SAMMENDRAG

”Tolkning av fotballreferater” er en Diplomoppgave utarbeidet ved NTNU og Institutt for Datateknikk og Informasjonsvitenskap. Oppgaven er gitt av Jon Atle Gulla som også fungerer som veileder.

Vi har i dette prosjektet utviklet et system, SoccerFinder, som analyserer setninger fra et fotballreferat. Vi tar utgangspunkt i referater som publiseres på nett av VG, mer spesifikt et referat fra kampen Skottland - Norge. Slike fotballreferater er skrevet av en som følger kampen direkte, og de inneholder derfor en del dårlig norsk samt ufullstendige setninger. Dette kompliserer arbeidet vårt, og vi er blant annet vært nødt til å skrive om en del setninger.

Informasjonen vi ønsker å ta vare på er hendelser som for eksempel dueller, utdeling av kort, tidspunkt for eventuelle bytter samt hvor mange cornere og frispark en kamp hadde.

For å analysere setninger bruker vi i prosjektet vårt et lingvistisk søkeverktøy, HoG, som er plassert på Dragvoll. Vi sender setninger til HoG ved å bruke XML-RPC, og får tilbake resultater i form av RMRS representasjoner på XML format.

RMRS er en teknikk for å representere semantiske strukturer der elementære predikasjoner knyttes til begivenheter. RMRS er komplekse representasjoner, og det har vært en tidkrevende prosess å sette seg inn og forså disse. Vi filtrerer ut ifra en RMRS, informasjon som for oss er interessant å ta vare på. Dette gjør vi ved å bruke XSLT, et stilsett som henter ut data fra XML dokumenter. Den filtrerte informasjonen legger vi så inn i en databasetabell, og kan seinere kjøre spørringer mot denne for å hente fram informasjonen.

Systemet vårt er en Java-Servlet som kjøres fra en Tomcat webserver. Vi har her enkle websider der vi velger om vi vil sende en setning til analysering, eller om vi ønsker å kjøre en spørring mot informasjon i en database.

Ved å utvikle et system som SoccerFinder vil vi være i stand til å hente fram aggregert informasjon fra et eller flere referater. Dette er det som hovedsakelig skiller oss fra dagens søkeverktøy, som for eksempel Google, der vi hovedsakelig vil kunne søke oss fram til forekomster av ord.

Arbeidet med denne oppgaven er utført våren 2005 og er en avslutning på Masterutdanningen i Datateknikk.

FORORD

Denne Diplomoppgaven er skrevet ved Institutt for Datateknikk og Informasjonsvitenskap ved NTNU. Oppgaven ble tatt ut 18. januar 2005 og levert 14. juni 2005.

Oppgaven går ut på å lage et system for å søke gjennom, hente ut og lagre relevant informasjon fra fotballreferater publisert av nettaviser, for eksempel VG.

Jeg vil takke Jon Atle Gulla som har vært min veileder. Ønsker også å takke Lars Hellan ved Lingvistisk Institutt samt Atle Prange ved Businesscape som har vært behjelpelig med den tekniske utviklingen av systemet.

Trondheim, 14. juni 2005

Stian Junge Strand

_____

INNHOLDSFORTEGNELSE

1.0 Introduksjon

1.1 Oppgavetekst

Oppgaveteksten er som følger:

Det skal lages et system som tolker setninger i fotballreferater og representerer deres innhold i en strukturert database. For analysen av setninger skal en bruke en norsk Minimal Recursion Semantics-grammatikk utviklet ved lingvistisk institutt. En må selv definere databasen og bruke XML til å mappe MRS-strukturene til felter i databasen. En skal videre lage et enkelt menysystem for å hente ut informasjon fra databasen. Mens en for setningsanalysen skal bruke et verktøy implemenetert i C++, må en utvikle resten av systemet selv i Java eller Python. Et liknende system for turistbeskrivelser er allerede laget og kan brukes som modell.

1.2 Problemstilling

Det finnes i dag mange nettsteder som publiserer referater fra fotballkamper. Dette er lange tekster delt inn etter hvilket spilleminutt forskjellige hendelser tok plass. Man kan selvfølgelig lese hvem som spilte mot hverandre og resultatet direkte fra overskriften på referatet, men er man derimot interessert i mer detaljert informasjon og tendensene i kampen, må man sette seg ned å studere teksten nærmere. Dette vil være en tungvint jobb da man rett og slett må gå igjennom referatet linje for linje og notere ned den informasjonen man er interessert i.

Videre kan det også være interessant å sammenlikne forskjellige kampforløp og tendenser i et lag, for eksempel det norske landslaget, i den pågående VM-kvalifiseringen. Et slikt arbeid ville blitt mye enklere og effektivt med et passende verktøy for søk og representasjon av data.

Hvordan kan vi så klare å hente informasjonen ut fra et fotballreferat? Hvilken informasjon er det som er interessant? Hvordan vil det være fordelaktig å representere denne informasjonen?

1.3 Tilnærming

Det vi ønsker å utvikle i dette prosjektet er et system for å søke igjennom og hente ut informasjon fra fotballreferater. Dette vil i utgangspunktet være korte, enkle spørsmål til referatene. Ved videre arbeid vil det også være ønskelig å spørre mer intrikate spørsmål som krever mer tolkning av fotballreferatet. Dette kan for eksempel være ønske om å se om en spiller har gjort en god kamp, vært mye involvert og liknende.

Vi vil i dette prosjektet utføre søk ved hjelp av et lingvistisk søkeverktøy. Dette verktøyet utfører en grammatisk analyse av gitte setninger og gir oss tilbake semantiske strukturer i form av RMRS (Robust Minimal Recursion Semantics) representasjoner. Søkeverktøyet presenterer disse som XML dokumenter som det videre er ønskelig å trekke ut relevant informasjon fra. Vi vil på denne måten gå igjennom gitte referater linje for linje, for så å hente ut interessant informasjon.

Det vi i utgangspunktet ønsker å lagre er hendelser som dueller, utdeling av kort, når eventuelle bytter blei utført, hvor mange cornere og frispark en kamp hadde og så videre. Denne informasjonen kan i neste omgang brukes til å sammenlikne forskjellige kampforløp.

Vi vil i utgangspunktet lese inn en og en linje manuelt fra fotballreferatet. Fotballreferatene vil være referater publisert på nett av VG (VG 2005). Til å begynne med tar vi for oss ett bestemt referat, Skottland – Norge, men i et ferdigutviklet system vil det være ønskelig å takle et hvilken som helst referat.

1.4 Resultat \ Bidrag

Resultatet av dette prosjektarbeidet er en applikasjon programmert i Java.
Applikasjonen tar for seg linjer fra et fotballreferat, kjører disse igjennom ett lingivstisk søkeverktøy og presenterer informasjonen i en database. Systemet er en Java-Servlet som kjøres fra en Tomcat webserver. For å utføre søk bruker vi søkeverktøy HOG (Heart of Gold) som befinner seg på Lingvistisk institutt på Dragvoll. Kontakt med HOG oppnår vi ved å bruke XML-RPC (Remote Procedure Call) som er anvendelig sammen med Java.

Søkene vi utfører resulterer i XML dokumenter bestående av kompliserte RMRS strukturer. Vi må ut ifra disse hente ut de data som er relevante for oss, for så å mappe disse til en database. Vi kan så kjøre spørringer mot databasen ved hjelp av en enkel meny.

Selve brukergrensesnittet vil være en webside der vi kan skrive inn setningen vi vil analysere samt enkle valg over hvilke spørringer vi ønsker å utføre. Slike spørsmål vil for eksempel være:

- Hvor mange gule kort var det i kampen?
- Hvor mange røde kort var det i kampen?
- Hvor mange mål var det i kampen?
- Hvor mange bytter var det i kampen?

Det er også ønskelig å kunne hente ut aggregert informasjon fra forskjellige referater, for eksempel finne i hvilke av kampene det ble laget flest mål eller hvilken kamp som hadde flest gule kort.

1.5 Strukturen på rapporten

Vi starter rapporten med å se på dype- og grunne prosesseringsteknikker samt hvordan disse kan kombineres i spørresystemer. Videre presenterer vi RMRS som er representasjonen vi vil få tilbake fra søkeverktøyet etter å ha analysert en setning. Kapittelet avsluttes med å se på alternative metoder for å finne informasjon i fotballreferater. Søkeverktøyet HoG presenteres deretter med et underkapittel som omhandler XML-RPC.

Kapittel 4 tar for seg konstruksjonen av systemet som vi har valgt å kalle SoccerFinder. Vi presenterer her UML diagrammer, forklarer samtlige klasser og gir en kort introduksjon til de forskjellige utviklingsverktøy som er brukt.

Vi legger så fram en evaluering av systemet, og til slutt kommer en konklusjon på arbeidet som har vært utført våren 2005. Til slutt i rapporten er det referanselister, oversikt over alle figurer, eksempel på fotballreferat og RMRS representasjon og til slutt vedlagt programkode.

2.0 Teoretisk Bakgrunn

2.1 Dyp eller Grunn Prosessering?

Dype og grunne prosesseringsteknikker har utfyllende styrker og svakheter. En dyp analyse vil være ønskelig ved en mindre mengde detaljert data, og grunn analyse ved rask prosessering av store mengder data. Vi ser her på noen av egenskapene ved de to teknikkene samt forskjellen mellom de to.

Vi kan si at det finnes 3 forskjellige egenskaper som begrenser den praktiske nytten av systemer basert på dyp tekstprosessering(Copestake 2003). For det første kan antall resulterte analyser rett og slett bli for høyt og gjøre det urealistisk å få til noen form for filtrering i etterbehandlingen. Vi har teknikker som for eksempel *packaging*, men dette utsetter bare problemet, da det er få programmer som kan håndtere de pakkede representasjonene. Det foregår også arbeid der man forsøker å integrere statistiske metoder ved analytisk utvelgelse, men dette er ikke ferdig utarbeidet.

Det andre problemet går på robustheten. ”Dype systemer” forutsetter en grad av grammatisk korrekthet noe som medfører at inndata som ikke møter disse kriteriene ikke vil bli analysert. For eksempel ved rå tekst kan små typografiske feil føre til at det hele blir avbrutt.

Fart er det tredje problemet, da dype prosesseringsteknikker ikke er raske nok til å håndtere store mengder med tekst som gjerne kreves i for eksempel IR, IE og spørresystemer.

Det finnes flere teknikker for prosessering som tilbyr en lingvistisk struktur på et grunnere nivå. Vi har for eksempel part-of-speech tagging (POS-tagging) som går ut på å klassifisere alle ord i en setning til hvilke ordklasser de tilhører, og også NP chunking. Fordi disse inneholder en mindre detaljert representasjon, har de en tendens til å ha en lavere grad tvetydighet.

Fordi en dyp prosessering omfatter et større søkeområde enn grunn prosessering er den derfor i seg selv tregere. Den er også mindre robust da den krever en viss grammatisk korrekthet og mer detaljert leksikalsk informasjon. Den viktigste forskjellen mellom dyp og grunn prosessering angår de leksikalske kravene. Mer konkret så krever dyp prosessering detaljert subkategorisering av informasjonen, mens grunn prosessering kun trenger en indikasjon på mulige Part-of-speech samt informasjon om irregulær morfologi (lex: en språklig enhet med betydning).

2.2 Dype og Grunne teknikker kombinert

Vi legger her fram måter disse fremgangsmåtene kan kombineres for å gjøre nytte av fordelene ved de to.

Et spørresystem er et godt eksempel på en applikasjon med klar deling mellom dyp og grunn prosessering. På den ene siden er det nødvendig med en grunn fremgangsmåte for å kunne håndtere store mengder svartekst for å kunne finne mulige svar. Det er derimot også færre relativt korte spørsmål som skal besvares, så en presis analyse er også essensiell. I et spørresystem er derfor en dyp prosessering av spørsmål både mulig samt nyttig(Copestake 2003). Spørsmål kan nemlig inneholde avhengigheter som en grunn teknikk ikke vil være i stand til å oppdage.

Andre applikasjoner der en kombinasjon vil være aktuell er:

- Intelligent summarization som går ut på å finne målet med en artikkel ved å identifisere setninger (robust). Vi vil også se dype prosesseringsteknikker da de ofte består av anstrengt vokabular og syntaks.

- Information Extraction . Et IE system må kunne håndtere en enorm mengde med tekst så dyp prosessering alene er håpløst. Derimot er de fleste eksisterende grunne prosesseringssystemer avhengig av store mengder håndkoding for hvert domene, noe som ikke er veldig presist. Det ideelle ville derfor vært å bruke grunn prosessering med minimal domeneavhengighet for å finne interessante setninger, for deretter å sette i gang en dyp prosessering med en domene uavhengig grammatikk for presist å kunne hente ut den informasjonen som er interessant.

Det er altså forskjellige fremgangsmåter for å kombinere dyp og grunn prosessering, blant annet ved å utvide en dyp prosessor med grunne teknikker. Dette vil for eksempel være aktuelt hvis en dyp prosessering er rask nok, men vi ønsker å bedre robustheten ved for eksempel ”parse failure”. Dette kan oppnås ved at vi har en strategi der vi går over til grunn prosessering dersom den dype skulle feile.

2.3 Robust Minimal Recursion Semantics

Alle strategier som er nevnt krever en viss integrasjon av dyp og grunn prosessering, noe som skaper en del problemer samt uenigheter. (Copestake 2003) foreslår en fremgangsmåte der man produserer en semantisk representasjon fra dype og grunne analyser med sikte på å oppnå en høy grad av semantisk kompatibilitet. Grunner til at en slik standardisert semantisk representasjon vil være nyttig er blant annet at:

- Forskjellige parsere og generatorer kan kobles til forskjellige underliggende systemer.
- Vi kan presist lese semantiske representasjoner konstruert ved hjelp av grunne analyser og underspesifisert form.
- I tillegg vil begrensninger ved aktuelle domener komme fram i semantikken, noe vi kan bruke videre for å begrense domene ved både grunn og dyp analyse.
- De formelle egenskapene vil også være klarere, og vi får derfor en representasjon som er mer generelt anvendelig.

Den semantiske representasjon som legges frem er RMRS (Copestake 2003), Robust Minimal Recursion Semantics, som er en modifisert utgave av MRS (Copestake et al. 2003).

(Copestake 2003) starter introduksjonen av RMRS ved hjelp av den semantiske representasjonen av setningen ”*Every fat cat sat on some table*”. Videre antar vi at den semantiske representasjonen av denne setningen med definisjonsområder er (1):

every(x, fat(x) ^ cat1(x), some(y, table1(y), sit1(e_{spast}, x) ^ on2(e`, e, y)))
some(y, table1(y), every(x, fat(x) ^ cat1(x), sit1(e_{spast}, x) ^ on2(e`, e, y)))

Figur 1 - RMRS1

Representasjonene bruker generaliserte kvantorer, noe som også er fundamentalt i det videre arbeidet. En viktig ide bak utviklingen av RMRS er at tradisjonelle semantiske representasjoner kan relateres direkte til underspesifisert semantisk informasjon(Copestake 2003). Vi starter med å se på en POS tagger og resultatet den gir for setningen vår (2):

Every_AT1 fat_JJ cat_NN1 sat_VVD on_II some_DD table_NN1

Figur 2 - RMRS2

Vi kan videre anta at POS taggeren gir oss noe semantisk informasjon, for eksempel ved lemmatisering som finner fram til baseformen av et ord. Vi vil da i setningen vår få at *sat* lemmatiseres til *sit.* Taggeren kan også være med å fjerne tvetydighet ved at vi for eksempel klarer å skille subjektet *table* fra verbet *table*. POS tagging gir oss informasjon om relasjonen mellom

ordene, men ingen informasjon om definisjonsområdet, og er derfor underspesifisert i forhold til en dyp prosessering. En mulig representasjon av den semantiske informasjonen knyttet til POS taggeren er da (3):

_every_q(x0), _fat_j(x1),_cat_n(x2),_sit_v(e3$_{past}$),_on_p(e4),_some_q(x5),_table_n(x6)

Figur 3 - RMRS3

Vi ser her at vi har en entitet per ord i setningen der *x* viser objekter og *e* begivenheter. Taggene tilsvarer relasjoner og argumenter, og en relasjon sammen med sitt argument er det vi kaller for en elementær predikasjon(EP). EP-ene er satt sammen for å vise den semantiske representasjonen av setningen.

Spørsmålet er nå om vi kan argumentere for at dette er en underspesifisert semantisk representasjon(Copestake 2003)? Og hvordan kan den semantiske representasjonen vi får fra dyp prosesseringen manipuleres for å gjøre koblingen mot en grunnere semantikk tydeligere(Copestake 2003)?

Det vi nå må gjøre er å splitte den semantiske representasjonen opp i individuelle enheter. Ved første tilnærming mot dette benytter(Copestake 2003) seg av MRS(Copestake et al. 2003). Representasjonen som stemmer overens med (1) blir nå (4):

l0:every(x,h1,h2), l1:fat(x),l1:cat(x),l4:sit1(e$_{spast}$,x),l4:on2(e´,e,y),l5:some(y,h6,h7),l6:table(y), qeq(h1,l1),qeq(h6,l6)

Figur 4 - RMRS4

Hovedpoengene ved en MRS representasjon er(Copestake et al. 2003):

- MRS-en kan spesialiseres til eksakt å utrykke strukturene med de gitte definisjonsområdene (1).
- Hver EP er merket med etiketter, *l0, l1* og så videre..
- Argumenter til definisjonsområdene er angitt ved *h1, h2* og så videre.. Disse kan så fylles med EP-er etter samsvar med restriksjonene på definisjonsområdet.
- Restriksjoner av definisjonsområdet er gitt ved *qeq*-relasjoner.

