-consuming part of the process is adding users to your web of contacts. Luckily, LinkedIn have implanted a way that lets you export all your contacts from Outlook, Palm or any other information device, upload them to the website and find out if they are already registered. If not, an easy way of inviting them is presented. The problem is getting them to actually go though the same process you have just done. In a later section I have conducted a small experiment to see how many of my friends I could get to register and actually use this service.

Searching for jobs, old contacts or employees is the main function of LinkedIn, and there are several options to consider when you search. If you are looking for a person you can look up keywords, the exact name of the person or search his references. (If you know the person worked for Microsoft 2001-2004 and in Apple from 2004 – present time you can enter those dates and get a list of candidates that match). A search for "web designer" in Norway gave two results, but only one of them was relevant. The other was an industrial designer that had no relations with what I was looking for. From this we can see that LinkedIn doesn't "understand" what I'm looking for, but just looks up the words like any other search engine. That means the search algorithm looks for things like degree of separation and search words, but not how "popular" this person is when searching for web-designers. When the result is presented it is sorted by degrees away from you.

Finding a job is easier and doesn't give you that many options. Simply put in the keywords you are looking for in a position, select your industry and location (Only two options: "Located in or near Norway" or "Anywhere"). A quick search for web designer in the job-listings gave 17 hits and of those 5 jobs was ONLY being offered on LinkedIn. Once again we get the problem that not all jobs are relevant, but about 60% has something to do with web-design/graphics design. There is also an advanced option that lets you put in more details on the job you are looking for, like position, job function, experience level and so on.

Finding Services is the last "search function" currently available on the LinkedIn website, and this is where you can "point and click" you way to legal advice from lawyers, branding questions or questions about architecture. The list you get when clicking on a service is created from what you put in your profile so again it shows how important it is to have a complete LinkedIn profile.

Why does it work?

LinkedIn works because of a theory called The Small World phenomenon, as discussed in the book Linked by Albert-László Barabási.

The idea was put forward by Paul Erdos and Frigyes Karinthy in 1929 when Karinthy published a book containing 52 short stories, called "Everything is different". One of the stories is called Chains, and the story goes like this: A man makes a bet that he "could name any person among earth's one and a half billion inhabitants and though at most five acquaintances, one of witch he knew personally. " In this book the lead character links himself to a Nobel Prize winner using the above method, but only 3 middle-links.

Chains is the first reference we have to what now is commonly known as "six degrees of separation", a term we got from Stanley Milgram who is/was a Harvard professors researching on human interconnectivity. Let's simplify this a lot, and say: Two friends have 100 friends/acquaintances each (not known by the other) and they again have 100 people, gives a network of 20,000 people. (2*100*100). So any of these 20,000 could be reached within three introductions. It doesn't take many more links before we got number way over billion. The problem is to figure out who knows who, and that's where LinkedIn comes in. They do it for you!

In May 1973 Mark Granovetter published a study called "The Strength of Weak Ties" in the American Journal of Sociology, and it is now considered one of the most influential sociological papers ever written. What he suggest here is that when you are looking for a job or a spouse (Friendster), you are much more likely to find it outside of your closest friends because you all move in the same circles, and it's a limited amount of people. So when it comes down to it, you are more likely to find a job though you're weak ties then your closest friends. Again, this is exactly what LinkedIn is based on: keeping track of your weak ties.

The Test

The whole idea behind LinkedIn is to build up your network, sending out invitations to other people you think belong in this network and using your current contacts to get introduced to others. I decided to invite 10 people using the standard e-mail from LinkedIn and see how many people took it seriously, as spam or just weren't interested.

I selected 10 "subjects":

- 2 current students at Norwegian School of Economics and Business Administration.
- 1 law student from Belgrade, currently studying in Australia.
- 1 Master of Science from the University in Oslo.
- 1 studying graphical information system (databases and maps).
- 1 doing a Bachelor in programming from Bergen,
- 1 economics student at NTNU in Trondheim.
- 1 graphical design student in Grimstad.
- 1 unemployed IT bachelor from HiMolde (graduated).
- 1 economics student from HiÅlesund.

These are all "serious" people that I know, that have complete higher education or are in the process of completing it. Some I haven't talked to in years and wanted to test the "getting in touch with former colleagues" concept of LinkedIn and others are people I talk to on a regular basis. They would all benefit from having a larger network of contacts and knowing me and my contacts would be a nice way of getting introduced.

Within 24hours 3 people had signed up and created their profile. The Master of Science and the bachelor in programming took the setup seriously, but didn't bother adding any information at all. Not sending out invitations to their friends, adding in education, work-goals, past experience or any of the important stuff. They simply did the "60 second registration" and closed the service and forgot about it. The last one, a 21 year old economics student from Ålesund registered as well, but didn't take the service seriously. He set himself up as owner of Nordea Bank (witch he is not) and then lost interest and left the page.

The one studying graphical information systems is a close friend of mine, and I talk to him pretty much every week. When he got the mail, he read it and deleted in right away and called me up to tell me to stop sending him spam. The wording wasn't at all "personal" and it sounded like a mass sending (which it was) and that's why he didn't take it seriously. After I explained the concept to him he was more open, but still didn't want to register. This is one of the fundamental problems I see with LinkedIn. They should not make it so easy to send out the "standard" mails, but instead encourage people to describe the service in their own words and explain why that one person should join. Spam is a HUGE part of our everyday lives and we are getting more and more skeptical to opening emails that "doesn't make sense". The standard invite from LinkedIn is one of those.

After the first three registrations and that one call about spam, I didn't hear anything from anyone. I gave it 4-5 days before I started contacting the subjects to find out why they didn't sign up. Didn't they like the service, didn't they need it or had they simply not received the e-mail?

Two of the users, law student and the bachelor graduate from Molde, looked into the service but didn't see they had any use for it. Both changed their mind after I sent them a 2nd email explaining how LinkedIn could benefit them, but still didn't sign up. Another three (both NHH students and the graphical design student considered it spam and didn't even read the mail) and said they would threat a personalized email completely different. The student at NTNU in Trondheim thought it was some kind of pyramid scheme, even after I explained that it wasn't and he refused to sign up.

So, as a test conclusion I got 3 out of 10 to signup on LinkedIn, but of those three not a single one have logged back on since the initial registration. Why are they so skeptical? Spam has destroyed the e-mail system and in my mind it's just a matter of time before the service dies out and something new comes along. With the huge amount of spam sent everyday a personal, important and maybe even life-changing mail will be considered spam because it was sent to more then one person. These 10 were chosen by me as "most likely to sign up", but not a single one added something to my network. I guess I still need to keep the old ways like MSN, blogs and phone calls to learn about business possibilities from them.

In general, older people are more open to mass-invites, but don't see the use for LinkedIn. They already have their big network of contacts and maintain it the old way, phone calls, lunches, emails and meetings. Young professionals in the start-up phase don't have a web of professional contacts to brag about so LinkedIn would be a great tool for them, but they are very skeptical to new services and think they got a good network on MSN, AIM or blogs and see LinkedIn as a bad version of that. Using LinkedIn in that phase of your life might not work wonders, but it sure won't make it worse either.

Cheating your way to jobs and offers.

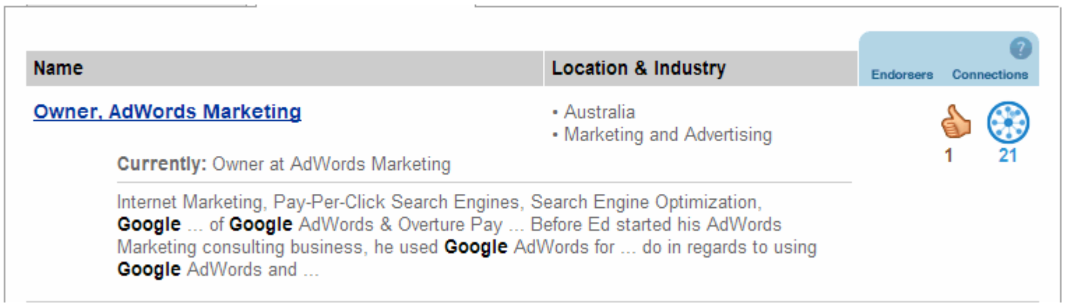
As I was looking for information on the search algorithm that LinkedIn uses I came across a website called Sacred Cow Dung: Cheater's Guide to LinkedIn.

The website in question is written by Christian Mayaud and his specialty (in his own words) is to "Optimize company performance". His Cheaters Guide is written to help people maximize the value of their LinkedIn network, and minimizing the time invested. While some of the tip he gives is downright against the "rules" of LinkedIn, some are just tricks that make life easier when forwarding contacts and getting new introductions. He gives 15 main points with several sub ideas, to many for me to present here so I will just discuss the once I feel is most important.

#1 List your contact info publicly in the body of the profile.
LinkedIn redid their system on in the third quarter of 2005, and since then putting your name and email in the info field like "basic information/qualifications" have been very important if you want people to contact you. What LinkedIn did was to limit who can see your name and send you emails down to 3 degrees of separation. Anything over that you had to use InMail, a LinkedIn function you have to pay to get access to. What they did was simply to hide the name and the email address of the person, but display all other information. So if you add your name and e-mail to one of the other information fields it will be possible to contact you no matter how big your degree of separation is. I haven't read any rules from LinkedIn saying this isn't allowed, but it sure goes against the way they want the service to work.

#2 Add you current email address, in parentheses, to your listing name field.
This goes to the same as the one above, but it also removes the need for introductions. If someone wants to add you to their network, they can just add you directly using your name and the e-mail address you put behind it.