I standard MRS er bindeord angitt ved like etiketter, for eksempel har *fat* og *cat* samme etikett (*l1*). I den videre utviklingen her er derimot ikke dette ønskelig, og en variant der alle etiketter på EP-er er unike blir brukt(Copestake 2003) (5):

l0:every(x,h1,h2),l1:fat(x),l2:cat1(x),l3:CONJ,l4:sit1(e$_{spast}$,x),l14:on2(e´,e,y),
l9:CONJ,l5:some(y,h6,h7),l6:table1(y),
qeq(h1,l3),qeq(h6,l6),in-g(l3,l1),in-g(l3,l2),in-g(l9,l4),in-g(l9,l14)

Figur 5 - RMRS5

CONJ er her symbolet som angir bindeord og *in-g* står for ”in group” og forteller hvilke bindeord som hører sammen. Det antas her at vi vet antall argumenter som følger med hvert predikat noe som ikke brukes i en grunn semantisk situasjon. Vi må med andre ord finne en representasjon der argumenter kan behandles individuelt. (Copestake 2003) legger da frem en notasjon på såkalt ”Parsons style” da den bygger på ideer fra(Parsons 1990) (6):

lo:every(x),l1:fat(x),l2:cat(x),l3:CONJ,l4:sit1(e$_{spast}$),l14:on2(e´),l9:CONJ,l5:some(y),
l6:table1(y),qeq(h1,l1),qeq(h6,l6),in-g(l3,l1),in-g(l3,l2),in-g(l9,l4),in-g(l9,l14)
RSRT(l0,h1),BODY(l0,h2),RSTR(l5,h6),BODY(l5,h7),

ARG1(l4,x),ARG1(l14,e),ARG2(l14,y)

Figur 6 - RMRS6

For å gå fra MRS til denne representasjonen må argumentene til de forskjellige predikatene navngis, ARG1, ARG2, RSTR, BODY og så videre. Den mest markante forskjellen på MRS og RMRS er da at vi i MRS vil ha en representasjon på formen *Skyte(e, x, y, z)*, mens vi på RMRS form vil presentere dette som *Skyte(e), arg1(e, x), arg2(e, y), arg3(e, z)*. Videre har vi to hovedforskjeller på (6) og Parsons representasjon. For det første brukes ARG1, ARG2 og så videre. istedenfor agent og klient (Copestake 2003). For det andre brukes etiketter istedenfor begivenheter som referansepunkt for argumenter.

Vi er nå bare to steg unna den ønskelige robuste representasjonen og det første går ut på å ha distinkte navn på alle variabler samt å vise sammenheng mellom dem ved hjelp av likhetstegn(*x0=x1*)(7):

l0:every(x0),l1:fat(x1),l2:cat1(x2),l3:CONJ,l4:sit1(e3$_{spast}$),l13:on2(e4),l9:CONJ,
l5:some(x5),l6:table1(x6),
qeq(h1,l1),qeq(h6,l6),in-g(l3,l1),in-g(l3,l2),in-g(l9,l4),in-g(l9,l14),
RSTR(l0,h1),BODY(l0,h2),RSTR(l5,h6),BODY(l5,h7),ARG1(l4,x2),ARG1(l14,e3),
ARG2(l14,x5), x0=x1, x1=x2, x5=x6

Figur 7 - RMRS7

Dette gjør vi for å vise den ekstra informasjonen som er involvert. Til slutt er det nødvendig med en navngivings konvensjon (det er ikke noen standard ennå) slik at konstruerte predikater direkte kan relateres til den dype representasjonen. Her er *sit_V_1* mer spesifikk enn *sit_V*, *spast* mer spesifikk enn *past*, og *table_N_1* mer spesifikk enn *table_N* (8):

l0:_every_q(x0),l1:_fat_j(x1),l2:_cat_n(x2),l3:CONJ,l4:_sit_v_1(e3),l14:on(e4),l9:CONJ,
l5:some(x5),l6:table_n_1(x6),
qeq(h1,l1),qeq(h6,l6),in-g(l3,l1),in-g(l3,l2),in-g(l9,l4),in-g(l9,l14),
RSTR(l0,h1),BODY(l0,h2),RSTR(l5,h6),BODY(l5,h7),ARG1(l4,x2),ARG1(l14,e3),
ARG2(l14,x5), x0=x1, x1=x2, x5=x6

Figur 8 - RMRS8

Vi sitter nå igjen med en underspesifisert semantisk representasjon, der både dype og grunne NLP-teknikker er integrert, nemlig Robust Minimal Recursion Semantics(RMRS).

2.4 RMRS representasjon i SoccerFinder

Vi ser her nærmere på RMRS representasjonene vi får tilbake fra søkevertøyet HoG og hva de forskjellige notasjonene betyr. Som eksempel tar vi for oss en setning fra det første minuttet i kampen Skottland – Norge: ”*Lundekvam vinner duellen til slutt*”. I programkoden vil i tillegg hver RMRS representasjon være presentert som et element i <SoccerFinder_Result> for å markere start og slutt på hver representasjon.

```
<ep>
  <gpred>named_rel</gpred> <label vid="3"/> <var sort="x" vid="4"/>
</ep>
<ep>
  <gpred>def-q-rel</gpred> <label vid="5"/> <var sort="x" vid="4"/>
</ep>
<ep>
  <label vid="8"/> <var sort="e" vid="2" tense="present" delimited="+"/>
  <realpred lemma="vinne" pos="v"/>
</ep>
<ep>
  <label vid="10"/> <var sort="x" vid="9" bounded="+"/> <realpred lemma="duell" pos="n"/>
</ep>
<ep>
  <gpred>def-q-rel</gpred> <label vid="11"/> <var sort="x" vid="9" bounded="+"/>
</ep>
<ep>
  <gpred>card_rel</gpred> <label vid="14"/> <var sort="x" vid="9" bounded="+"/>
</ep>
<ep>
  <gpred>til-slutt_j-rel</gpred> <label vid="10001"/> <var sort="u" vid="15"/>
</ep>
<ep>
  <gpred>prpstn_rel</gpred> <label vid="1"/> <var sort="h" vid="16"/>
</ep>
<rarg>
  <rargname>CARG</rargname> <label vid="3"/> <constant>Lundekvam</constant>
</rarg>
<rarg>
  <rargname>RSTR</rargname> <label vid="5"/> <var sort="h" vid="6"/>
</rarg>
<rarg>
  <rargname>ARG1</rargname> <label vid="8"/> <var sort="x" vid="4"/>
</rarg>
<rarg>
  <rargname>ARG2</rargname> <label vid="8"/> <var sort="x" vid="9" bounded="+"/>
</rarg>
<rarg>
  <rargname>RSTR</rargname> <label vid="11"/> <var sort="h" vid="13"/>
</rarg>
<rarg>
  <rargname>CARG</rargname> <label vid="14"/> <constant>non-plur-rel</constant>
</rarg>
<rarg>
  <rargname>ARG1</rargname> <label vid="10001"/>
  <var sort="e" vid="2" tense="present" delimited="+"/>
</rarg>
```

Figur 9 - RMRS i SoccerFinder

<gpred> named_rel</gpred> vil si at vi har en navnerelasjon. På samme måte angir def_q_rel at vi har en definisjon (det er en ”ting”) og card_rel vil si at vi har kardinalitetsrelasjon. Videre forteller ”label” oss om hvilke noder som hører sammen i xml treet. Den første EP’en har <labell vid = 3> noe som forteller oss at den hører sammen med CARG som også har <labell vid = 3>. ”var”

identifiserer objektet, ”sort” forteller oss om det er et objekt, en begivenhet, udefinert og så videre. ”vid” forteller oss id’en til hver ”var”.

For å vise et eksempel ser vi her at vi har en navnerelasjon, vi har å gjøre med et objekt med id 4, og EP’en hører sammen med en ARG med ”vid” lik 3.

```
<ep>
  <gpred>named_rel</gpred> <label vid="3"/> <var sort="x" vid="4"/>
</ep>
```

Figur 10 - RMRS eksempel A

Vi ser at EP’en hører sammen med CARG som har ”vid = 3”. Objektet er en konstant, nemlig egennavnet Lundekvam.

```
<rarg>
   <rargname>CARG</rargname> <label vid="3"/> <constant>Lundekvam</constant>
</rarg>
```

Figur 11 - RMRS eksempel B

På samme måte kan vi da flette sammen alle nodene i xml-representasjonen.

2.5 Logikkbaserte spørresystem

Når vi skal utvikle et spørresystem er det også en mulighet å bruke logikkbaserte metoder. Det er da naturlig å bruke programmeringsspråket Prolog som bygger på første ordens logikk. Prolog går ut på at utvikleren beskriver problemene ved hjelp av logikk, og gir utrykk for hva som skal utføres framfor hvordan. Et program består av sett med regler, og systemet beregner konsekvensene av disse for å svare på en forespørsel.

Richard Montague(Montague 1970) var den første som introduserte ideen om at naturlig språk kunne beskrives formelt ved hjelp av logikkteknikker. Han presenterte en logikk med muligheter for semantisk tolkning av frie syntaktiske strukturer. Dette blei illustrert i PTQ, også kjent som Montague Grammar(Montague 1973) og i rammeverket ”Universal Grammar” (Montague 1970). Her gjorde en logikk det mulig å uniformt tolke NP’er (Noun Phrases) som *every man, the man, a man* med generaliserte kvantorer.

Arbeidet til Montague er senere blitt videreutviklet, og Pereira og Warren presenterte i 1980 Definite Clause Grammars [Pereira and Warren 1980]. Definite Clause Grammars (DCG) er den mest kjente logikkgrammatikken, og er en innebygd del av de fleste Prolog systemer. DCG er en måte å presentere grammatiske relasjoner som logikkprogram, og kan for eksempel brukes til arbeid i naturlig språk. Som et eksempel kan vi se på en generell DCG(Warren and Schieber 1987):

s(S) --> np(NP), vp(VP). (sentence --> nounphrase, verbphrase.)

Dette vil i prolog se ut som (Warren and Schieber 1987):

```
Sentence(s1, s2) :-
        nounphrase(s1, s3),
        verbphrase(s3, s2).
```

For å vise et eksempel på et logikkbasert spørresystem beskriver vi kort systemet BusTUC (Amble 1998, 2002) som har tatt logikkteknologien til sitt ytterste. BusTUC fungerer i dag som et spørresystem der man ved naturlig språk kan søke etter bussruteopplysninger i Trondheim. BusTUC er å finne på TEAM-trafikk sine nettsider som en web-applikasjon (BussOrakel 2005). Systemet

bygger på TUC, The Understanding Computer, et logikkbasert kunnskapsprosjekt ved NTNU som blei starta i 1991.

For å gå fra spørsmål til svar må BusTUC gjennom flere naturlig språk operasjoner. Hovedkomponentene består av et parsersystem med ordbok, leksikalsk prosessor og grammatikk, en todelt kunnskapsbase(semantisk- og applikasjonskunnskapsbase) og en prosessor for spørringer som inneholder bussrutedata og rutelogikksystem. Systemet er i tillegg tospråklig, da det kan håndtere engelsk og norsk.

Domene til BusTUC består av 420 substantiv, 150 verb, 165 adjektiver, 60 preposisjoner og 1300 grammatikkregler. For å dekke alle stopp, har systemet lagret 3050 stedsnavn i tillegg til alle holdeplasser. Ved å plassere alle substantiver i et "a-kind-of" hierarki i den semantiske kunnskapsdatabasen, kan så holdeplassene relateres til stedsnavnene.

Parsingsystemet til BusTUC er ansvarlig for å gjøre om spørringer til uttalelser (statements). Dette gjøres ved hjelp av grammatikken Consensical Grammar (CONtext SENSItive Categorical Augmented Logic Grammar) som er en enklere generalisering av Definite Clause Grammars. Ved å kjøre setningen "Whose dog barked?" gjennom en slik parser, vil setningen bli forandret som under:

Whose dog barked?
Who has a dog that barked?
Which person has a dog that barked?
For which X is it true that the (x) person has a dog that barked?

Figur 12 - Logikk eksempel A

Den siste linjen vil her analyseres som en uttalelse(statement). Denne uttalelsen gjøres så om til språket TQL(TUC Query Language). Setningen "When does bus 5 leave today?" vil i TQL se slik ut:

```
which (A) : : (5 isa bus, A isa time, leave/5/B,
                event/real/B, srel/intime/time/A/B,
                srel/today/time/nil/B)
```

Figur 13 - Logikk eksempel B

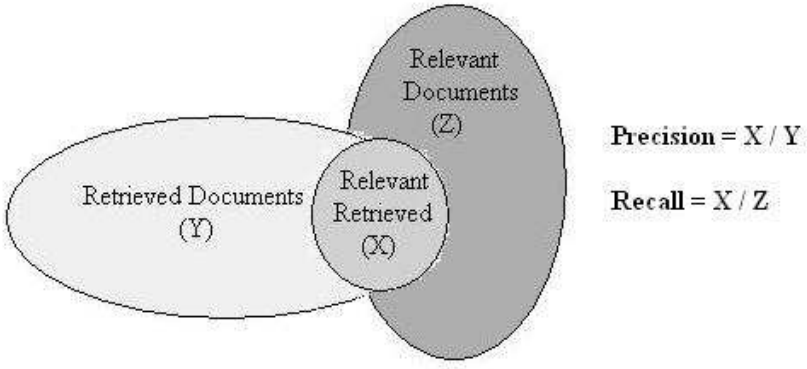
TQL uttrykket er bygd opp av regler som stammer fra uttalelsens semantiske representasjoner. Disse må så kobles til konkrete objekter i databasen med semantiske sammensetninger og ord. Er dette vellykket, har vi funnet et svar. Dette må genereres i naturlig språk, en operasjon med i alt 120 regler, før svaret returneres.

Et slikt logikkbasert spørresystem er foreløpig begrenset til små og stabile domener. Så lenge vi holder oss innefor det begrensede domene til systemet vil vi få pålitelige og intelligente svar. En ulempe kan i enkelte tilfeller være at vi føler vi bruker et "enten-eller" system, det finnes ikke noe rom for tvil. Ting er enten sanne eller usanne, og det vil være vanskelig å lage generelle regler som er sanne eller gale i alle situasjoner. Det er også et veldig utfordrende arbeid å lage en grammatikk for et slikt system. Selv ved å bruke TUC sin generelle grammatikk, vil dette være en omfattende prosess å sette seg inn i.

2.6 Alternativ fremgangsmåte ved bruk av søkeverktøy

En mulig fremgangsmåte for å finne informasjon i fotballreferater kunne være å bruke et søkeverktøy, for eksempel Google (Google 2005). Et slikt søkeverktøy benytter seg av IR for å hente ut informasjon fra samlinger av dokumenter.

IR, Information Retrieval, er en teknikk der man forsøker å finne relevante dokumenter på bakgrunn av en brukers spørringer. IR systemer bygger på en veldig enkel fremgangsmåte der man rett og slett finner ord i dokumenter og matcher disse med brukeren sin spørring. Dette gjøres statistisk ved å sjekke om for eksempel tekststrenger matcher, meningen med teksten er lik, det samme vokabularet brukes og så videre. Vi kan også si at IR teknikker er heuristiske da vi ikke vet hva som er riktig svar på et søk. Søket går derfor ut på å komme så nært som mulig det riktige svaret. Vi har i IR noe som kalles ”precision” og ”recall”. Precision er delen relevante dokumenter i forhold til totale antallet dokumenter. Recall er relevante dokumenter vi finner fram til i forhold til den totale samling relevante dokumenter.



Figur 14 - IR - precision \ recall

Søkeverktøyet må altså ha en mengde dokumenter tilgjengelige for gjenfinning, og får tak i disse ved å sende ut en såkalt ”spider” på nettet. Neste steg er å foreta en indeksering av de aktuelle dokumentene. Dette gjøres fordi en sekvensiell gjennomsøking av dokumentene rett og slett vil ta for lang tid. En indeksering vil si at vi oppretter en liste med ord som alle har pekere til sidene de er hentet fra. En liten beskrivelse av det aktuelle dokumentet lagres også gjerne i indeksen så det skal være enklere å finne dokumentet man er på utkikk etter.

Et søkeverktøy som Google utfører også en fjerning av stoppord fra spørringer som skal kjøres. Dette vil si at ord som forekommer veldig ofte, og ikke har noen innvirkning på meningen, fjernes.

Når søkeverktøyet nå skal utføre et søk på bakgrunn av vår spørring, gjør den dette ved å søke i indeksene som er blitt produsert. Etter å ha funnet dokumenter som inneholder søkeordene, blir disse sendt til rangering for å finne de dokumentene som på best mulig vis besvarer spørringen. Resultatet sendes så tilbake til brukeren, typisk resultatsiden til Google, som en liste linker.

En fremgangsmåte for oss er å indeksere et fotballreferat for så å søke etter interessant informasjon. Dette kan vi gjøre ved å laste ned og installere Google Desktop Search (Google Desktop 2005) som indekserer dokumenter på datamaskinen vår. Vi kan for eksempel gjøre et søk der vi prøver å finne ut om Steffen Iversen scoret et mål i kampen. Vi kan da søke på ordene <Iversen mål>.

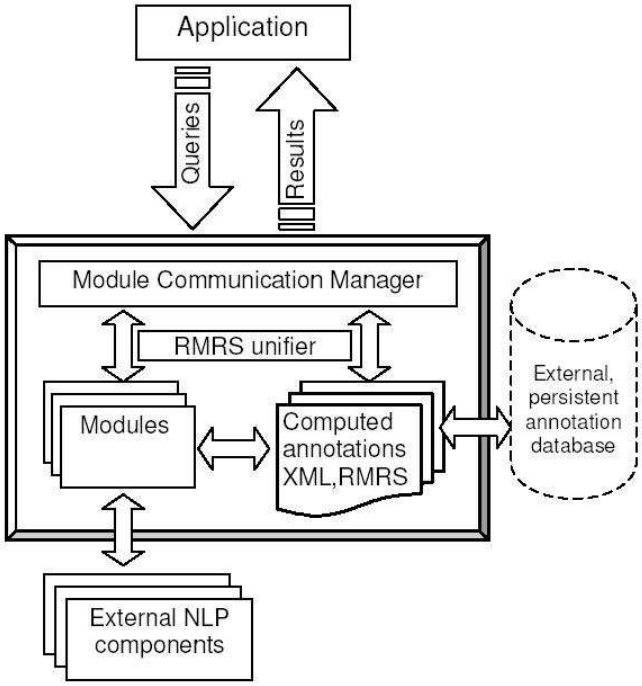
3.0 Systemomgivelser

3.1 The DeepThought Core Architecture Framwork – Heart of Gold (HoG)

HOG (Callmeier et al. 2004) er en infrastruktur for å utvikle applikasjoner som bruker RMRS-baserte (Copestake 2003) naturlig språk komponenter. Kjernen er skrevet i Java, men det er også mulig å bruke andre språk for å utvikle komponenter og applikasjoner.
Hovedmålene og egenskapene ved HOG er blant annet:

- Fleksibel integrering av NLP (Natural Language Processing) komponenter.
- Kommunikasjonen er plattform- og programmeringsspråkuavhengig da XML-RPC brukes.
- Presentasjon av resultater skjer i form av RMRS-representasjoner på XML format.
- Baserer seg på teknologier som XML, XML:DB, XPath, XML-RPC og XSLT.