Example:

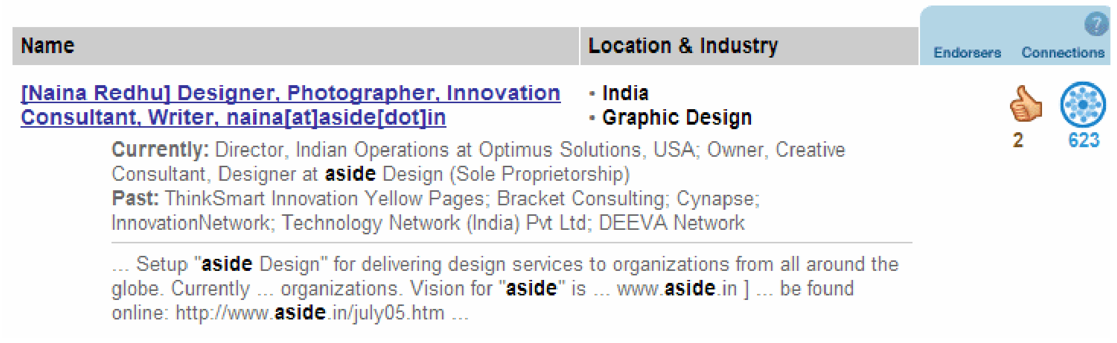

Name | Location & Industry | Endorsers Connections

Owner, AdWords Marketing
• Australia
• Marketing and Advertising
1 21

Currently: Owner at AdWords Marketing

Internet Marketing, Pay-Per-Click Search Engines, Search Engine Optimization, **Google** ... of **Google** AdWords & Overture Pay ... Before Ed started his AdWords Marketing consulting business, he used **Google** AdWords for ... do in regards to using **Google** AdWords and ...

The picture above shows how your profile looks to people outside 3rd degree of separation, and the picture below shows how it will look after you change the text to add your name and email. You can now contact me without the introductions and without paying for the extra service called InMail.


Name | Location & Industry | Endorsers Connections

[Naina Redhu] Designer, Photographer, Innovation Consultant, Writer, naina[at]aside[dot]in
• India
• Graphic Design
2 623

Currently: Director, Indian Operations at Optimus Solutions, USA; Owner, Creative Consultant, Designer at **aside** Design (Sole Proprietorship)
Past: ThinkSmart Innovation Yellow Pages; Bracket Consulting; Cynapse; InnovationNetwork; Technology Network (India) Pvt Ltd; DEEVA Network

... Setup "**aside** Design" for delivering design services to organizations from all around the globe. Currently ... organizations. Vision for "**aside**" is ... www.**aside**.in] ... be found online: http://www.**aside**.in/july05.htm ...

Pictures from BizNetWorking (see references)

#3 Accept all requests to connect.
The idea with LinkedIn is to have a network of people you know and trust, and not people that just want to exploit you and your connections. By accepting all requests, you do it by thinking "everyone is valuable – until proven otherwise". So, if it turns out a contact isn't offering anything, only taking (known as a lecher) you can just delete that contact. This isn't against the rules, it's just morally wrong and again it's not the way LinkedIn is intended to work. LinkedIn has this to say about who you should invite: "Only invite those you know well, only invite those you trust and only invite those you want to forward things to you." These cheats/tips bring us onto the next subject…

Quality or quantity?

For a long time now experts have been discussing if we should focus on quality or quantity in our professional networks. The leaders of three social networking sites like LinkedIn were asked about this.

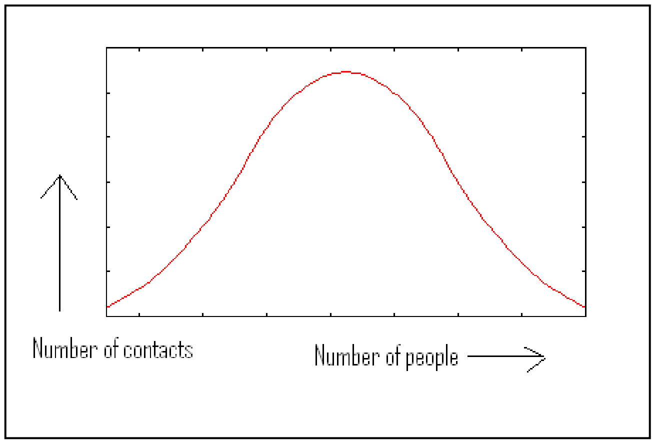
- Thomas Power of Ecademy says, "Go for volume over 'quality,'"
- Mike Walsh, CEO of Leverage Software, says, "Look for quality,"
- Adrian Scott of Ryze, said, "We'd like to create an environment that encourages quality, rather than quantity for its own sake."

When talking about this we often use the expression "weak/strong ties". A strong tie is what we got to our family and close friends, where a weak tie is a person you know but rarely meet or talk to. Strong ties mean a low number of people, while weak ties are a high number. The ideal situation would be high number and high strength, but since there only is 24hours in a day you simply don't have the time to "work" on all those relations.

We have talked about LinkedIn's stand on this before, they want quality over quantity, but they also reward quantity. The more people you're directly connected to, the fewer number of degrees away

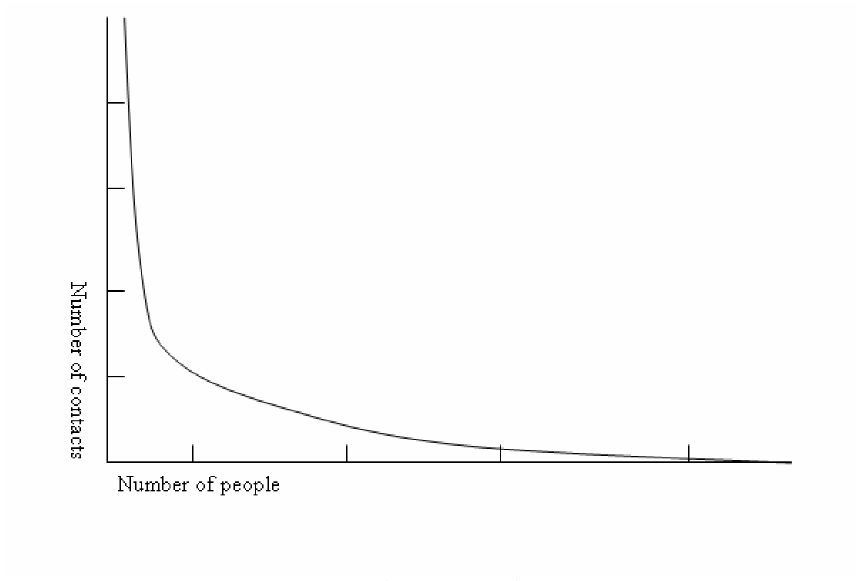
you are from people, on average. With more direct connections, you can see more people, more people can see you, and you're more likely to come up at the top of searches.
But there is also a drawback of being "hub". If I had 10000 1st degree contacts (it's rare but there are people with that many in LinkedIn) I will most likely get a ton of emails about invites and introduction requests. Having a hub in your close network GREATLY increases your ranking inside the LinkedIn community, even if you have no idea who that person is.

There is something called power-laws and Poisson distribution (bell curve). If we assumed everyone had a random number of connections we could get a distribution that looked something like this.



Example of a Poisson Distribution

But there is nothing random about human beings making social contacts. Social networks of college students studied by Malcom Gladwell in his book The Tipping Point, showed that some people have a knack for making friends and that the spread of the average number of contacts is high. Other studies of social networks have also shown a distribution of contacts that follows a power law, that Barabási calls a scale-free distribution. Since there are no limits to how many contacts one person can have in LinkedIn, it is possible that this network is also a scale-free network. We have established that humans don't form relationships at random and on LinkedIn there is no limit to the amount of relationships that can be formed. Therefore we predict the LinkedIn contact network (in real life) follows a power-law distribution where very few have a huge number of contacts, and very many have few contact.. (This cannot be verified however without knowing the number of contacts for a larger sample of the network participants than was evaluated in this study.)



Example of a Scale-Free distribution

I went through the website and selected 8 hubs at random and sent them an email asking about their experiences with having so many contacts. The ones I chose had from 2000 1st hand connections and up to 8000+. The only way I could reach them was using their email (EVERY one of them had listed the e-mail and name in their description, just like I discussed in the "cheats" above).
When I sent these mails out I really didn't expect any answers. I figured their inbox would have to be flooded with invites, forward requests and general spam, and as a result they all got private e-mail addresses they don't give out. Why? To keep all the LinkedIn-business separate from the serious e-mails.

It didn't take more then 20min before the replies started to dump into my inbox, and to my surprise there were some irregularities in the answers. The first mail surprised me a lot, stating that there were NO cons of being a hub. He never got any emails or introduction requests from people he didn't know or just wanted to add a large hub to the close network. But he was the only one that felt that way, cause as the next emails came in I think I got a more realistic image. Average time spent every day on forwarding and invites was around 10 minutes, and around 1hour a week looking for new and interesting contacts. One of the people I had contact with, Jan Karel Kleijn told me about a situation that I though illustrated the cons of being a hub. A recruiter was trying to fill a job, and sent 15 introduction requests to Jan, wanting to get in touch with 15 different people for the same job. And that was just 15 though Jan, god knows how many others he have sent the request to. Jan declined after the second introduction and sent a mail back telling the recruiter to grow a network on his own.

Overall though, they saw the value of a small valuable network, but chose the large ones because of the extended reach and possibilities it presented them. Another thing several of them talked about was that they didn't want to be a "gatekeeper" and forwarded most of the requests because "it isn't my place to decide if an opportunity is interesting or appropriate for someone else."