Den generelle arkitekturen går ut på at HOG fungerer som et mellomledd mellom applikasjoner og komponenter. Applikasjonene sender forespørsler til HOG som prosesserer disse på bakgrunn av konfigurasjonen. Resultatet sendes så tilbake til applikasjonen. Under er vist en figur over oppbygningen av HOG.

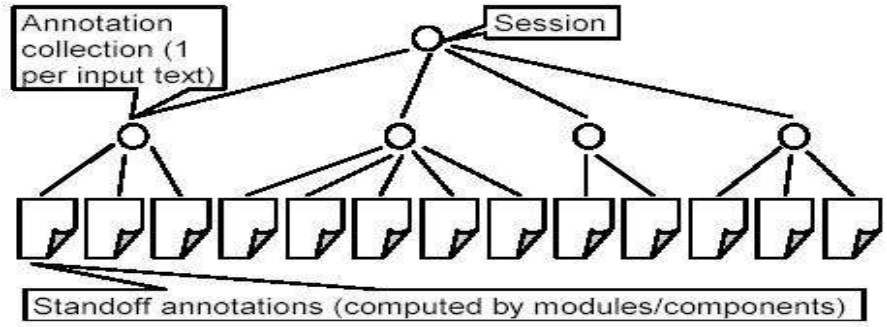


Figur 15 - HoG (Callmeier et al. 2004)

HOG består av en Module Communication Manager (MoCoMan) som formidler kommunikasjonen mellom applikasjoner og NLP-komponentene. MoCoMan mottar en ordre (typisk tekstdokumenter eller setninger) fra en applikasjon, sender denne til de konfigurerte komponentene, får tilbake en analyse av teksten og returnerer denne til applikasjonen. MoCoMan er også ansvarlig for rekkefølgen de forskjellige komponentene trigges i.

Til å begynne med startes en instans av HOG med nødvendig konfigurasjonsdata for de forskjellige komponenter. MoCoMan starter så disse komponentene, det vil si eksisterende NLP programmer. Siden det er mulig å utvikle komponenter i andre språk enn Java, har HoG også definert en XML-RPC klasse. Denne brukes til å koble seg til komponenter programmert i andre språk, for eksempel på andre servere.

MoCoMan har også en ”Session Manager” som gjør at forskjellige økter med flere inndokumenter kan behandles. MoCoMan håndterer økter som hver består av en samling annotasjoner. En slik annotasjon samsvarer med en tekst eller et dokument som har kommet inn, og består av beregnede RMRS annotasjoner.



Figur 16 - Session Manager (Callmeier et al. 2004)

Økter med annotasjonssamlinger kan også lagres i en XML database. Hensikten med en slik database er hovedsakelig å kunne lagre og håndtere et stort antall XML dokumenter, og det er støtte for innsetting og sletting samt spørringer basert på XPath.

Etter at HOG er konfigurert, sendes forespørsler bestående av en tekst og integer som graderer analysen til MoCoMan. Disse formidles videre til de konfigurerte komponentene som svarer med å sende tilbake beregnede RMRS annotasjoner til applikasjonen. HOG gir oss tilbake RMRS-representasjoner på XML-format ved et vellykket søk.

3.2 XML-RPC

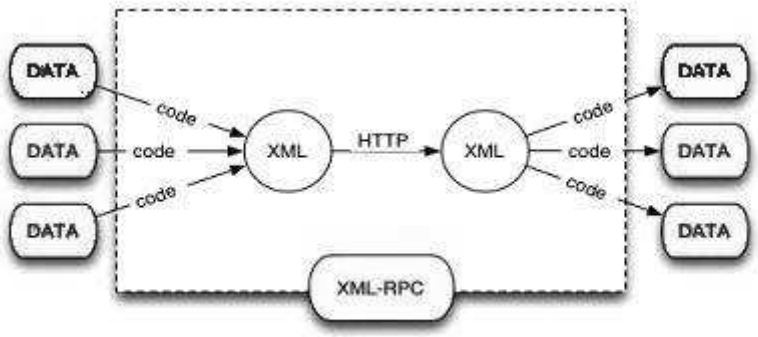
XML-RPC (Xml-Rpc 2005) er en metode for enkelt å kunne gjøre kall på ikke-lokale prosedyrer, for eksempel over nettet mot en server. XML-RPC kan sies å være en ny måte å bruke eksisterende teknologi på. XML-RPC støttes av de fleste programmeringsspråk og kan brukes på forskjellige plattformer. XML-RPC benytter HTTP(HyperText Transfer Protocol) som transportprotokoll og kan derfor enkelt implementeres på de fleste webservere.

XML (XML 2005) står for eXtensible Markup Language og er en teknologi for å lagre og strukturere data. XML minner om HTML(HyperText Markup Language), men inneholder ingen informasjon om hvordan man ønsker å presentere dataene. Dette kan en derimot få til ved hjelp av XSL, eXtensible Stylesheet Language, ved at man linker XML-dokumentet opp til en XSL fil som beskriver hvordan dataene skal presenteres. En annen egenskap som skiller XML fra HTML er at brukeren definerer egne ”Tags” som gjerne er beskrivende for informasjonen vi lagrer. Det er i tillegg viktig å merke seg at XML ikke erstatter HTML, det er et tillegg.

RPC (Remote Procedure Call) er en klient-server modell som benytter seg av at operasjoner som utføres i en datamaskin er resultater av kall på prosedyrer. RPC tar dette et steg videre og lar oss utføre prosedyrekall på andre fjerne maskiner. Det opprettes en forbindelse mellom prosedyrer som kjøres på forskjellige maskiner.

XML-RPC er altså en måte å sende, samt motta XML beskjeder for å utføre kall på prosedyrer både lokalt og fjernt. En XML-RPC ordre(request) er en http-post beskjed som spesifiserer metodenavn samt aktuelle parameter i XML-format. På samme måte er svaret en http-response melding med en returverdi, også på XML-format. Så lenge det ikke er skjedd en feil, sendes det i tillegg med en http-

200 OK melding med returverdien. Ved å bruke XML for strukturering av data oppnår vi at XML-RPC er plattformuavhengig.



Figur 17 - XML-RPC (XML-RPC 2005)

XML-RPC kan håndtere en del av de vanligste datatyper, som for eksempel int, boolean, string, double og struct. En av hovedtankene bak XML-RPC er enkelhet. For å lage en enkel XML-RPC klient må man generelt gjennom 4 steg. Man må opprettes et klient-objekt, konstruere en forespørsel og sende denne anmodningen til serveren. Responsen fra serveren må så bearbeides.

I vårt system utfører vi RPC kall over nettet og må derfor bruke en XML-RPC proxyklasse. Vi har satt opp en XML-RPC klient og sender forespørsler til serveren HoG.

For å bruke XML-RPC bruker vi Java implementasjonen til apache. Her finnes eksempler på server og klient implementasjon samt all dokumentasjon rundt prosjektet. (Apache XML-RPC 2005)

4.0 Konstruksjonen av systemet

Under følger en beskrivelse av systemet. Vi starter med å presentere modeller i UML (UML 2005). Videre går vi inn på den tekniske delen og en beskrivelse av metoder og klasser legges fram. Vi forklarer også de forskjellige XSLT transformasjoner som utføres. Til slutt har vi tatt med en liten beskrivelse av utviklingsverktøyene som er brukt i utformingen av systemet vårt SoccerFinder.

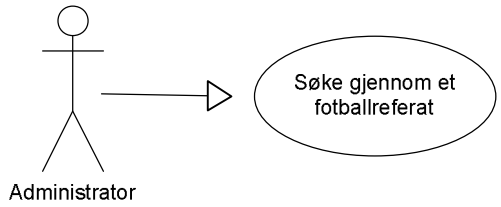
4.1 UML-beskrivelse

Vi viser her Use Case beskrivelser, sekvensdiagrammer samt klassediagram over det implementerte systemet. Til slutt har vi også valgt å ta med et deploymentdiagram for å vise hvordan systemets deler er fordelt på maskinvaren. Dette gir et fint overblikk over de forskjellige elementene i systemet vårt.

4.1.1 Use Case Modeller

Vi har laget to Use Caser over systemet vårt. Use Case 1 viser hendelsesforløpet når en bruker ønsker å analysere en ny fotballsetning og Use Case 2 når vi ønsker å kjøre en spørring mot databasen.

Use Case 1

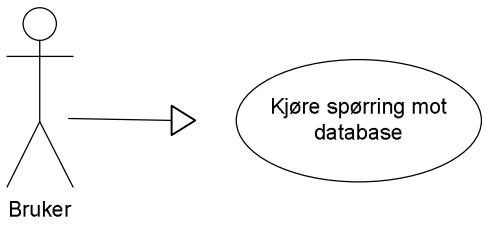


Hendelsesforløp

- Administratoren skriver inn teksten han ønsker å søke gjennom.
- Inndata parses om til xml-format for å kunne sendes.
- En proxy klasse må opprettes, og en XML-RPC klient må initialiseres før systemet kontakter HOG. Hvis kontakt med HOG feiler, mottar vi en feilmelding og søket avsluttes.
- HOG mottar en request fra systemet og kjører metodene som følger med som argumenter med XML-RPC kallet.
- RMRS strukturer blir konstruert ved hjelp av NLP komponentene HoG gjør seg nytte av, og lagres på xml-format.
- RMRS strukturer på xml-format sendes så tilbake til SoccerFinder ved hjelp av XML-RPC.
- Systemet må nå hente ut data som er interessante for oss fra XML-dokumentet. Vi filtrerer derfor dokumentet ved å benytte XSLT.
- De filtrerte RMRS representasjonene legges inn i databasen sammen med originalsetningen ved å bruke JDBC så de kan hentes frem seinere.

Figur 18 - Use Case 1

Use Case 2



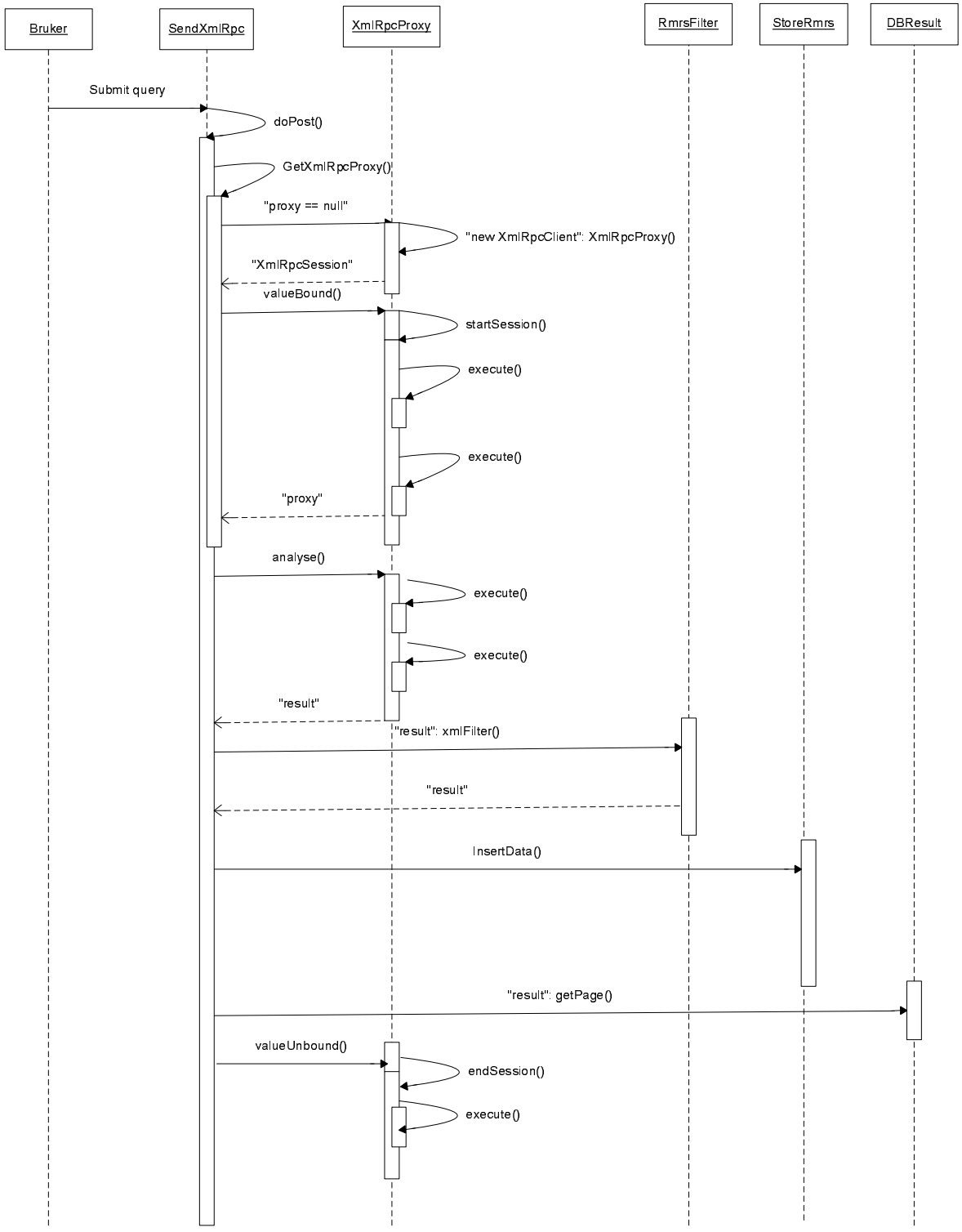
Hendelsesforløp

- Brukeren velger ved hjelp av menyen den spørringen han ønsker å kjøre mot databasen.
- Systemet må så opprette kontakt med databaseserveren før spørringer kan utføres. Dette gjøres ved hjelp av JDBC og utføres i to steg.
- Drivere som vi ønsker å bruke, må lastes før vi oppretter kontakt, blant annet ved hjelp av metoden getConnection().
- Metoden executeQuery() kan så brukes for å kjøre spørringer mot databasen.
- getString() brukes for å hente resultatet fra spørringen.
- Resultatet sendes tilbake fra databaseserveren til servleten og vises i web-browseren.

Figur 19 - Use Case 2

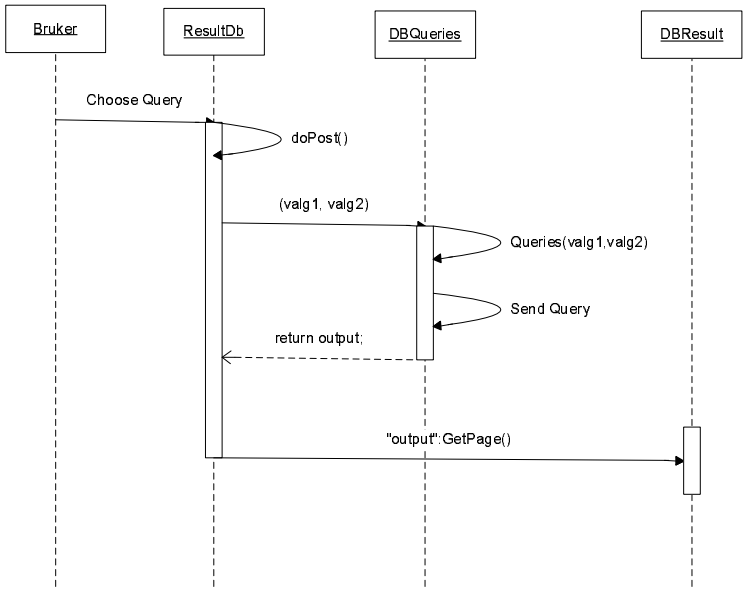
4.1.2 Sekvensdiagrammer

Vi har modellert to sekvensdiagrammer som korresponderer med hver sin Use Case. Det første sekvensdiagrammet viser gangen i systemet fra en bruker skriver inn en setning han \ hun vil analysere, til resultatet er skrevet ut og sesjonen med HoG avsluttes.



Figur 20 - Sekvensdiagram 1

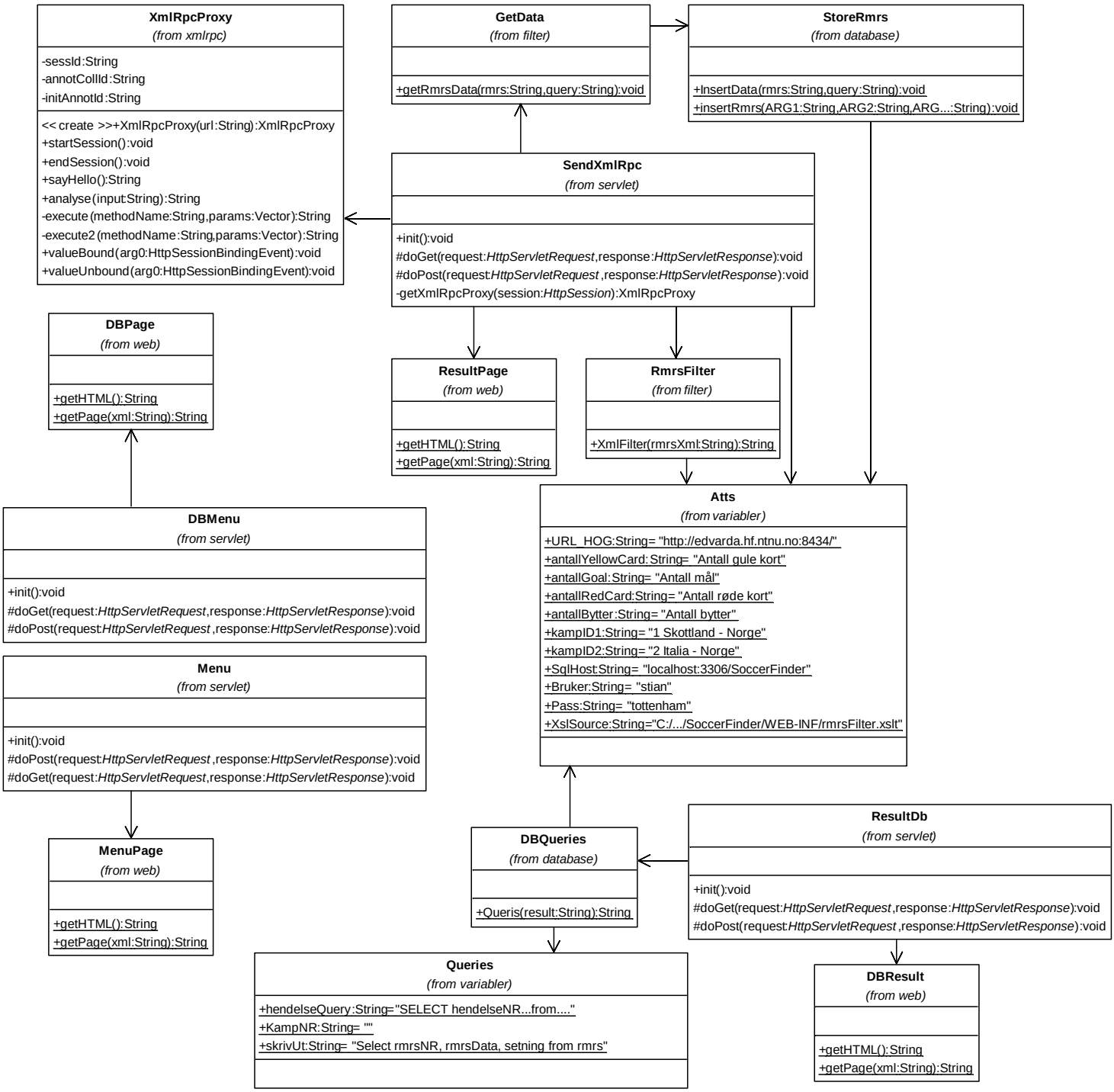
Vårt andre sekvensdiagram viser gangen i systemet når en bruker velger en spørring han \ hun ønsker å kjøre mot databasen.



Figur 21 - Sekvensdiagram 2

4.1.3 Klassediagram

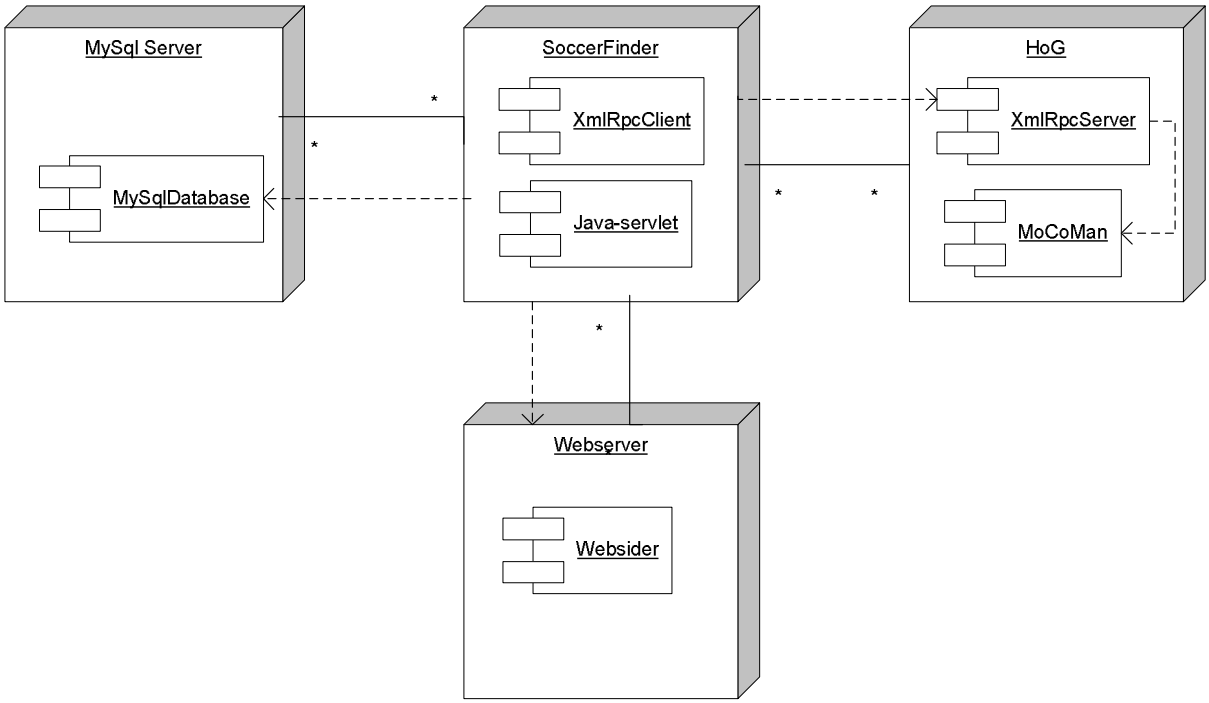
Figuren under viser klassediagrammet for systemet vårt. En nærmere beskrivelse av de forskjellige klassene er beskrevet i kapittel 4.2.



Figur 22 - Klassediagram

4.1.4 Deploymentdiagram

For å vise en oversikt over komponentene i systemet vårt har vi valgt å ta med et enkelt deploymentdiagram.



Figur 23 - Deploymentdiagram

Det er viktig å merke seg at selv om MySqlserveren, SoccerFinder og webserveren på tegningen befinner seg på forskjellige noder, står kjøres de i virkeligheten på samme maskin. Vi har valgt å representere dem på forskjellige noder på tegningen for oversiktens skyld.

4.2 Teknisk Beskrivelse av systemet

Vi beskriver i dette kapittelet oppbygningen av systemet og forklarer samtlige klasser og metoder. For oversiktens skyld er klassene med tilhørende metoder forklart i hvert sitt underkapittel. Til slutt forklares web.xml fila som er nødvendig når vi bruker Java servleter.

4.2.1 SendXmlRpc

init()
Første metoden som kjøres når web-serveren laster inn servleten.
doGet()
Standard servlet metode for å håndterer http forespørsler.
doPost()
Standard servlet metode for å håndtere http forespørsler. Metoden henter inn teksten vi ønsker å analysere og starter getXmlRpcProxy(). Metoden analyse() i [XmlRpcProxy] kjøres så med setningen vår. Resultatet sendes så til filtrering i metoden xmlFilter() før den filtrerte representasjonen lagres i en databasetabell.
getXmlRpcProxy
Metoden initierer en proxy klasse vi er avhengig av når vi skal kjøre rpc kall over nettet. Metoden xmlRpcProxy kjøres med url'en til XML-RPC serveren som i vårt tilfelle er HoG.

Figur 24 - SendXmlRpc

4.2.2 DBMenu

init()
Første metoden som kjøres når web-serveren laster inn servleten.

DoGet()
Standard servlet metode for å håndterer http forespørsler.
doPost()
Standard servlet metode for å håndterer http forespørsler. Kjører metoden getHTML() og skriver ut resultatet for å få fram grensesnittet til databasemenyen.

Figur 25 - DBMenu

4.2.3 Menu

init()
Første metoden som kjøres når web-serveren laster inn servleten.
DoGet()
Standard servlet metode for å håndterer http forespørsler.
doPost()
Standard servlet metode for å håndterer http forespørsler. Kjører metoden getHTML() og skriver ut resultatet for å få fram hovedmemyen.

Figur 26 - Menu

4.2.4 ResultDb

init()
Første metoden som kjøres når web-serveren laster inn servleten.
DoGet()
Standard servlet metode for å håndterer http forespørsler.
doPost()
Standard servlet metode for å håndterer http forespørsler. Henter inn valgene som gjøres i databasemenyen og lagrer disse. Kjører deretter metoden Queries() for å kjøre ønskede spørringer, og skriver til slutt ut resultatet fra databasen..

Figur 27 - ResultDb

4.2.5 MenuPage, DBPage og ResultPage

Alle disse klassene er likt bygd opp ved at de inneholder HTML kode for de forskjellige sidene i brukergrensesnittet vårt. Servletene får tak i koden ved å kalle getHTML() metodene.

Figur 28 - MenuPage, DBPage, DBResult og ResultPage

4.2.6 XmlRpcProxy

XmlRpcProxy()
Oppretter en ny XML-RPC klient. Kjører deretter metoden sayHello() for å sjekke om vi oppnår kontakt med HoG eller ikke.
startSession()
Kjører createSession på HoG ved å kjøre execute() metoden vår. Dette gjøres for å få tilbake vår sessionID som må brukes i påfølgende HoG-metoder. Vi oppretter en vektor med plasseringen til konfigurasjonsfilen for norsk språk i HoG, og kjører execute() med metode og argumenter. Vi har nå fått en verdi på sessionID og bruker denne videre for å kjøre neste metode på HoG som er createAnnotationCollection. Vi har på samme måte en vektor med metodenavn og argumentet som nå er sessionID. Vi oppretter her en ny annotasjonssamling som vil ta vare på alle annotasjoner HoG produserer for tekstene vi ønsker å analysere. Vi får tilbake annotationCollectionID.

endSession()
Vi opretter en vektor der vi legger til sessionID. Kjører så xmlrpcmetoden execute() med metodenavnet closeSession og vektoren vår. HoG-session'en vår blir så avsluttet.
sayHello()
Kjører execute() metoden vår med HoG-metoden sayHello og en tom vektor.
analyse()
Oppretter en vektor med parameterne sessID, annotCollID, teksten vår, språk-koden for norsk, samt navnet på klienten vår (SoccerFinder). Kjører så execute() metoden vår med HoG metoden createInitialAnnotation og vektoren som parameter. Metoden gir oss tilbake en annotasjons id, initAnnotId, "xmltext". Vi kjører så den siste metoden, analyse, på HoG ved å bruke execute() metoden for å få tilbake et resultat. Vi har som alltid med en vektor med blant annet sessionID, annotationCollectionID og initAnnotId. HoG gir oss nå tilbake en rmrsrepresentasjon på xml format, eventuelt null hvis setningen vår ikke kunne analyseres.
execute()
Dette er metoden vi kjører hver gang vi ønsker å sende en metode med argumenter til HoG. Vi kjører her XML-RPC sin execute() metode med argumenter som vi har spesifisert. Metoden returnerer resultatet fra HoG.
valueBound()
Denne metoden startes automatisk ved hjelp av HttpSessionBindingListener, og setter i gang startSession(). Starter automatisk ved at *event* informerer objektet om at den er bundet til en session som den identifiserer.
valueUnbound()
Denne metoden startes automatisk ved hjelp av HttpSessionBindingListener, og setter i gang endSession(). Starter automatisk ved at *event* informerer objektet om at den er bundet til en session som den identifiserer.

Figur 29 - XmlRpcProxy

4.2.7 Atts

Dette er en klasse som inneholder en samling konstanter som brukes i programkoden. Vi finner her blant annet kilden til xsl-transformasjonen, database brukerinformasjon og plasseringen av søkeverktøyet HoG. Hvis man skal forandre på oppsettet, er det mye enklere å gjøre det en gang her, istedenfor hardkoding i hele programmet.

Figur 30 - Atts

4.2.8 Queries

På samme måte som Atts klassen, inneholder Queries ferdige templater på spørringer som skal kjøres mot en database.

Figur 31 - Queries

4.2.9 RmrsFilter

XmlFilter()
Denne metoden er ansvarlig for å filtrere bort informasjonen som ikke er interessant for oss i resultatet vi får fra HoG. Metoden henter inn RMRS-representasjonen, xslt-dokumentet og bruker TransformerFactory til å kjøre xslt dokumentet på representasjonen vår. Resultatet gjøres om til en tekststreng og returneres.

Figur 32 – RmrsFilter

4.2.10 GetData

getRmrsData()
Denne metoden starter henholdsvis metodene insertRmrs og insertData metodene for å få lagret rmrs-dataene våre.

4.2.11 StoreRmrs

InsertData()
Metoden henter inn argumentvariablene vi har fått etter å ha filtrert rmrs-representasjonen i detalj og legger disse inn i databasetabeller.
InsertRmrs()
Metoden henter søketeksten vår samt det filtrerte resultatet og legger dette inn i database-tabellen rmrs.

Figur 33 - StoreRmrs

4.2.12 DBQueries

Queries()
Denne metoden henter inn valgene våre fra databasemenyen og velger ut i fra dette hvilken spørring vi ønsker å kjøre mot databasen. Resultatet returneres og skrives så ut i servleten ResultDb.

Figur 34 - DBQueries

4.2.13 <web.xml>

<webapp> </webapp>
Inneholder alle innstillinger for servletene.
<servlet> </servlet>
Inneholder navnet på servleten med en beskrivelse av dens funksjon. Vi har også her servletklassen vi linker opp mot.
<servlet-mapping> </servlet-mapping>
Mappingen knytter den enkelte servlet opp mot hvilke navn i url appliksjonen skal reagerer på.

Figur 35 - <Web.xml>

Som eksempel kan vi se på informasjonen for ”ResultDb” servleten:

```
    <servlet>
       <servlet-name>ResultDb</servlet-name>
       <display-name>ResultDb</display-name>
       <description>Displays results from query</description>
       <servlet-class>servlet.ResultDb</servlet-class>
    </servlet>
    <servlet-mapping>
       <servlet-name>ResultDb</servlet-name>
       <url-pattern>/displayDB</url-pattern>
    </servlet-mapping>
```

Figur 36 - <web.xml> eksempel

Vi ser at servleten heter ResultDb og at dens oppgave er å presentere resultatet fra en spørring. Klassen den er linket opp til er ”ResultDb” som ligger i pakken ”servlet”. Videre er ”url-triggeren” for servleten ”/displayDB”.

4.3 XSLT i SoccerFinder

Vi kjører i programmet vårt flere transformasjoner av representasjonene vi får tilbake fra HoG. Dette må gjøres for at vi skal være i stand til å legge inn ønskede data på riktige plasser i en databasetabell. Vi forklarer her samtlige av disse med tilhørende eksempler.

4.3.1 Filtrering av interessante data

Den første transformasjonen som utføres på resultatet fjerner rett og slett all info som for oss ikke er interessant. XML-treet vi sitter igjen med er på samme format som det som er beskrevet i kapittel 2.4. Under viser vi XSLT dokumentet som utfører denne operasjonen.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xalan="http://xml.apache.org/xslt">

  <xsl:output method="xml"
              encoding="UTF-8"
              indent="yes"
              xalan:indent-amount="2"/>

       <xsl:template match="/">
              <SoccerFinder_result>
              <xsl:for-each select="/pet/rmrs/ep">
                     <ep>
                     <xsl:copy-of select="gpred"/>
                     <xsl:copy-of select="label"/>
                     <xsl:copy-of select="var"/>
                     <xsl:copy-of select="realpred"/>
                     </ep>
              </xsl:for-each>

              <xsl:for-each select="/pet/rmrs/rarg[rargname != 'BODY']">
                     <rarg>
                     <xsl:copy-of select="rargname"/>
                     <xsl:copy-of select="label"/>
                     <xsl:copy-of select="var"/>
                     <xsl:copy-of select="constant"/>
                     </rarg>
              </xsl:for-each>

              </SoccerFinder_result>
       </xsl:template>

</xsl:stylesheet>
```

Figur 37 - XSLT rmrsFilter.xslt

Filen starter med standard informasjon om rettigheter og versjoner av forskjellig teknologi. Videre har dokumentet <xsl:template..> elementer som har i oppgave å finne mønster i xml dokumentet. Den første templaten består av '/' som forteller at vi tar for oss hele xml-dokumentet. Vi setter deretter inn et element, *<SoccerFinder_result>*, som vi bruker til å markere start og slutt på dataene. Videre plukker vi ut de elementene vi ønsker å ta med oss videre til resultatdokumentet. Dette gjør vi ved å fortelle at vi i grenen */pet/rmrs/ep* ønsker å kopiere med feltene *gpred*, *labell*, *var*, og *realpred*. På samme måte forteller vi at vi i */pet/rmrs/rarg* vil ta med feltene *rargname*, *label*, *var* og *konstant*. Vi har i tillegg spesifisert at vi ikke vil ha med felter fra noden *BODY*. Fila avslutter til slutt elementene *SoccerFiner_result*, *xsl:template* og *xsl:stylesheet*.

Neste steg går ut på å trekke ut den interessante informasjonen fra RMRS dokumentet. Vi har her utviklet to framgangsmåter, en for korte og greie setninger, men også en som egner seg på lenger setninger der flere personer er involvert. Selv om den siste fremgangsmåten også vil fungere på korte setninger med kun en aktør, forklarer vi her begge. Vi har også brukt begge i systemet da en kort setning transformert ved fremgangsmåten for kompliserte setninger vil ta opp unødig plass i en databasetabell.

4.3.2 Filtrering av hendelse – fremgangsmåte 1

Den første fremgangsmåten er delt opp i flere operasjoner, og vi bruker her 3 transformasjoner for å finne en hendelse. Transformasjonene går ut på å se sammenhenger mellom ARG1, ARG2, CARG og resten av RMRS-representasjonen. Vi ser først på ”rmrsFilterNames.xslt”:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xalan="http://xml.apache.org/xslt">

       <xsl:template match="/">
              <xsl:for-each select="

           /SoccerFinder_result/rarg[rargname = 'CARG'and not(contains(constant, '-rel'))] ">

                     <xsl:value-of select="constant" />
              </xsl:for-each>
       </xsl:template>
</xsl:stylesheet>
```

Figur 38 - XSLT – rmrsFilterNames.xslt

Filen starter med standard informasjon om rettigheter og versjoner. Som i den forrige fila starter vi med å fortelle at vi vil se på innholdet i hele xml-dokumentet. Videre sier vi at vi i *SoccerFinder_result/rarg* ønsker å hente ut data fra noden *rargname* når den har verdi *CARG* samt når noden *constant* ikke inneholder en verdi lik *–rel*. Hvis dette er tilfelle, henter vi så ut verdien fra noden *constant* som i eksempelet vårt i kapittel 2.4 ville vært *Lundekvam*. Denne fremgangsmåten vil alltid kunne gi oss tilbake egennavn i en RMRS representasjon.

Vi forklarer videre fila ”rmrsFilterArg1.xslt” som går et steg videre ved å kjenne igjen en begivenhet.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
       <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
       <xsl:template match="/">

       <xsl:for-each select="/SoccerFinder_result">

       <xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'ARG1']">

              <xsl:variable name="vid">
                            <xsl:value-of select="label/@vid"/>
              </xsl:variable>

              <xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $vid]">
                     <xsl:if test="realpred/@lemma">
                            <xsl:value-of select="realpred/@lemma" />
                     </xsl:if>
              </xsl:for-each>
       </xsl:for-each>
       </xsl:for-each>

       </xsl:template>
</xsl:stylesheet>
```

Figur 39 - XSLT rmrsFilterArg1.xslt

Vi starter med å si at her skal vi se på noden *rargname* der den har verdi *ARG1*. Videre oppretter vi variabelen *vid* og gir denne verdien til feltet *vid* i noden *label*. Vi sier så at vi vil finne elementet *SoccerFinder_result/ep* der *vid* har den samme verdien som den vi fant for *ARG1*. Vi sjekker så om vi på denne noden finner en *realpred* med verdi *lemma*. Hvis dette er tilfellet, tar vi med denne verdien, *realpred/@lemma*. Ved å kjøre denne XSLT fila på RMRS eksempelet vårt (se kapittel 2.4) vil vi finne igjen følgende informasjon(vi viser kun den delen av XML treet som er relevant i dette tilfelle):

```
...
<ep>
  <label vid="8"/> <var sort="e" vid="2" tense="present" delimited="+"/>
  <realpred lemma="vinne" pos="v"/>
</ep>
...
<rarg>
  <rargname>ARG1</rargname> <label vid="8"/> <var sort="x" vid="4"/>
</rarg>
...
```

Figur 40 - XSLT rmrsFilterArg1 resultat

Som vi ser av eksempelet over finner vi først at *ARG1* har *label vid* lik *8*. Deretter finner vi *ep* med matchende *vid = 8*, og sjekker så om denne inneholder *realpred*. Hvis dette er sant, tar vi med oss verdien, i dette tilfelle *vinne*.

Videre skal vi så finne neste argument ved en nesten identisk fremgangsmåte. Eneste forskjellen er her at vi tar utgangspunkt i noden med verdi *ARG2*, og sjekker på *var vid* sin verdi. Vi vil i RMRS representasjonen da finne igjen dataene som vist under.