Ecademy is one of the competitors of LinkedIn in the networking business, and I quoted one of the founders at the beginning of this chapter saying that quantity was better then quality. However, he recognizes that not everyone feel that way and has created the Ecademy Blackstar program. For $4,500 you can become a lifetime member and get exclusive coaching, introductions, and other services. This sounds ridiculously expensive to me, but there is actually a waiting list of people that wants to join. They clearly see the value of quality connections. Of the 46,000 users of Ecademy, 2000 have applied but only 25 are accepted every month. Why? "Because Thomas Power and Roger Hamilton (founders of Ecademy) have limited time available to serve BlackStar Life Members and wish to provide exceptional service and intimacy levels, and to ensure quality levels, service levels, and qualification levels." Sounds like a rip-off to me, but apparently people who work their networks more seriously then I do think it's worth it.

So, how do you find the balance between quality and quantity? It all depends on what your goals are. You first step should be to determine how good a relationship you need with someone in order to get what you want from them, and then maximize the number of connections you can have at that level. Examples can be someone selling investment banking services. In this case you need a high level of knowledge and thrust for someone to let you work their money. A low number of quality customers like CEOs or senior executives who can buy your services. Spreading yourself out to much will result in less trusted relationships and less customers.

If you are a farmer cutting down trees just before Christmas to sell Christmas trees from your barn, you need a big network of people to know about you and choose to come to your house and your quality trees instead of going to a mall and getting the tree from there. Most likely your trees will be more expensive, but your customers keep returning because you give them a quality service and keep a weak tie to the person. The customer is someone you only speak to once a year, so you don't spend much time on each client, but can move quickly to the next one. (This example is based on my dad. Every year we drive about 20min to a farmer that has supplied my dad with Christmas-trees since he built his own house 25years ago. On the way we pass several others selling trees, but because my dad and the dealer got a "connection" he choose to go there instead, even if it is more expensive.)
There is not one right solution overall, your needs will likely be different from one context to another. Because time is the constraining factor, seek out strategies that allow you to build stronger relationships or reach more people with the same amount of effort.

Conclusion

LinkedIn works.

The above statement is the short answer to the question I wanted to explore in this paper. I didn't get the help I was hoping to from the LinkedIn crew, but instead I got some useful insight from long time users of the service. Another problem I ran into was that LinkedIn reworked their website and how to see other contacts not long after I presented my project outline. At that time you could see the degree of separation to everyone, but lately that have been changed to see max three. The reason for this was all the spam. When someone wanted to contact a person 10degrees away, it resulted in a ton of emails and requests being forwarded, so they just removed that option entirely and inserted a "mail" system used inside LinkedIn for direct contact, but access to this requires a monthly fee.

My test didn't work quite as I had expected, but I think that has a lot to do with the age-group I approached and the location. People already in a job or currently looking for one are probably more likely to find this service more useful then a student, and Norway probably isn't the strongest nation right now when it comes to building online networks like this. One thing I did learn though was that if you want people to take you seriously online, every mail you send them must be "personal" even if it contains no personal info. The language you use, the wording, it all counts in on the reader's decision to call it spam or something useful.

As I said earlier, there are some cheap tricks one can use to improve the chance of getting something valuable out of LinkedIn, and while it may be cheap and against the rules it works very well! Until I see some direct guidelines on this from LinkedIn, I will use the tricks as best I can. If it hadn't been for the "email in summary" I wouldn't have been able to contact all those hubs, and when three of those hubs included me in their network, the number of contacts I had went from 50,000 to over 500,000. If someone is looking for an employee with my qualifications, the chance of that person finding me is 10 times higher just because I added three hubs to my network.

References:

Books:
"Linked", by Albert-László Barabási.

Website:
www.linkedin.com and sub pages.
www.friendster.com and sub pages
http://www.sacredcowdung.com/archives/2005/05/cheaters_guide.html
http://biznetworking.blogspot.com/2005/08/served-hot-linkedin-cheatsheet-1-how.html
http://www.ecademy.com/
http://en.wikipedia.org/wiki/Scale-free_network
http://www.fastcompany.com/resources/networking/teten-allen/010305.html

Extra:
E-mail correspondence with Wen Wen Lam, Press contact.
E-mail correspondence with hubs in the network: Peter R. Luiks, Jan Karel Kleijn, Jerry Laiserin, Rhonda Campbell.
E-mail correspondence with Joe Bartling, long time user of LinkedIn.

Conversation with the test-subjects on MSN and phone.

Six degrees of separation

In Albert-László Barabási's book "Linked" the author writes about a Hungarian writer named Frigyes Karinthy. Karinthy published a concept called "six degrees of separation". He believed that the world at his time was so closely connected that you could link any two people to each other with a maximum of six links.

The concept was to link people together with acquaintances. One person knows another person, who in turn knows another person, and that person knows the person we were trying to connect the first person to.

The American professor, Stanley Milgram, tested this theory in real life. He sent letters around to people in the U.S.A. were he asked them to forward the letter to people they knew who may know the person on the bottom of the list. They each wrote down their name and sent it to someone they thought might be closer to the targeted person. He got about 50 letters back and found an average degree of separation of 5.5. This was close to the six degrees of separation.