```
...
<ep>
  <label vid="10"/> <var sort="x" vid="9" bounded="+"/> <realpred lemma="duell" pos="n"/>
</ep>
...
<rarg>
  <rargname>ARG2</rargname> <label vid="8"/> <var sort="x" vid="9" bounded="+"/>
</rarg>
...
```

Figur 41 - XSLT rmrsFilterArg2 resultat

Som vi ser av eksempelet over finner vi først at *ARG2* har *var vid* lik *9*. Deretter finner vi *ep* med matchende *vid = 9*, og sjekker så om denne inneholder *realpred*. Hvis dette er sant, tar vi med oss verdien, i dette tilfelle *duell*.

Vi har nå nok informasjon til å relatere en fotballspiller til en hendelse med et gitt utfall. Dette plasseres videre i en databasetabell for å ta være på resultatet fra et søk.

Det er også mulig å få med tidsbegrepet, *til slutt*, i eksempelsetningen vår. Hvordan dette blir gjort illustrerer vi i eksempelet under, ”rmrsFilterTime.xslt”.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
	<xsl:template match="/">

	<xsl:for-each select="/SoccerFinder_result">

	<xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'ARG1']">

		<xsl:variable name="vid2">
			<xsl:value-of select="label/@vid"/>
		</xsl:variable>

		<xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $vid2]">

			<xsl:value-of select="gpred" />

		</xsl:for-each>
```

```
        </xsl:for-each>
        </xsl:for-each>

        </xsl:template>
</xsl:stylesheet>
```

Figur 42 - XSLT rmrsFilterTime.xslt

Dette gjøres som tidligere ved at vi ser på noden *rargname* med verdi *ARG1*. Videre oppretter vi variabelen *vid2* og gir denne verdien til feltet *vid* i noden *label*. Vi sjekker så om vi finner en node *gpred*, og tar med verdien hvis den finnes. Vi kjører så fila på eksempelet vårt, og finner *til-slutt_j-rel*.

```
...
<ep>
  <gpred>til-slutt_j-rel</gpred> <label vid="10001"/> <var sort="u" vid="15"/>
</ep>
...
<rarg>
  <rargname>ARG1</rargname> <label vid="10001"/>
  <var sort="e" vid="2" tense="present" delimited="+"/>
</rarg>
...
```

Figur 43 - XSLT rmrsFilterTime resultat

Transformasjonene som vi har forklart over vil fungere på enkle og korte setninger som eksempelsetningen vår ”Lundekvam vinner duellen til slutt”.

4.3.3 Filtrering av hendelse – fremgangsmåte 2

For å kunne takle en lengre setning med flere aktører har vi laget regler for en mer avansert transformasjon. Vi går her igjennom en hel RMRS representasjon, og henter ut hele hendelser med aktuelle aktører. Koden fra fila ”rmrsFilterAlt.xslt” er å finne under Vedlegg E, og vi har tatt med deler av den her for å forklare gangen i transformasjonen.

```
...
<xsl:for-each select="…/rarg[rargname = 'CARG' and not(contains(constant, '-rel'))] ">
...
               <xsl:value-of select="constant" />

               <xsl:variable name="labelvid1">
                      <xsl:value-of select="label/@vid" />
               </xsl:variable>
...

       <xsl:for-each select="…/ep[label/@vid = $labelvid1]">

               <xsl:variable name="varvid1">
                      <xsl:value-of select="var/@vid"/>
               </xsl:variable>
...
               <xsl:for-each select="…/rarg[rargname='ARG1' and var/@vid = $varvid1]">

               <xsl:variable name="labelvid2">
                      <xsl:value-of select="label/@vid"/>
               </xsl:variable>
...
               <xsl:for-each select="…/ep[label/@vid = $labelvid2]">

                      <xsl:if test="realpred/@lemma">
                             <xsl:value-of select="realpred/@lemma" />
                      </xsl:if>
```

```
                </xsl:for-each>
…
                <xsl:for-each select="…/rarg[rargname='ARG2' and label/@vid = $labelvid2]">

                        <xsl:variable name="varvid2">
                                <xsl:value-of select="var/@vid" />
                </xsl:variable>
…
                <xsl:for-each select="…/ep[var/@vid = $varvid2]">

                        <xsl:if test="realpred/@lemma">
                                <xsl:value-of select="realpred/@lemma" />
                        </xsl:if>

        </xsl:for-each>
…
…
        </xsl:template>
</xsl:stylesheet>
```

Figur 44 - rmrsFilterAlt.xslt

Vi starter med å finne *rargname* node med verdi *CARG*, og kopierer innholdet. Vi deklarerer så en ny variabel, *labelvid1*, og gir denne verdien til *label*. Vi finner videre en *ep* node med matchende verdi på *label*. En ny variabel må så deklareres, *varvid1*, og vi tilegner denne verdien til *var* sin *vid*. Neste node vi finner frem til er så en *ARG1* med *var vid* verdi lik *varvid1*. Vi deklarerer så nok en variabel, *labelvid2*, som får samme verdi som den aktuelle *label*. Vi finner nok en gang matchende *ep* node med *labell vid* verdi lik *labelvid2*, og sjekker om den inneholder et felt lik *realpred*. Hvis dette er tilfelle tar vi vare på denne verdien.

Vi finner videre en node *ARG2* med *label* verdi lik *labelvid2*. *varvid2* deklareres så og gis samme verdi som *var*. Til slutt leiter vi fram en *ep* med verdi på *var* lik *varvid2*, sjekker om det her finnes en *realpred*, og tar vare på denne.

Vi viser under hvordan vi ved dette kan finne igjen informasjon i den mer kompliserte RMRS'en til setningen "Lundekvam spiller ballen til Magne Hoseth som mister ballen til James McFadden.". Vi har gjort om på rekkefølgen de forskjellige data presenteres i så det skal være lett å følge tankegangen vår. Se vedlegg C for komplett RMRS av setningen.

```
…
<rarg>
  <rargname>CARG</rargname>
  <label vid="3"/>
  <constant>Lundekvam</constant>
</rarg>
…
<ep>
  <gpred>named_rel</gpred>
  <label vid="3"/>
  <var sort="x" vid="4"/>
</ep>
…
<rarg>
  <rargname>ARG1</rargname>
  <label vid="8"/>
  <var sort="x" vid="4"/>
</rarg>
…
<ep>
  <label vid="8"/>
  <var sort="e" vid="2" tense="present" delimited="+"/>
```

```
  <realpred lemma="spille" pos="v"/>
</ep>
...
<rarg>
  <rargname>ARG2</rargname>
  <label vid="8"/>
  <var sort="x" vid="9" bounded="+"/>
</rarg>
...
<ep>
  <label vid="10"/>
  <var sort="x" vid="9" bounded="+"/>
  <realpred lemma="ball" pos="n"/>
</ep>
```

Figur 45 - rmrsFilterAlt.xslt resultat

Til å begynne med vil vi få ut *Lundekvam* som er egennavnet *CARG* angir. Vi følger så *vid=3* til navnerelasjonen med samme verdi. Vi registrerer nå *var vid=4* noe som spores til *ARG1*, med *labell vid=8*. Vi finner så *ep* med denne verdien, finner at det her er en *realpred lemma*, og tar vare på verdien *spille*. Vi registrerer nå at *label* verdien her er *8* noe som fører oss til *ARG2* med *label 8*. Etter å ha lagret *var vid=9* fører dette oss til vår siste *realpred* med verdi *ball*.

Vi kunne også i dette eksempelet tatt med kode for å finne ordet *til*, men dropper det da ikke er et ord av betydning for oss. Det er heller ikke noe poeng å gjøre dette mer komplisert enn det allerede er.

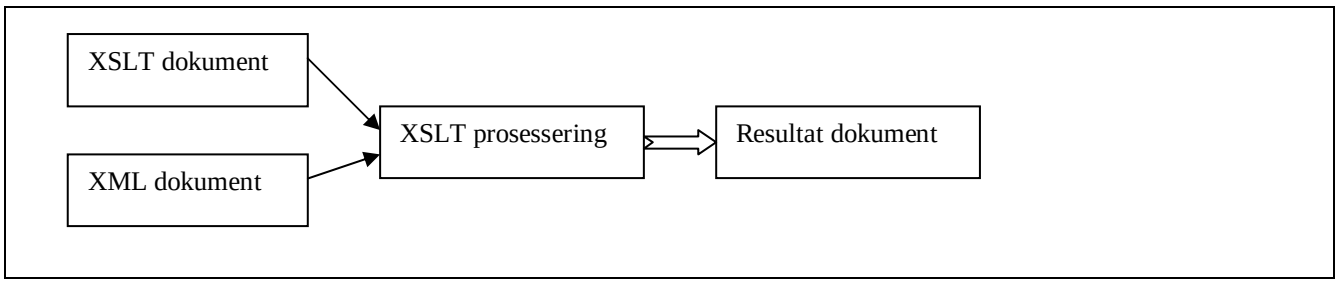
Når vi så kjører XSLT transformasjonen vi nå har forklart på hele RMRS representasjonen vil vi først få ut *Lundekvam spille ball*, som vi har vist over. Ved videre gjennomgang av representasjonen, se vedlegg B, vil vi i tillegg få ut *Magne Hoseth miste ball James McFadden*. Vi har nå all informasjon vi trenger for å lagre hendelsen i en databasetabell.

4.4 Diverse verktøy

Her følger en kort innføring i programmeringsteknikker og liknende som er benyttet under implementasjonen av systemet.

4.4.1 XSLT

XSLT (XSLT 2005) er en forkortelse for Extensible Stylesheet Language Transformations og er en spesifikasjon utviklet av Worldwide Web Consortium (www.w3.org 2005). XSLT er et språk for å omforme xml dokumenter til andre dokumenter, et xml dokuement, en webside i HTML eller til ren tekst. En slik transformasjon i XSLT utføres ved at vi oppretter regler for hvordan et kilde-tre skal omformes til et resultat-tre.



Figur 46 - XSLT figur

For å utføre en slik transformasjon trenger vi først et velformet XML-dokument. Videre må vi lage et XSLT dokument der vi angir alle ønskede egenskaper for omformingen.
For å utføre selve transformasjonen har vi brukt Apache sin Xalan implementasjon for bruk av XSLT i java programmer.

4.4.2 Xalan-Java

Xalan-Java (Xalan-Java 2005) er en XSLT prosessor for omforming av xml-dokumenter. Den implementerer XSLT versjon 1.0 og XPath versjon 1.0, og kan blant annet brukes i en Applet, en servlet eller som en modul i et program.

Fremgangsmåten for å få til en transformasjon i et javaprogram kan deles opp i tre hovedmomenter. Først må vi i opprette en ny subklasse av *TransformerFactory* ved å kjøre metoden *newInstance()*. Videre må koden *TransformerFactory newTransformer(Source stylesheet)* brukes for å prosessere omformingsinstruksjonene i XSLT dokumentet vårt og generere en *Transformer*. Som siste steg må *transform(Source xmlSource, Result transformResult)* metoden kjøres for å bruke transformasjonsinstruksene på XML-dokumentet vårt, og produserer et resultatdokument.

4.4.3 Jakarta Tomcat 5.0.28

Tomcat (Tomcat 2005) er en Open Source applikasjonsserver utviklet av The Jakarta Project under Apache Software Foundation (Apache 2005). At det er Open Source vil si at produktet er gratis å laste ned og bruke, samt at brukeren har fritt innsyn i kildekoden. Jakarta prosjektet utvikler løsninger til Java programmering som er gratis å benytte seg av. Tomcat serveren er en Javabasert applikasjon som ble utviklet for å kjøre Servlets og Java Server Pages(JSP), og fungerer som en slags JSP og servletmotor.

Ved bruk av Tomcat kan en velge å bruke den som et frittstående produkt med en egen intern webserver. Det er også mulig å bruke Tomcat sammen med andre webservere som for eksempel Microsoft Internet Information Server, Apache Server og Netscape Enterprise Server. Tomcat er også avhengig av en utgave av Java Runtime Environment (Java 2005).

4.4.4 Java-Servlet og JSP

Utviklere har lenge jobbet med å tilby dynamiske tjenester over nettet, og Applets (et lite program som sendes med en nettside til brukeren) var blant de første. Etter hvert ble CGI-script (Common Gateway Interface) hovedteknologien for å tilby dynamiske tjenester. CGI er fremdeles mye brukt, men har enkelte svakheter, som for eksempel at det er OS avhengig og lite skalerbart. Med ønske om å lage en mer komplett tjeneste for dynamiske løsninger, ble Java Servlets (Servlet 2005) utviklet.

En javaservlet er et program som ligger og kjøres på en webserver, og kan sies å være den grunnleggende byggesteinen for å lage en web-basert applikasjon. Servlets mottar og svarer på ordre som kommer inn, som regel over HTTP.

JSP(Java Server Page) er den som kontrollerer innholdet eller utseende til en servlets webside. Sun Microsystems referer også til JSP som en servlets API. En Java Server Page kaller et Javaprogram som så kjøres på webserveren.

Alle servlets går igjennom en felles livssyklus, som generelt kan deles inn i tre faser. Til å starte med initialiseres servleten, det vil i praksis si at servlet metoden init() kjøres. Den er nå klar til å ta i mot requests noe som gjøres gjennom metoden service(). Denne metoden håndterer standard HTTP-requests ved å videresende et request-objekt til metoden som er designa for å utføre denne. Serveren avslutter til sist servleten ved å kjøre destroy() metoden.

4.4.5 Apache ANT

Ant (Ant 2005) er et javabasert build tool fra Apache Software Foundation, også dette med fri tilgang til kildekoden. ”Build Tool” vil si at det setter sammen alle delene av et program. Ant har mange likhetstrekk med et annet build tool, GNUMake, men Ant er enklere å bruke. Ant er også OS

uavhengig, fordi utviklerne bruker XML for å beskrive modulene i oppbygningen av verktøyet. Ant ble utviklet av James Duncan Davidson mens han arbeidet med å utvikle programvare på tvers av flere operativsystemer. Han møtte da på så mange problemer med build tools at han utiklet sitt egne operativsystemuavhengige. Navnet Ant har det fått fordi ”Det er en liten ting som kan bygge store ting”.

4.4.6 SQL og MySQL

SQL(Structured Query Language) er et språk som brukes til å kommunisere med et databaseprogram. Det SQL gir oss er et språk for å kunne opprette og plassere informasjon i en database, hente frem data, slette data samt mye mer. SQL er et standard språk på linje med for eksempel HTML og CSS, og det finnes i dag en rekke databaser(SQL servere) som benytter SQL som språk. Den mest populære og best kjente er MySQL (MySQL 2005) og er også det vi har brukt i dette prosjektet.

MySQL (MySQL 2005) er altså et program som håndterer databaser, og benytter seg av database-språket SQL. MySQL er et såkalt open source prosjekt. MySQL brukes hovedsakelig i forbindelse med webapplikasjoner, og er veldig populært på grunn av sin stabilitet og fart. Det er vanlig å kjøre en MySQL server under Unix, men den kan også kjøres på Windows-operativsystemer og Mac OS.

For å gjøre det mulig å kontakte en database gjennom programmeringsspråket java er API’en (Application Programming Interface) JDBC, Java DataBase Connectivity, utviklet (JDBC 2005). Den gjør det mulig å legge inn SQL utsagn i Javakoden, som deretter sendes til det aktuelle databaseprogrammet, i dette tilfellet MySQL. JDBC inneholder en rekke objektorienterte klasser som programvareutvikleren kan bruke for å lage SQL spørringer. Det finnes to JDBC drivere for MySQL, og den ene av disse er MySQL Connector/J som fritt kan lastes ned for test og bruk (Connector/J 2005).

4.4.7 Google Desktop Search

Google Desktop Search is how our brains would work if we had photographic memories. (Google Desktop 2005).

Google Desktop Search er et søkeverktøy som tilbyr søk i e-mail, filer på datamaskinen og websider vi har besøkt. Programmet oppretter en indeks til all søkbar informasjon på datamaskinen, og lagrer denne. Indekseringen er en operasjon som utføres etter å ha installert programmet. For hele tiden å være oppdatert, utfører Google Desktop Search kontinuerlig slik indeksering når nye dokumenter lagres på datamaskinen. Dette gjør at vi kan søke etter informasjon på vår egen pc på samme måte som vi søker etter informasjon over nettet.
Spesifikt er det vi kan søke etter (Google Desktop 2005):

- *Email from Outlook 2000+, Outlook Express 5+, Netscape Mail 7.1+, Mozilla Mail 1.4+ and Thunderbird*
- *All the files on your computer, including text, Word, Excel, Powerpoint, PDF, MP3, image, audio, and video files. You can even search your media files by meta-tag: for instance, by artist name and song title, not just the file name.*
- *Web pages you've viewed using Internet Explorer 5+, Netscape 7.1+, Mozilla 1.4+ and Firefox.*
- *Chats from AOL 7+ and AOL Instant Messenger 5+*

Vi bruker Google Desktop Search til å indeksere og søke igjennom et fotballreferat for så å se hva slags informasjon vi klarer å trekke ut av referatet ved å kjøre enkle søk.

5.0 Evaluering

Vi legger her fram en evaluering av arbeidet som er blitt utført våren 2005. Vi starter med å fortelle hvilken informasjon vi anser som interessant i et fotballreferat. Deretter går vi inn på egenskapene ved systemet og ser litt på hva slags setninger systemet takler, eventuelt ikke takler, samt årsaken(e) til dette. Begrensninger og lignende som har vært med på å komplisere arbeidet diskuterer vi også her. I den siste delen av evalueringa sammenlikner vi vårt system med søkeverktøyet Google Desktop Search som blant annet utfører søk ved IR og indeksering.

5.1 Hvilken informasjon er det som er interessant?

Fotballreferatet vi har basert oppgava vår på er tatt fra VG sin live-dekning av kamper, og er et referat av kampen Skottland – Norge. Vi har gått igjennom dette for å finne informasjon vi anser som interessant, elementer som kan virke kompliserende, standard notasjoner og liknende.

Informasjonen vi i utgangspunktet vil trekke ut av fotballreferatet er hendelser som for eksempel dueller, utdeling av kort, når eventuelle bytter blei utført, hvor mange cornere og frispark en kamp hadde og så videre. Vi har i den sammenheng også satt opp en rekke spørsmål som representerer denne informasjonen samtidig som det fungerer som en mal på hvilke spørringer det vil være naturlig å kjøre mot en database. Spørsmålene til fotballreferatet Skottland – Norge er:

- I hvilket minutt ble det skåret mål?
- Hvem skåret mål?
- Hvor mange kort blei delt ut?
- Hvilke bytter blei utført og når?
- Hvor mange frispark var det i kampen?
- Hvor mange cornere var det i kampen?
- Hvor mange sjanser var det i kampen?

Informasjonen vi ønsker å ta vare på er av typen man ikke kan lese rett ut av fotballreferatet. For eksempel kan vi finne vinneren av kampen fra den første linja i referatet. Er vi derimot interessert i å finne ut av hvor mange cornere som blei tatt i løpet av kampen, krever dette en nøye gjennomlesing av referatet. Hvis vi derimot har lagret all relevant informasjon fra kampen i en database, er vi kun en spørring fra å finne antall cornere. På samme måte kan dette utnyttes ved å spørre etter antall bytter, hvor mange gule kort som blei delt ut og så videre.

Vi kan videre være i stand til å sammenlikne flere referater. For å kunne hente frem aggregert informasjon, har vi definert nye spørsmål som bygger på informasjonen fra et enkelt referat.

- I hvilken kamp blei det skåret først?
- I hvilken kamp blei det skåret flest mål?
- I hvilken kamp blei det først delt ut kort?
- I hvilken kamp blei det delt ut flest kort?
- Hvilket lag blei tildelt flest kort?
- Hvilket lag fikk \ forårsaket flest frispark?
- I hvilken kamp var det flest cornere?

Det er også mulig å ta dette enda et steg videre for å finne kvalitativ informasjon. Dette kan for eksempel være om en spiller har vært mye involvert og gjort en god kamp, om spilleren har hatt en positiv utvikling de siste kampene, tendensen i et lag og lignende.

Et kompliserende element er at fotballreferatene er skrevet av en som følger kampen direkte. Dette gjør sitt til at det ikke bare er korte og enkle setninger vi skal se på, men også mye dårlig norsk og ufullstendige setninger. Dette var også noe som viste seg å være et større problem enn vi i utgangskpunktet antok.

Videre finnes det også enkelte standard notasjoner vi kan gjøre oss nytte av. Et mål vil for eksempel angis med ”0-1” etterfulgt av etternavnet på målskåreren. Også bytter er stort sett alltid på samme form, *SpillerA kommer inn for SpillerB*.

5.2 Egenskaper ved systemet

Vi har under utviklingen av systemet møtt på en del problemer. For det første skal vi koble oss opp mot et søkeverktøy lokalisert på Dragvoll, Heart of Gold. Det har til tider vært enkelte problemer med HoG’en, den har ikke vært tilgjengelig for RPC kall. Dette gjorde utviklingen av metodekallene mot søkeverktøyet vanskelig, da vi til tider ikke visste om feilen lå i programkoden vår, eller om søkeverktøyet rett og slett ikke var tilgjengelig.

Etter å ha fått kontakt med HoG, er vi også avhengig av at den skal kjenne igjen og analysere setningene vi sender med XML-RPC kallene. Grammatikken som brukes på HoG har i tillegg blitt forandret siden starten av Masteroppgaven, noe som har komplisert Lars Hellan sitt arbeid med å legge til nye setninger i biblioteket.

5.2.1 RMRS representasjonene

RMRS representasjonene har helt klart også vært en meget kompliserende faktor i arbeidet vårt. Vi har brukt utallige timer og dager på å lese og forstå disse kompliserte strukturene. Når dette endelig begynte å bli forståelig, lå utfordringen i å filtrere ut relevant informasjon.

5.2.2 Setningene i et fotballreferat

Setningene vi i utgangspunktet fikk lagt til i biblioteket er tatt fra de åtte første minuttene av kampen mellom Skottland og Norge.

Claus Lundekvam og Paul Dickov i en tett duell som Lundekvam til slutt vinner. Han spiller til Magne Hoset som mister ballen til Darren Fletcher og lager et farlig frispark på 25 meters hold.
James McFadden prøver et skudd som Thomas Myhre redder uten store problemer.
Flott raid av Carew som kommer ned til kortlinjen, slår tunnel på Gary Caldwell og legger ut i feltet, men skottene får klaret til corner.
Hjørnesparket blir det ikke noe farlig ut av.
Skottland slår ballen inn i feltet, men Lundekvam får til slutt headet tilbake til Myhre.
Hoset skyter fra 25 meter, men treffer Russell Anderson i ansiktet. Nordmannen får en ny mulighet, men treffer dårlig og muligheten koker bort.

Figur 47 - Eksempel på fotballreferat

Disse er videre blitt modifisert for å kunne takles av biblioteket til HoG. Dette har gått ut på å dele opp lange setninger, og også omformulere enkelte. De nye setningene blei seende ut som under.

Claus Lundekvam og Paul Dickov er i en tett duell. Lundekvam vinner duellen til slutt. Lundekvam spiller til Magne Hoset som mister ballen til Darren Fletcher.
Han lager et farlig frispark på tjuefem meters hold.
James McFadden prøver et skudd. Thomas Myhre redder uten store problemer.
Det er et flott raid av Carew som kommer ned til kortlinjen. Carew slår tunnel på Gary Caldwell og legger ut i feltet.
Skottene klarerer til corner.

Hjørnesparket er ufarlig.
Skottland slår ballen inn i feltet, men Lundekvam header tilbake til Myhre til slutt.
Hoset skyter fra tjuefem meter, men treffer Russell Anderson i ansiktet. Nordmannen får en ny mulighet, men treffer dårlig og muligheten koker bort.

Figur 48 - Modifiserte fotballsetninger

Dette er setningene vi har brukt for å få til å analysere, filtrere og lagre data i en database. Vi kan ta for oss setningen ”Lundekvam vinner duellen til slutt”. Det første vi må gjøre er å sende setningen til analysering.

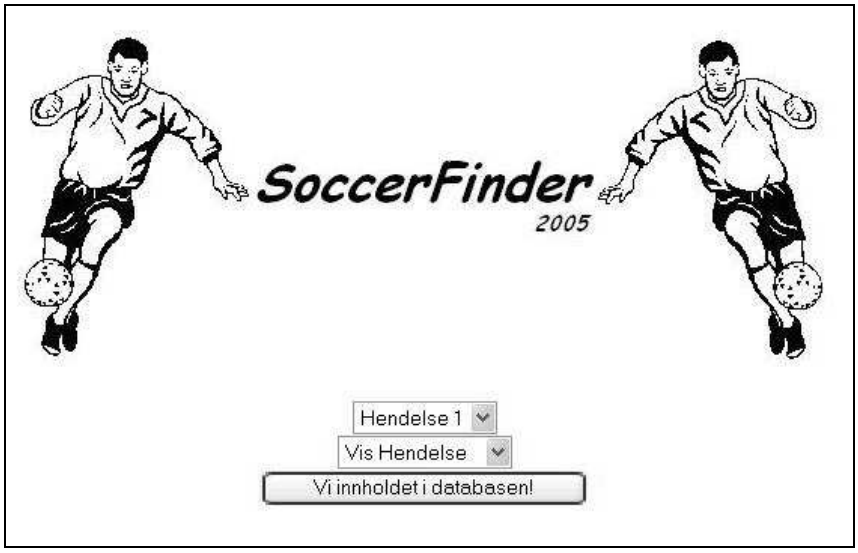
Figur 49 - Grensesnitt søk

Setningen sendes til HoG som sender tilbake en RMRS representasjon på XML format, eventuelt en feilmelding. Vi utfører så en filtrering av XML dokumentet for å kunne lagre interessante data i en databasetabell. Som vist tidligere i rapporten vil filtrering av RMRS for setningen ”Lundekvam vinner duellen til slutt” gi oss ordene *Lundekvam*, *vinne* og *duell*. Vi kan da lagre denne informasjonen i en enkel databasetabell.

HendelseNR	Aktør	Hendelse	Utfall
1	Lundekvam	duell	vinne

Figur 50 - Database-tabell eksempel

Vi kan så videre utføre spørringer mot databasen for å hente fram allerede lagret informasjon, i dette tilfelle en enkel spørring for å hente fram en hendelse.



Figur 51 - Grensesnitt database

Vi vil da få ut *Lundekvam vinne duell* i brukergrensesnittet vårt. På akkurat samme måte vil systemet fungere når vi analyserer en setning, eller ønsker å hente fram allerede lagrede data.

5.2.3 Hvilke setninger klarer vi å analysere?

Setninger som fungerer bra å søke i gjennom og filtrere er setninger som starter med et egennavn. Det er nemlig det vi har tatt utgangspunkt i i filtreringen av representasjonene. Vi knytter en eller flere spillere opp mot hendelser og utfall av hendelser. Setninger som for eksempel ”Hjørneparket er ufarlig” vil vi derfor ha problemer med å håndtere.

5.2.4 Kompliserende notasjoner i et fotballreferat

Det er enkelte notasjoner som har gjort analysering av setninger problematisk. Når et fotballreferat forteller oss at det er skåret et mål, gjøres det på formen ”(0-1) Iversen”. Dette er problematisk da søkeverktøyet ikke klarer å håndtere parenteser, kolon og lignende. Måten vi kan løse dette på er at vi kjenner igjen setninger som starter med ”(X - Y) egennavn”, og henter dette ut fra søkestrengen før vi sender setningen til analysering. Vi kan deretter knytte setningene vi har fått analysert til målscorer og stilling.

Biblioteket vil heller ikke være i stand til å skille synonymer. Så hvis vi hadde sendt de to formuleringene ”Iversen banker ballen i mål.” og ”Iversen skyter ballen i mål.” til HoG, ville den ikke vært i stand til å se at dette er en og samme hendelse. Dette vil for oss ikke være en aktuell problemstilling da hendelsene i et fotballreferat kun adresseres en gang.

5.2.5 Aggregerte data

Vi har i systemet vårt ikke fått gjennomført spørringer med aggregerte data. Biblioteket som søkeverktøyet bruker har hittil bare lagt inn enkelte setninger fra vår utvalgte kamp, og vi har ikke mulighet for å gjennomføre de planlagte spørringene. Dette ville derimot ikke krevd noen stor utvidelse av programmet. For å finne aggregert data fra flere referater kunne vi rett og slett ha lagret en variabel, kampNR, med alle databasetabeller. Hvis vi så ønsket å finne hvilken kamp som hadde flest hjørnespark, kunne vi bare telle antallet i for eksempel kampNR 1 og 2, sammenlikne verdiene og returnere kampen med høyest antall cornere.

5.3 Egenskaper ved fotballreferater

Publiserte fotballreferater inneholder mange lange komplekse setninger, noe som gjør vårt arbeid problematisk. Vi har for eksempel en del indre egenskaper ved setningene vi behandler. Referater

inneholder skrivefeil, noe HoG ikke kan håndtere. Vi har i tillegg spesialuttrykk som ”gult kort” som for oss betyr noe helt spesielt, men som for søkeverktøyet ganske enkelt er et adjektiv og subjekt. Ufullstendige setninger er også et element som vil skape problemer i spørresystemet vårt.

HoG er et generelt lingvistisk verktøy, og det baserer seg på korrekte og fullstendige setninger. Formatet og kvaliteten et fotballreferat leveres på er derfor et problem for oss.

Det er også vanskelig å vite hva som er en god måte å representere informasjonen vi klarer å hente ut fra et referat. Selv med et fullt fungerende system og søkeverktøy er dette en stor utfordring.

5.4 Sammenlikning med søkeverktøy

Vi har til slutt valgt å gjøre en kort sammenlikning av systemet vårt og Google Desktop Search (Google 2005). Vi kan nemlig bruke Google Dekstop Search til å indeksere fotballreferatet Skottland – Norge, som vi har lagret som et Word-dokument. Videre utfører vi enkle søk i Google for å se hvilken informasjon vi er i stand til å finne.

Under har vi et eksempel på dette der vi søker på ordene ”Lundekvam duell”. Ved å søke i det indekserte referatet vil vi få tilbake:

Figur 52 - Eksempel fra Google Desktop

Vi kan ut ifra resultatet se at Lundekvam var involvert i en duell. For å finne ut hvordan det gikk, må vi inn å lese oss fram til det. Vi finner her i tillegg ut når kampen blei spilt og hva sluttresultatet var. Dette er bare tilfeldig, da setningen vi søkte oss fram til rett og slett er helt i starten av referatet.

Ved å bruke vårt system kan vi også finne igjen at Lundekvam vant en duell tidlig i kampen. Den store forskjellen mellom de to fremgangsmåtene vil være at vi i Google vil søke oss fram til tilfeldige forekomster av ord i referatet, mens vi i systemet vårt vil søke etter informasjon vi veit er lagret. Det vil med andre ord være lettere å finne akkurat den lille informasjonen vi er på leiting etter ved å bruke SoccerFinder. I Google vil vi også ofte finne de korrekte opplysninger vi leiter etter, men kan også ende opp med for mye eller for lite informasjon.

Den største forskjellen mellom Googe og vært spørrsesystem er at vi vil være i stand til å hente ut aggregert informasjon fra et eller flere referater. Vi vil for eksempel være i stand til å finne antall gule kort i en setning ved en spørring. Hvis vi videre hadde lagret data fra flere kamper i databasen vår kunne vi ved hjelp av spørringer funnet aggregert informasjon fra flere kamper. Vi kunne da funnet fram til hvilken kamp som hadde flest eller færrest gule kort. Dette er egenskaper Google Desktop Search ikke vil være i stand til å ”matche”.

5.5 Samlet vurdering

Når vi henter ut setninger fra et fotballreferat publisert på nett, inneholder disse alt fra skrivefeil til ufullstendige setninger. Dette kompliserer arbeidet vårt da vi skal sende setninger til et lingvistisk verktøy, HoG. Det har i løpet av prosjektet også vært en del problemer med søkeverktøyet, og det har vært lengre perioder vi ikke har fått kontakt. Dette har vært med på å forsinke utviklingen av systemet vårt. Søkeverktøyet er heller ikke i stand til å håndtere kolon, parenteser og lignende, og vi

er nødt til å fjerne dette ut av setninger. Vi må også merke oss at HoG ikke er i stand til å kjenne igjen for eksempel synonymer.

Når vi sender en setning til HoG, vil vi som resultat motta semantiske representasjoner i form av RMRS. Dette er komplekse representasjoner som det har vært en tidkrevende prosess å sette seg inn i og forså.

Vi klarer med systemet vårt å analysere enkle setninger med en eller flere aktører. Vi filtrerer RMRS representasjoner og legger den interessante informasjonen inn i databasetabeller. Disse kan vi seinere kjøre spørringer mot for å finne igjen lagrede data.

Det som skiller vårt spørresystem fra dagens informasjonssystemer er at vi vil være i stand til å finne igjen aggregert informasjon. Vi kan for eksempel finne antall gule kort i en kamp, eller ved å lagre data fra flere referater finne hvilken kamp det blei delt ut flest kort i. Dette vil ikke et søkesystem som for eksempel Google være i stand til.

6.0 Konklusjon

Vi har i dette prosjektet utviklet et system, SoccerFinder, som analyserer setninger fra et fotballreferat ved å sende disse til et lingvistisk søkeverktøy. Ved å bruke XML-RPC får vi tilbake en RMRS representasjon for setningen, som vi så må hente ut og lagre interessante data fra. Vi kan deretter kjøre spørringer mot en database for å hente frem lagrede hendelser fra fotballreferatet.

I arbeidet med oppgaven har vi brukt et fotballreferat fra VG sine nettsider av kampen Skottland – Norge. Slike referater har vist seg å inneholde mye rart språk, og også en del ufullstendige setninger. De innholder også enkelte tegn søkeverktøyet HoG ikke vil kunne håndtere, noe som har gjort arbeidet noe mer komplisert for oss.

Vi har i utviklingen av systemet satt oss inn i Robust Minimal Recursion Semantics, en måte å uttrykke semantiske strukturer på. RMRS er en modifisert utgave av MRS, Minimal Recursion Semantics, og skal være bedre egnet for bruk i komputasjonelle systemer. RMRS strukturene vi får tilbake fra søkeverktøyet er avanserte, og tidkrevende å sette seg inn i. En slik RMRS er bygget opp av argumenter med hver sin tilhørende *ep*. Alle variabler er i tillegg angitt, og samtlige har identifikatorer.

Vi ser helt klart nytten i en slik RMRS representasjon, men arbeidet vårt kunne blitt mye greiere dersom det blei utviklet en forenkling. Det virker som det er en del lingvistisk interessant informasjon i representasjonene som ikke er nødvendig for de analysene vi kjører i dette prosjektet. Kanskje kunne all indekseringen vært gjort på et mer håndterbart vis.

Det fine med RMRS er at vi alltid finner igjen grunnformen av ordene i setningene. Det er også enkelt å hente ut egennavn fra en slik representasjon, noe som er essensielt når vi arbeider med et fotballreferat. Dette gjør også sitt til at vi ikke har problemer med å skille for eksempel egennavn og subjekt i en setning.

Utviklingen av et spørresystem som analyserer og tar vare på informasjon fra et fotballreferat har vist seg å være en formidabel utfordring. Til å begynne med var motivasjonen å sitte igjen med et system der vi hentet inn et helt referat, analyserte og lagret all interessant informasjon fra dette, for så å kunne kjøre spørringer på de lagrede data. Dette har vist seg å være mer eller mindre umulig da slike fotballreferater publiseres med mange varierende notasjoner. Det har vært nødvendig å skrive om setninger, og fotballterminologien må også legges til i biblioteket til søkeverktøyet. RMRS representasjonene vi har fått tilbake har gitt oss den informasjonen vi er på utkikk etter, men har også vært på et såpass avansert format at det å kjenne igjen mønstre og filtrere ut informasjon har vært problematisk.