The goal of my project

The relationship between people is a network and so is the relationship between web pages. I was fascinated with the six degrees of separation theory and I wanted to make a tool that could find the degrees of separation between one web page and another web page. The tool could also be used to show how much work the reader had to put in to get from one page to another, because the degrees of separation is in fact how many clicks the user must make to get from A to B.

The mapping of the network on a server could be done in two ways:

1) Go through each page and find which pages it is connected to.
2) Make a spider that crawls through the pages and collects the links for you.

The second option was definitely the easiest, most flexible and fastest option. With a spider I could search through any web page, and I could just lean back and let my computer do the information gathering for me.

I had already used a pre developed spider in a different course. The spider was made in Java, and its purpose was to gather words to make a small search engine. I wanted to make my own spider with a graphical user interface made in Microsoft's .Net framework that only concentrated on the links. The goal was pretty clear to me, so I began building a spider.

Web spiders

The web spider is also called a robot or a crawler. I prefer the word spider though. A spider crawls from one page to another, picking up new URLs from each page it visits. These URLs are then visited and the spider collects new URLs, and so it continues until it has visited all the pages it can reach.

I knew I had to download the HTML-code from the web server somehow. The spider I had used before did this for me, so now I had to start from scratch. A quick search on Google gave me examples of spiders made in C#.

The work of the spider

The spider is given a link to a web page. It downloads the HTML-code of the page and then scans the code for A-tags. The A-tag is presented on the web browser as a link that the reader can click on to go to another page. It doesn't matter if the A-tag is wrapped around a picture, or if it is commented out of the code. The spider will collect it anyway. The spider extracts the link from the A-tag, but to prevent the spider from downloading and searching a page twice, we check that this page hasn't been registered before. If the page isn't registered we put it in a queue. If it is already downloaded we ignore it. This procedure continues until the whole HTML-code is searched.

When the page is searched it checks if there are any new pages in queue. If there aren't the spider has completed its task and can end. If the queue contains one or more items it will pick the first item out of the queue, download the page, search for new links, and so on.

1. User inputs domain
2. Domain is inserted into queue
3. If queue is empty then quit. If not, download page
4. Search page for A-tags
5. Extract links from A-tag
6. Put searched page into list of pages that has been searched
7. Put links to pages, that hasn't been searched, into queue
8. Register the searched page's outgoing links
9. Return to 3

This is in general what the spider does, but I also had to implement some other features to the spider to make it better for my application. For instance; the GUI had to be updated for each page that had been searched, each error that occurred, for each time the link counter changed, for each time new bytes were downloaded, and for each time a new page was put into queue.

Extraction of links

The extraction of the URL was a bit of a problem. First I used the substring to search the document for the A-tags, but I soon realised that this was an incredibly slow way to search the document. The CPU-load was huge, and it took a long time to go through each sign. I then discovered the Regex object, which searched the document in a much faster way then the substring.

I only managed to extract the whole A-tag with the use of the Regex object, and I would get strings like:

<a target=_blank href='http://www.himolde.no/page.html' class="link">

There were several problems with the links. The first problem is to find where the link starts, and where it ends. We know that it starts after href=, but there are three ways to specify this attribute. The first one is to start and end the attribute with the sign ". The second possibility is to use the sign ' instead of ". The last one is to use no signs at all.

So I made an algorithm that used substring to find the text href=, and then I found the next letter. If the letter was " or ', I would search for the next letter of the same kind in the string. The text between the two signs was the link. If the string didn't start with " or ' the text would be from href= to the first blank space in the text, or to the end of the text.

Since I only want URLs to pages that are on the same domain I would have to do some more checks. The first thing I checked was the start of the link. A link can start with a name of a protocol. E.g. http, ftp, irc, and so on. I only wanted links that used the http protocol. The other links where thrown away.

The next thing I had to check was if this was a page that was on the same domain as I started searching from. If the URL didn't start with http:// I knew that this was a page on the same domain as I started from. Without http:// the link will be an internal link. If the URL starts with http:// there is a chance that this is an external link. I had to make a check if the domain that was given after http:// was the same as the domain that I started from. If the domain was different, the URL was thrown away.

All of the link extraction procedures were put into a class called UrlCleaner. Check appendix for class diagram

Tests and results

I decided to analyse the himolde.no-domain, but I soon realised that this domain was too big to get any average degrees of separation between the pages, having 1672 pages. But I managed to do some tests. I picked out a random page with an approximately average amount of links. The page (http://www.himolde.no/index.cfm?pageid=1914) had 56 links, which is quit a lot, but the reason is the sidebar and header on the page. The sidebar contains a lot of links to central pages, like the main page and page "for studentene" and "for ansatte".

The average degree of separation from the page to all other pages on the network was 2.77. This tells me that I could get to many pages from this page with three clicks. This is not at all surprising. The sidebar is included in most of the pages and connects them closely.

The pages at www.himolde.no are dynamical pages, and this is also a factor that has to be considered. If the pages weren't dynamic the sidebars would probably disappear, because static pages can't include code from other files, and therefore has to have a unique collection of links on each page. Maintaining a sidebar in static pages would be a nightmare; if the webmaster wanted to change one link, he would have to change the links on all of the pages.