Systemet vårt er i stand til å analysere og filtrere korte setninger med èn aktør, men også situasjoner der flere aktører er involvert. Hendelsene vi takler er tatt fra de første 8 minuttene av kampen Skottland – Norge, da dette er setningene som er lagt til i biblioteket til HoG. Vi kan videre lagre denne informasjonen i enkle databasetabeller for seinere å kjøre spørringer mot disse.

6.1 Fremtidig arbeid

I det videre arbeidet med systemet vil det være interessant å utvide biblioteket til HoG med flere setninger fra fotballreferatet vi har arbeidet med. Vi kan da analysere og legge til nye hendelser i databasetabeller, og kjøre spørringer mot disse.

Grammatikken søkeverktøyet bruker er siden begynnelsen av vårt prosjekt veldig forandret, og dette kan bety at RMRS representasjonene vi får tilbake vil være noe annerledes. Dette vil med andre ord si at XSLT transformasjonene vi utfører må endres. Hvor store forandringer dette er snakk om er i skrivende øyeblikk ikke mulig å si, da HoG den siste tiden har vært nede grunnet vedlikehold og grammatikkjustering.

Et naturlig tillegg vil videre være å få setninger inn i biblioteket fra et annet fotballreferat. Dette vil ikke kreve nevneverdige justeringer av programkoden, men vil gi oss økt funksjonalitet ved at vi kan sammenlikne fakta fra to forskjellige kamper. Vi kan for eksempel da finne ut hvilke av to kamper som hadde flest gule og røde kort eller antall hjørnespark.

7.0 Referanser

[Ambler 2005]Scott W. Ambler (2005). Modeling Style Guidelines. http://www.agilemodeling.com/style/

[Amble 2002] Tore Amble (2002). The Understanding Computer –Natural Language Understanding in Practice, Preliminary Version.

[Amble 1998] Tore Amble (1998). BusTuc – A natural language bus route oracle.

[Ant 2005]Ant Build Tool (2005). http://ant.apache.org/

[Apache 2005] The Apache Software Foundation (2005). http://www.apache.org

[Apache XML-RPC 2005] Apache XML-RPC (2005). http://ws.apache.org/xmlrpc

[Bruke 2004]Eric M. Bruke (2000). Developing, Applying and Optimizing XSLT with Java Servlets.

[BussOrakel 2005] BussOrakelet(2005). www.team-trafikk.no/asttweb/bussOrakel2.asp.

[Connector/J 2005] Connector/J 3.1 (2005). http://dev.mysql.com/downloads/connector/j/

[Copestake 2003]Ann Copestake(2003). Report on the design of RMRS (preliminary version)

[Callmeier et al. 2004]Ullrich Calllmeier, Andreas Eisele, Ulrich Schäfer, Melanie Siegel (2004). The DeepThought Core Architecture Framework.

[Copestake et al. 2003]Ann Copestake, Dan Flickinger, Carl Pollard Ivan A. Sag(2003). Minimal Recursion Semantics: An Introduction.

[Databases and Java.] http://www.cs.usfca.edu/~parrt/course/601/lectures/db.html

[Frank et al. 2004]Anette Frank, Kathrin Spreyer, Witold Drozdzynski, Hans-Ulrich Krieger and Ulrich Schäfer (2004).Constraint-Based RMRS Construction from Shallow Grammars.

[Google 2005]Google (2005). www.google.com

[Google Desktop 2005] Google Desktop Search (2005). http://desktop.google.com

[Java 2005] Java Technology. http://java.sun.com/

[JDBC 2005] Java DataBase Connectivity (2005). http://java.sun.com/products/jdbc/

[Montague 1970]Richard Montague (1970). English as a Formal Language.

[Montague 1970]Richard Montague (1970). Universal Grammar.

[Montague 1973]Richard Montague (1973). The Proper Treatment of Quantification in Ordinary English (PTQ).

[MySQL 2005]MySQL (2005). www.mysql.com

[Omondo 2005]Omondo EclipseUML (2005). http://www.omondo.com/

[Parsons 1990]Parsons, T. (1990). Events in the Semantics of English, Cambridge, MA: MIT Press

[Pereira and Schieber, 1987] Pereira, F.C.N and Schieber, S.M. Prolog and Natural-Language Analysis.

[Servlet 2005]Java Servlet technology (2005). http://java.sun.com/products/servlet/

[Tomcat 2005]Tomcat Webserver(2005). http://jakarta.apache.org/tomcat/

[UML 2005]Unified Modeling Language (2005.)http://www.uml.org/

[VG 2005]VG-live (2005). live.vg.net.

[Warren and Pereira, 1980] Pereira, F.C.N and Warren, D.H.D. Definite Clause Grammars for Language Analysis – A Survey of the formalism and a Comparison with Augmented Transition Networks.

[Wikipedia 2005]Wikipedia – The free Encyclopedia (2005). http://en.wikipedia.org.

[www.w3.org 2005]Worldwide Web Consortium - http://www.w3.org/

[Xalan-Java 2005]Xalan-Java version 2.6.0 (2005). http://xml.apache.org/xalan-j/

[XML 2005]XML – eXtensible Markup Language (2005). www.w3.org/XML/

[Xml-Rpc 2005]XML-RPC (2005). www.xmlrpc.com

[XSLT 2005]Extensible Stylesheet Language Transformations (XSLT). http://www.w3.org/TR/xslt

8.0 Vedlegg A - Figurliste

9.0 Vedlegg B – Skottland – Norge

Skottland - Norge 0 - 1

9/10-2004 - kl.16:00

Kampen er igang - Skottland tar avspark.

1. Claus Lundekvam og Paul Dickov i en tett duell som Lundekvam til slutt vinner. Han spiller til Magne Hoset som mister ballen til Darren Fletcher og lager et farlig frispark på 25 meters hold.

3. James McFadden prøver et skudd som Thomas Myhre redder uten store problemer.

3. Flott raid av Carew som kommer ned til kortlinjen, slår tunnel på Gary Caldwell og legger ut i feltet, men skottene får klaret til corner.

4. Hjørnesparket blir det ikke noe farlig ut av.

6. Skottland slår ballen inn i feltet, men Lundekvam får til slutt headet tilbake til Myhre.

8. Hoset skyter fra 25 meter, men treffer Russell Anderson i ansiktet. Nordmannen får en ny mulighet, men treffer dårlig og muligheten koker bort.

11. En lang ball slås i retning av Barry Ferguson - Myhre er våkent ute og rydder opp.

12. Hoset slår hardt inn fra venstre, men en forsvarer får blokkert inne i feltet foran bena til Jan Derek Sørensen.

16. Steffen Iversen stusser et langt kast fra John Arne Riise, men ingen er på plass ved første stolpe og Skottland-keeper Craig Gordon holder.

17. André Bergdølmo legger inn fra høyre, men Carew og Iversen går på samme ball og er heldige som får en corner.

18. Hjørnesparket havner rett i klypene til Gordon.

18. Bergdølmo feller Dickov på hjørnet av 16-meteren. Darren Fletcher slår ballen inn i feltet - Myhre plukker ned.

19. Ferguson skyter over fra drøyt 20 meters hold.

21. Hoset kommer seg elegant fri på venstrekanten og slår fint inn - Carew er centimetere fra å nå frem med pannebrasken.

21. En farlig norsk corner som det skotske forsvaret ikke klarer å få unna. Jan Gunnar Solli prøver et skudd, men treffer dårlig og ballen blokkeres.

23. Hoset slår en corner på hodet til Carew. Headingen går like over fra fire meters hold.

Jevnspilt så langt, men hjemmelaget har vært skuffende i det offensive spillet.

29. Dickov er svært forbannet etter en duell med Erik Hagen og skjeller ut dommeren når han får frispark mot seg. Men den hissige skotten slipper kort.

30. Solli gir til corner etter en duell med McFadden.

31. Kanonsjanser til vertene. Corneren løftes inn og det norske forsvaret redder to ganger på streken. Myhre blokkerer mesterlig det første forsøket fra Dickov. Richard Huges avslutter til slutt, men Iversen står på streken og blokkerer.

36. Hoset slår et frispark inn i feltet og Iversen får en stor sjanse, men Erik Hagen har forårsaket et frispark som dommeren blåser for.

37. Dickov sparker til Hagen i en duell nede ved kortlinjen. Han er meget heldig som slipper unna med gult kort.

39. Hagen kommer høyest på bakre stolpe og header på mål - en skotte står i veien og blokkerer.

40. McFadden slår en corner hardt inn på første stolpe, Iversen er på plass og klarerer. Småfaig av hjemmelaget.

Skottland presser nå.

42. Dickov prøver et skudd fra distanse som Myhre redder enkelt.

42. Dickov får seg en smell i foten i en duell med Hagen. Spissen halter, men får ikke behandling.

43. Carew kommer seg fri på venstrekanten og slår inn i feltet - Iversen når ikke fram.

44. Steffen Iversen stusser et langt innkast fra Riise igjen, men denne gangen får han frispark mot seg. Iversen har uansett vært meget god i lufta i 1. omgang.

Pause (0 - 0)

2. omgang er igang - Norge tar avspark.

47. Dickov spilles igjennom på høyrekanten, men vinkes av for offside. Reprisen viser at det er feil - Hagen opphever offsiden.

48. Solli slår en håpløs pasning som Fletcher fanger opp, men Riise holder hodet kaldt og får ryddet opp i situasjonen.

49. Hoset dribler bort tre skotter, men pasningen i retning Riise er altfor upresis.

50. Sørensen får en corner etter at et innleggsforsøk blokkeres.

51. Hoset slår en farlig corner som skrur innover, men ingen nordmenn får avsluttet.

53. Fin spill og Carew får en pasning på høyrekanten, men han er presset og må nøye seg med en corner.

53.  STRAFFE TIL NORGE. Hoset slår inn og Lundekvam header mot mål. McFadden redder med hånden på streken.

54.  RØDT KORT: McFadden. Everton-spilleren får helt korrekt det røde kortet.

54.  **0 - 1** IVERSEN (str.). Iskald straffe fra Vålerenga-spissen. Han setter ballen lavt til høyre for keeper som kaster seg feil veg.

56.  Ferguson løfter et frispark inn på bakre stolpe og Andy Webster bør muligens ha straffe i en duell med Hagen.

58.  Morten Gamst Pedersen kommer inn for Hoset.

60.  Dickov er på farten igjen og kommer seint inn i en duell med Myhre. Etterslengen kvalifiserer til gult kort, og det har han fra før. Men dommeren lar det passere.

63.  Stephen Pearson kommer inn for Hughes.

65. Fletcher kommer til innlegg fra høyre - Lundekvam gjør en viktig inngrepen og header unna.

66.  Dickov vender opp på 16-meteren med Hagen i ryggen. Skuddet får han brukbart treff på, men Myhre redder.

68.  Sørensen feller Pearson altfor sent og får et fortjent gult kort.

70.  Carew spilles igjennom på venstrekanten. Men innlegget er slurvete fra spissen. Burde fått mye mer ut av den situasjonen.

74.  Caldwell klipper ned Gamst Pedersen og får gult kort.

74.  Martin Andresen kommer inn for Sørensen.

75.  Kenny Miller erstatter Dickov i det skotske angrepet.

76.  Andresen header ned til Gamst Pedersen inne i feltet. Han klarer å rive seg løs fra oppasseren og er helt alene på fire meter. Men Blackburn-spilleren avslutter høyt over.

77. Gamst Pedersen skyter på et frispark fra 30 meter - håpløst over og utenfor.

79. Russell Anderson stopper Carew som prøver å løpe igjennom med ball. Godt stopperspill.

80.  Carew får behandling for en skade.

80.  Myhre feilberegner et høyt innlegg og ballen lander oppå tverrliggeren. Norges keeper slipper med skrekken.

81.  Steven Thompson kommer inn for Gary Holt.

81. Hagen stopper Miller inne i feltet med en kontant takling. Skottene får corner, men den blir det ikke noe farlig ut av.

82. Gamst Pedersen slår et ufarlig innlegg istedenfor å spille frem Riise på venstrekanten. Svakt igjen av innbytteren.

83.  Miller feller Gamst Pedersen og Norge får frispark fra skrå vinkel på venstrekanten.

84. Andresen slår frisparket, men forsvaret header enkelt unna.

- Norge klarer ikke å holde ballen i laget - det blir altfor mange upresise offensive pasninger.
- Norge spiller halvhjertet offensivt og klarer ikke å utnytte romene i det skotske forsvaret.

89.  Carew header på mål etter et langt innkast fra Riise igjen - keeper redder.

90. Solli mister ballen i farlig posisjon til Pearson, men innlegget fra sistnevnte er for upresist og sjansen koker bort.

Det legges til tre minutter.

90.  Solli forårsaker et noe billig frispark på venstrekanten.

90. Ferguson slår inn - Frode Johnsen klarerer.

90. Norge kontrer og det ender med at Riise avslutter fra skrått hold - en forsvarer blokkerer.

90.  Miller spiller i retning Pearson inne i feltet, og sistnevnte faller som en potetsekk og vil ha straffe - dommeren vinker videre.

Kamp slutt (0 - 1)

10.0 Vedlegg C – RMRS representasjon

Under følger RMRS representasjonen for setningen ”Lundekvam spiller ballen til Magne Hoseth som mister ballen til James McFadden.” Denne er med da den brukes som eksempel i kapittel 4.3.3.

```
<SoccerFinder_result>
  <ep>
    <gpred>named_rel</gpred>
    <label vid="3"/>
    <var sort="x" vid="4"/>
  </ep>
  <ep>
    <gpred>def-q-rel</gpred>
    <label vid="5"/>
    <var sort="x" vid="4"/>
  </ep>
  <ep>
    <label vid="8"/>
    <var sort="e" vid="2" tense="present" delimited="+"/>
    <realpred lemma="spille" pos="v"/>
  </ep>
  <ep>
    <label vid="10"/>
    <var sort="x" vid="9" bounded="+"/>
    <realpred lemma="ball" pos="n"/>
  </ep>
  <ep>
    <gpred>def-q-rel</gpred>
    <label vid="11"/>
    <var sort="x" vid="9" bounded="+"/>
  </ep>
  <ep>
    <gpred>card_rel</gpred>
    <label vid="14"/>
    <var sort="x" vid="9" bounded="+"/>
  </ep>
  <ep>
    <label vid="10001"/>
    <var sort="u" vid="15" semsort="end-of-path"/>
    <realpred lemma="til" pos="p"/>
  </ep>
  <ep>
    <gpred>named_rel</gpred>
    <label vid="18"/>
    <var sort="x" vid="16" semsort="endpnt-of-path"/>
  </ep>
  <ep>
    <gpred>def-q-rel</gpred>
    <label vid="19"/>
    <var sort="x" vid="16" semsort="endpnt-of-path"/>
  </ep>
  <ep>
    <gpred>_same-ind-in-new-event-rel</gpred>
    <label vid="10002"/>
    <var sort="u" vid="22"/>
  </ep>
  <ep>
    <label vid="24"/>
    <var sort="e" vid="23" tense="present" delimited="+"/>
    <realpred lemma="miste" pos="v"/>
  </ep>
  <ep>
    <label vid="26"/>
    <var sort="x" vid="25" bounded="+"/>
    <realpred lemma="ball" pos="n"/>
  </ep>
  <ep>
    <gpred>def-q-rel</gpred>
    <label vid="27"/>
    <var sort="x" vid="25" bounded="+"/>
  </ep>
```

```
<ep>
  <gpred>card_rel</gpred>
  <label vid="30"/>
  <var sort="x" vid="25" bounded="+"/>
</ep>
<ep>
  <label vid="10003"/>
  <var sort="u" vid="32" semsort="end-of-path"/>
  <realpred lemma="til" pos="p"/>
</ep>
<ep>
  <gpred>named_rel</gpred>
  <label vid="34"/>
  <var sort="x" vid="31" semsort="endpnt-of-path"/>
</ep>
<ep>
  <gpred>def-q-rel</gpred>
  <label vid="35"/>
  <var sort="x" vid="31" semsort="endpnt-of-path"/>
</ep>
<ep>
  <gpred>prpstn_rel</gpred>
  <label vid="1"/>
  <var sort="h" vid="38"/>
</ep>

<rarg>
  <rargname>CARG</rargname>
  <label vid="3"/>
  <constant>Lundekvam</constant>
</rarg>
<rarg>
  <rargname>ARG1</rargname>
  <label vid="8"/>
  <var sort="x" vid="4"/>
</rarg>
<rarg>
  <rargname>ARG2</rargname>
  <label vid="8"/>
  <var sort="x" vid="9" bounded="+"/>
</rarg>
<rarg>
  <rargname>CARG</rargname>
  <label vid="14"/>
  <constant>non-plur-rel</constant>
</rarg>
<rarg>
  <rargname>ARG1</rargname>
  <label vid="10001"/>
  <var sort="x" vid="9" bounded="+"/>
</rarg>
<rarg>
  <rargname>ARG2</rargname>
  <label vid="10001"/>
  <var sort="x" vid="16" semsort="endpnt-of-path"/>
</rarg>
<rarg>
  <rargname>CARG</rargname>
  <label vid="18"/>
  <constant>Magne Hoset</constant>
</rarg>
<rarg>
  <rargname>ARG1</rargname>
  <label vid="10002"/>
  <var sort="x" vid="16" semsort="endpnt-of-path"/>
</rarg>
<rarg>
  <rargname>ARG2</rargname>
  <label vid="10002"/>
  <var sort="e" vid="23" tense="present" delimited="+"/>
</rarg>
<rarg>
  <rargname>ARG1</rargname>
  <label vid="24"/>
  <var sort="x" vid="16" semsort="endpnt-of-path"/>
</rarg>
<rarg>
```

```
    <rargname>ARG2</rargname>
    <label vid="24"/>
    <var sort="x" vid="25" bounded="+"/>
  </rarg>
  <rarg>
    <rargname>CARG</rargname>
    <label vid="30"/>
    <constant>non-plur-rel</constant>
  </rarg>
  <rarg>
    <rargname>ARG1</rargname>
    <label vid="10003"/>
    <var sort="x" vid="29" bounded="+"/>
  </rarg>
  <rarg>
    <rargname>ARG2</rargname>
    <label vid="10003"/>
    <var sort="x" vid="31" semsort="endpnt-of-path"/>
  </rarg>
  <rarg>
    <rargname>CARG</rargname>
    <label vid="34"/>
    <constant>Darren Fletcher</constant>
  </rarg>
</SoccerFinder_result>
```