We could say that dynamic web pages are connected tighter than static web pages. Since almost all major sites are dynamic these days, it may be hard to test this theory.

I also tested my own web page located at www.frening.com. This is a much smaller site with only 22 dynamical pages. I got an average degree of separation of 1.49 for all the pages on the site. This is very little, and I not at all surprised. The network is small, and I also use sidebars at my page.

Conclusion

The fact that my application is pretty slow at finding the degrees of separation may ruin a bit of the project, but my project description in my project outline states that the application should only be used to find the degrees of separation from A to B, and not any average values. But I wanted to get these values anyway, since they were a bit more interesting than the path between to pages.

The results I got are reasonable. In the book "Linked" a team calculated that the web had 19 degrees of separation. The web is a huge network. www.himolde.no and www.frening.com is just a tiny drop in the ocean, and it appears that smaller networks have a smaller degree of separation, which makes sense. I'm just looking at a tiny part of the web.

References

Albert-László Barabási: "Linked"

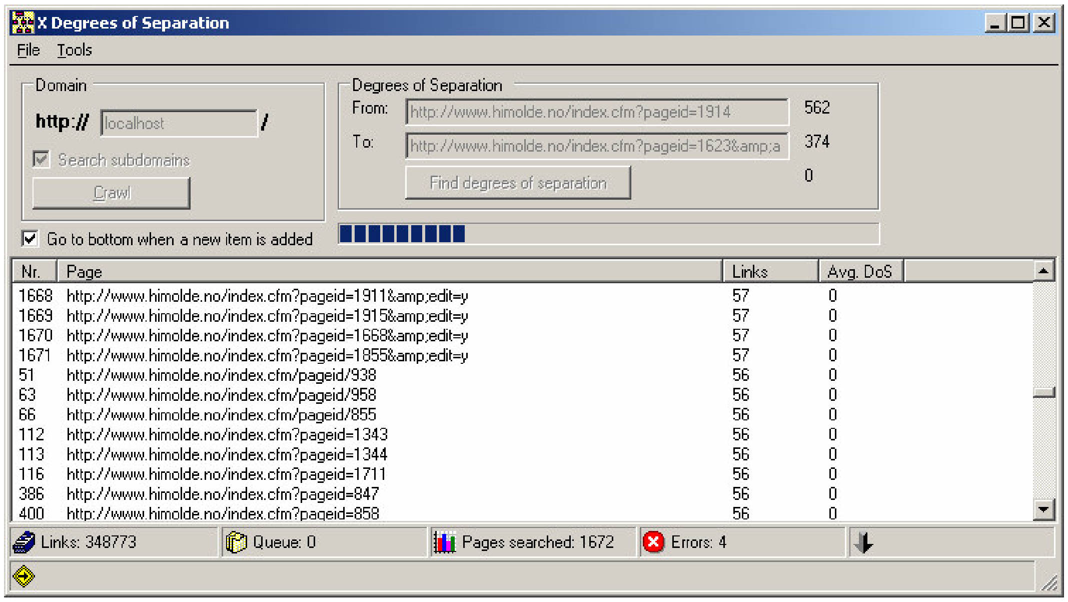
http://www.codeproject.com/aspnet/Spideroo.asp: Populating a Search Engine with a C# Spider - The Code Project - ASP.NET

Appendix

The application is developed with the use of Microsoft Visual Studio .NET 2003. The language used was the relatively new C#.

System requirements
Windows 2000 or newer
Microsoft .NET Framework 1.1 or newer.

GUI



From: http://www.himolde.no/index.cfm?pageid=1914 562
To: http://www.himolde.no/index.cfm?pageid=1623&amp;a 374
0

Nr.	Page	Links	Avg. DoS
1668	http://www.himolde.no/index.cfm?pageid=1911&amp;edit=y	57	0
1669	http://www.himolde.no/index.cfm?pageid=1915&amp;edit=y	57	0
1670	http://www.himolde.no/index.cfm?pageid=1668&amp;edit=y	57	0
1671	http://www.himolde.no/index.cfm?pageid=1855&amp;edit=y	57	0
51	http://www.himolde.no/index.cfm/pageid/938	56	0
63	http://www.himolde.no/index.cfm/pageid/958	56	0
66	http://www.himolde.no/index.cfm/pageid/855	56	0
112	http://www.himolde.no/index.cfm?pageid=1343	56	0
113	http://www.himolde.no/index.cfm?pageid=1344	56	0
116	http://www.himolde.no/index.cfm?pageid=1711	56	0
386	http://www.himolde.no/index.cfm?pageid=847	56	0
400	http://www.himolde.no/index.cfm?pageid=858	56	0

Links: 348773 Queue: 0 Pages searched: 1672 Errors: 4

The menu has two main elements: File and Tools.

File

New	Clears the spider's memory and makes it ready for a new search.
Open	Opens a saved search.
Save as	Saves a completed search.
Exit	Exits the application.

Tools
Find average degree for FROM-page - Finds the average degree of separation from the page in the from-page-textbox and to all the other pages on the network.

Find average degrees of separation - Finds the average degrees of separation between all the pages on the network.

The domain groupbox
The domain textbox - This is were the user inputs the domain that is going to be searched.

The search sub domains checkbox - If this checkbox is unchecked the spider will not add pages for the domains subdomains to its queue.

The crawl button - Starts searching the given domain.

The degrees of separation groupbox
The from-page-textbox - Contains the URL to page that we should find the degrees of separation from. This textbox is also used for determining which page we want o find the average degrees of separation from.

The to-page-textbox - The page we should find the degrees of separation to.

The find degrees of separation-button - Finds the degrees of separation between the from-page and the to-page. Also prints out the shortest path between the two pages.

The three numbers behind the textboxes - The three numbers are only put there as counters to visualize that the application is working. The number at the top shows the length of the queue in the degrees of separation algorithm. The number in the middle is the number of pages the algorithm has calculated. The number at the bottom has no function, but has not been removed. (You might call it a bug.)

Under the two groupboxes we'll find a checkbox and a statusbar. If the checkbox is checked the listbox will focus on the last item added to the listbox when the spider is searching.

The statusbar shows the progress for both the actions in the tools menu.

The listbox
The listbox has four columns which are all sortable.

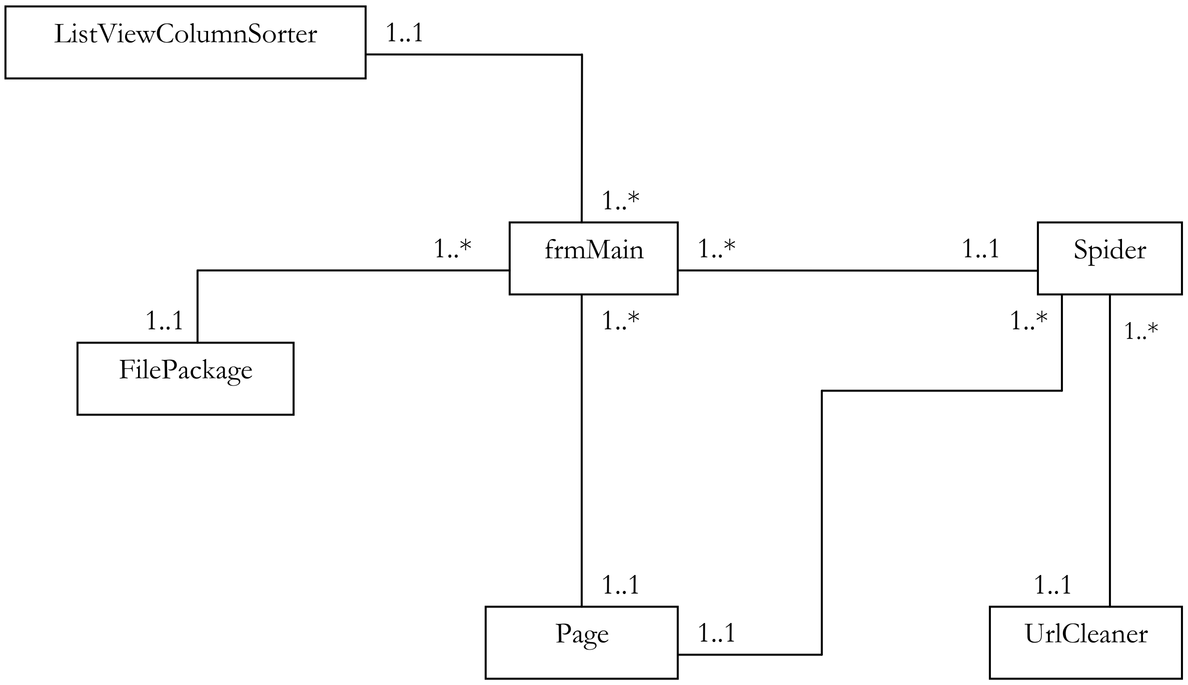
Nr.	Shows in which order the pages were added.
Page	The URL of the page that has been searched successfully.
Links	The number of outgoing links from the page.
Avg. DoS	Average degrees of separation from the page to all other pages.

The statusbars
The upper statusbar contains five panels.

Links	The total number of links from all the pages to other pages on the network.
Queue	The number of pages in queue to be scanned for links.
Pages searched	The number of pages that has been searched.
Errors	The number of errors that has occurred while the domain has been searched.
The downwards arrow	Shows the amount of data that has been downloaded.

The statusbar at the bottom shows the last error that occurred while searching.

Class diagram


ListViewColumnSorter
1..1
1..*
frmMain
1..*
1..*
1..1
Spider
1..1
FilePackage
1..*
1..*
1..*
1..1
Page
1..1
1..1
UrlCleaner