11.0 Vedlegg D – Java-kode

SendXmlRpc.java

```
package servlet;

/**
 * @author stianjun(sstrand81@hotmail.com)
*/
import filter.GetData;
import filter.RmrsFilter;
import java.io.IOException;
import java.net.MalformedURLException;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import variabler.Atts;
import web.ResultPage;
import xmlrpc.XmlRpcProxy;

public class SendXmlRpc extends HttpServlet {
    public void init() {
  System.out.println("SendxmlRpc - initialiserer klient");
  //     returnerer en streng med den initialiserte verdien, evt. null.
  String home = getInitParameter("home");
  System.out.println("Home : " + home);

    } //init()

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          doPost(request,response);
          System.out.println("doGet i SendXmlRpcServlet");
    } //doGet()

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          System.out.println("doPOst.SendXmlRpc");
          String test =request.getParameter("text");
          //sjekker at det er skrevet inn en setning.
          if(test.equals(null) || "".equals(test)){

                 response.getWriter().write(ResultPage.getPage("Du må skrive inn en fotball-setning!"));
          }
          else{
          //teksten som skal analyseres
          String query = request.getParameter("text");
          XmlRpcProxy proxy = getXmlRpcProxy(request.getSession());  //xmlrpc session
          System.out.println("requesst.getsession");
          System.out.println( "[SendXmlRpc.doPost] "+proxy );
          String result, rmrs;

          try {
                 result = proxy.analyse(query);//lagrer resultatet fra HoG

                 //sjekker om setningen blei analysert eller ikke. Skriver ut deretter.
                 if(!"".equals(result) || result.equals(null)){
                        response.getWriter().write(ResultPage.getPage("Analysen blei utført uten feil!"));
                        //Filtrering av resultatet, lagres i rmrs(String)
                        rmrs = RmrsFilter.XmlFilter(result, Atts.XslSource);
                         //sender resultataet til lagring-bearbeidelse
                        GetData.getRmrsData(rmrs, query);

                 }else//fungerte ikke
                 {
                        response.getWriter().write(ResultPage.getPage("Analysen av setningen feilet. " +
                                      "Setningen fantes ikke i leksikonet, eller det oppsto en feil."));
                 }

          } catch ( Exception e ) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
          }
          System.out.println("dopost i sendxmlrpc nederst");

          }
    } //doGet()
```

```
    private XmlRpcProxy getXmlRpcProxy(HttpSession session) throws NullPointerException {

          XmlRpcProxy proxy = (XmlRpcProxy)session.getAttribute("xmlrpcproxy");
          System.out.println("getXmlRpcProxy() linje 2." + " [proxy]: " + proxy + " session:  " + session);

          if(proxy == null) {
                 try {
                        String url = Atts.URL_HOG;
                        System.out.println("url som parameter: " + url + " proxy_verdi: " + proxy);
                        proxy = new XmlRpcProxy(url);
                        session.setAttribute("xmlrpcproxy", proxy);
                 }catch (NoClassDefFoundError e){
                        e.printStackTrace();
                 }catch (MalformedURLException e) {
                        e.printStackTrace();
                 }catch (Exception e) {
                        System.out.println("[SendXmlRpc] Finner ikke hog på "+Atts.URL_HOG);
                 }
          }
          System.out.println("return proxy;" + proxy);
          return proxy;
    } //getxmlrpcproxy()
 }//SendXmlRpc
```

DBMenu.java

```
package servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import web.DBPage;

/**
 *
 * @author Stianjun(sstrand81@hotmail.com)
 *
*/

public class DBMenu extends HttpServlet {

    public void init() {
          System.out.println("[DBMenu.init()] ");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          doPost(request,response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {

          String html = DBPage.getHTML();
          response.getWriter().print(html);
    }
 }
```

Menu.java

```
package servlet;

/**
 *
 * @author Stianjun(sstrand81@hotmail.com)
 *
*/

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import web.MenuPage;

public class Menu extends HttpServlet {

    public void init() {
          System.out.println("[Menu.init()] ");

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          doGet(request,response);
          System.out.println("[Menu.doPost()] ");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          String html = MenuPage.getHTML();
          response.getWriter().print(html);
          System.out.println("[Menu.doGet()] ");
    }
}
```

ResultDb.java

```
package servlet;

/**
* @author Stianjun(sstrand81@hotmail.com)
*/

import database.DBQueries;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import web.DBResult;

public class ResultDb extends HttpServlet {

    public void init() {
          System.out.println("[ResultDb.init()] ");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          doPost(request,response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {

          String xml = request.getParameter("query"); //henter inn den valgte spørringen
          String xml2 = request.getParameter("query2"); //henter inn den valgte spørringen
          String result;
          result = DBQueries.Queries(xml, xml2); //sender informasjon for å velge riktig spørring
          response.getWriter().write(DBResult.getPage(result)); //Skriver ut resultatet i Servleten

    }
}
```

MenuPage.java

```
package web;

/**
 *
 * @author Stianjun(sstrand81@hotmail.com)
 *
*/

public class MenuPage {

    public static String getHTML() {
         System.out.println("Laster inn MenySiden");
         return getPage(null);
    } //getHTML

    public static String getPage(String xml) {
         String side1 = "" +
         "<HTML>"+
                "<HEAD>"+
                       "<TITLE>SoccerFinder 2005</TITLE>"+
                "</HEAD>"+
                "<CENTER>"+
                "<BODY>"+
         "<TABLE  BORDER=0 FRAME=BOX RULES=NONE WIDTH=50% VSPACE=10>"+
                "<TR>"+
                       "<TD>"+
                       "<BR>"+
                              "<img src='http://www.stud.ntnu.no/~stianjun/SoccerFinder.JPG'>" +
                       "</TD>"+
                "</TR>"+
                "<TR>"+
                "<TR>"+
                       "<TH>"+
                              "<CENTER>"+"<FONT size=4>Velg ønsket funksjon under:</FONT>"+"</CENTER><BR>"+
                       "</TH>"+
                "</TR>"+
                "<TR>"+
                "<TR>"+
                "<TR>"+
                       "<TD>"+
                       "<CENTER>"+"<FONT size=3>Skriv inn fotballsetninga du vil analysere</FONT>"+"</CENTER>"+
                       "</TD>"+
                "</TR>"+
                "</TR>"+
                "</TR>"+
                "<TR>" +
                       "<TD>" +
                              "<CENTER>"+"<form action=\"send\" method=\"get\">" +"</CENTER>"+
                              "<CENTER>"+"<input type=\"text\" name=\"text\" size=\"50\"/>"+"</CENTER>"+
                              "<CENTER>"+"<input type=\"submit\">"+"</CENTER>"+
                              "</form>" +
                       "</TD>" +
                "</TR>" +
                "<TR>"+
                "<TR>"+
                       "<TD>"+
                "<CENTER>"+"<FONT size=4> <a href='showDB'>Utfør søk i Databasen</a></FONT>"+"</CENTER>"+
                       "</TD>"+
                "</TR>"+
                "</TR>"+
                "</TR>"+
                "<TR>" +
                "</TR>" +

                "</TABLE>"+

                "</BODY>"+

         "</HTML>";

         return side1;
    } //getPage()
 }
```

DBPage.java

```
package web;

/**
 *
 * @author Stianjun(sstrand81@hotmail.com)
 *
*/

import variabler.Atts;

public class DBPage {

    public static String getHTML() {
          System.out.println("Laster inn Database-siden!!");
          return getPage(null);
    }//getHTML

    public static String getPage(String xml) {
          String side1 = "" +
          "<HTML>"+
                  "<HEAD>"+
                          "<TITLE>SoccerFinder 2005</TITLE>"+
                  "</HEAD>"+
                  "<CENTER>"+
                  "<BODY>"+
                  "<TABLE>"+
                  "<TR>"+
                          "<TD>"+
                          "<BR>"+
                          "<img src='http://www.stud.ntnu.no/~stianjun/SoccerFinder.JPG'>" +
                          "<BR>"+
                          "<BR>"+
                          "</TD>"+
                  "</TR>"+

                  "<TR>"+
                          "<TD>"+
                                  "<form action=\"displayDB\" method=\"get\">"+

                                  "<CENTER><SELECT name='query'>"+
                                  "<OPTION>"+Atts.hendelse1 +
                                  "<OPTION>"+Atts.hendelse2 +
                                  "</SELECT> "+
                                  "<CENTER><SELECT name='query2'>"+
                                  "<OPTION>"+Atts.hendelse+
                                  "<OPTION>"+Atts.antallBytter+
                                  "<OPTION>"+Atts.antallYellowCard +
                                  "<OPTION>"+Atts.antallRedCard+
                                  "<OPTION>"+Atts.antallGoal+
                                  "</SELECT> "+

                                  "<BR><INPUT type='submit' name='spørring' value='Vi innholdet i databasen!'>"+
                                  "<BR></CENTER></FORM>"+
                          "</TD>"+
                  "</TR>"+
                  "</TR>"+
                  "<tr><td><CENTER><a href='start'>Tilbake til menyen.</CENTER></a></td></tr>" +

                 "</TABLE>"+

                  "</BODY>"+

          "</HTML>";

          return side1;
    } //getPage()
 }
```

ResultPage.java

```
package web;

/**
 *
 * @author Stianjun(sstrand81@hotmail.com)
 *
*/

public class ResultPage {

    public static String getHTML() {
         System.out.println("getHTML");
         return getPage(null);
    }//getHTML()

    public static String getPage(String xml) {
         String side2 = "" +
         "<HTML>"+
                 "<HEAD>"+
                         "<TITLE>SoccerFinder 2005</TITLE>"+
                 "</HEAD>"+
                 "<CENTER>"+
                 "<BODY>"+
                 "<TABLE>"+
                 "<TR>"+
                         "<TD>"+
                         "<BR>"+
                         "<img src='http://www.stud.ntnu.no/~stianjun/SoccerFinder.JPG'>" +
                         "<BR>"+
                         "<BR>"+
                         "</TD>"+
                 "</TR>"+
                 "<TR>"+
                 "<TR>"+
                 "<TR>"+
                         "<TD>"+
                                 "<CENTER>"+"<FONT size=4>Resultatet fra søket er:</FONT>"+"</CENTER>"+
                         "</TD>"+
                 "</TR>"+
                 "</TR>"+
                 "</TR>"+
                 "<TR>" +
                         "<td>" +
                                 "<BR><CENTER>"+xml+"</CENTER><BR>" +
                         "</td>" +
                 "</tr>" +
                 "<tr><td><CENTER><a href='start'>Tilbake til menyen.</CENTER></a></td></tr>" +

                 "</TABLE>"+

                 "</BODY>"+

         "</HTML>";

         return side2;
    }//getPage()
 }//ResultPage
```

XmlRpcProxy.java

```
package xmlrpc;

/**
 *
 * @author stianjun (sstrand81@hotmail.com)
 *
*/

import java.io.IOException;
import java.net.MalformedURLException;
import java.util.Vector;
import javax.servlet.http.HttpSessionBindingEvent;
import javax.servlet.http.HttpSessionBindingListener;
import org.apache.xmlrpc.XmlRpcClient;
import org.apache.xmlrpc.XmlRpcException;

public class XmlRpcProxy implements HttpSessionBindingListener {

    private XmlRpcClient xmlrpc;
    private String sessId;
    private String annotCollId;
    private String initAnnotId;

    public  XmlRpcProxy(String url) throws Exception, MalformedURLException {
          xmlrpc = new XmlRpcClient( url );
          try {
                if ( !"Hello".equals( sayHello() ) ) {
                      throw new Exception( "HoG'en finness ikke på: " + url + "!" );
                }
          } catch ( Exception e ) {
                throw new Exception( "HoG'en finness ikke på: " + url + "!", e );
          }
    }

    public void startSession() throws Exception {
          System.out.println( "[XmlRpcProxy.startSession] " );
          Vector params = new Vector();
          params.add( "conf/no_xmlrpcsession.cfg" ); // no_xmlrpcsession.cfg = konfigurasjonsfilplassering
          sessId = execute( "createSession", params );
          Vector params2 = new Vector();
          params2.add( sessId );
          annotCollId = execute( "createAnnotationCollection", params2 );
          System.out.println( "[XmlRpcProxy.startSession()] Session " + sessId + " started" );
    }

    public void endSession() {
          Vector params = new Vector();
          params.add( sessId );
          if ( sessId != null ) { // Close the session only if a session has been opened
                try {
                      xmlrpc.execute( "mocoman.closeSession", params );
                      System.out.println( "[XmlRpcProxy.endSession()] Session ended" );
                } catch ( XmlRpcException e ) {
                      e.printStackTrace();
                } catch ( IOException e ) {
                      e.printStackTrace();
                }
          }
    }

    public String sayHello() throws Exception {
          Vector params = new Vector();
          return execute( "sayHello", params );
    }

    public String analyse(String input) throws Exception {
          System.out.println( "[XmlRpcProxy.analyse()] Sender " + "'" + input + "'" + " for analysering" );
          String result = null;
          Vector params = new Vector();
          params.add( sessId );
          params.add( annotCollId );
          params.add( input );
          params.add( "no" ); // no = norsk
          params.add( "SoccerFinder" );
          initAnnotId = execute( "createInitialAnnotation", params );
```

```
        Vector params2 = new Vector();
        params2.add( sessId );
        params2.add( annotCollId );
        params2.add( initAnnotId );
        params2.add( new Integer( 0 ) );
        params2.add( new Integer( 1 ) );
        params2.add( new Integer( 100 ) );
        result = execute2( "analyse", params2 );
        return result;
    }

    private String execute(String methodName, Vector params) throws Exception {
        String result = null;
        try {

                String encodedString = (String) xmlrpc.execute( "mocoman." + methodName, params );
                byte[] bytes = encodedString.getBytes();
                result = new String( bytes, "UTF-8" );
                System.out.println( "[XmlRpcProxy.execute()] " + result );
                if ( "".equals( result ) )
                        System.out.println( "[XmlRpcProxy.execute()] Vi fikk ikke noe svar" );
        } catch ( Exception e ) {
                throw new Exception( e );
        }
        return result;
    }
//en ekstra metode så jeg slipper å se utskrifta hver gang...
//men, greit å se det på de intielle-metodene til HoG

    private String execute2(String methodName, Vector params) throws Exception {
        String result = null;
        try {

                String encodedString = (String) xmlrpc.execute( "mocoman." + methodName, params );
                byte[] bytes = encodedString.getBytes();
                result = new String( bytes, "UTF-8" );
                System.out.println( "[XmlRpcProxy.execute2()] Siste gangen!");
                if ( "".equals( result ) )
                        System.out.println( "[XmlRpcProxy.execute()] Vi fikk ikke noe svar" );
        } catch ( Exception e ) {
                throw new Exception( e );
        }
        return result;
    }
/*
   * (non-Javadoc)
   *
   * @see javax.servlet.http.HttpSessionBindingListener#valueBound(javax.servlet.http.HttpSessionBindingEvent)
   */
    public void valueBound(HttpSessionBindingEvent arg0) {
        try {
                startSession();
        } catch ( Exception e ) {
                e.printStackTrace();
        }
    }
/*
   * (non-Javadoc)
   *
   * @see javax.servlet.http.HttpSessionBindingListener#valueUnbound(javax.servlet.http.HttpSessionBindingEvent)
   */
    public void valueUnbound(HttpSessionBindingEvent arg0) {
        endSession();
    }
 }
```

getData.java

```
package filter;

/**
 *
 * @author stianjun (sstrand81@hotmail.com)
 *
*/

import database.StoreRmrs;
import filter.RmrsFilter;
import variabler.Atts;

public class GetData {

    public static void getRmrsData(String rmrs, String query) {

          //lagrer rmrs-strukturen i database-tabellen SoccerFinder.rmrs.
          System.out.println("[GetDATA.GetRmrsData()] RMRS= " + rmrs + " Query= " + query);
          StoreRmrs.insertRmrs(rmrs, query);

          //Lagrer de forskjellige delen av hendelsen i en database-tabell.
          String constant, constant2 = "";
          constant = RmrsFilter.XmlFilter(rmrs, Atts.XslSourceNames);
          constant2 = constant.substring(38);
          System.out.println("Etter å ha filtrert igjen <Constant>: " + constant2);

          String ARG11, ARG11_2 = "";
          ARG11 = RmrsFilter.XmlFilter(rmrs, Atts.XslSourceARG11);
          ARG11_2 = ARG11.substring(38);
          System.out.println("Etter å ha filtrert igjen <ARG11>: " + ARG11_2);

          String ARG12, ARG12_2 = "";
          ARG12 = RmrsFilter.XmlFilter(rmrs, Atts.XslSourceARG12);
          ARG12_2 = ARG12.substring(38);
          System.out.println("Etter å ha filtrert igjen <ARG12>: " + ARG12_2);

          String ARG2, ARG2_2 = "";
          ARG2 = RmrsFilter.XmlFilter(rmrs, Atts.XslSourceARG2);
          ARG2_2 = ARG2.substring(38);
          System.out.println("Etter å ha filtrert igjen <ARG2>: " + ARG2_2);


          StoreRmrs.insertData(constant2, ARG11_2, ARG12_2, ARG2_2);
/*
          String Constant, Constant_ = "";
          Constant = RmrsFilter.XmlFilter(rmrs, Atts.XslSourceAlt1); //filtrerer ut navnet fra rmrs-
representasjonen.
          Constant_ = Constant.substring(38);
          System.out.println("Etter å ha filtrert igjen <Hele setningen>: " + Constant_);
*/
    }//getRMRSData
 }//class GetData
```

rmrsFilter.java

```
package filter;

/**
 *
 * @author stianjun (sstrand81@hotmail.com)
 *
*/

import java.io.StringReader;
import java.io.StringWriter;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

public class RmrsFilter {
```

```
    public static String XmlFilter(String rmrsXml, String file) {

          //Henter inn xsl fila vår ..\WEB-INF\rmrsFilter.xsl
          StreamSource xslSource = new StreamSource(file);
          //System.out.println("xslSource = " + xslSource);

          //Henter inn rmrs resultatet
          StreamSource rmrsSource = new StreamSource(new StringReader(rmrsXml));
          //System.out.println("rmrsSource = " + rmrsSource);

          //oppretter result for å ta vare på resultatet
          StringWriter rmrsResult = new StringWriter();
          StreamResult result = new StreamResult(rmrsResult);

          //oppretter en instanse av TransformerFactory, filter1
          TransformerFactory filter1 = TransformerFactory.newInstance();

          //bruker TransformerFactory til å prosessere en .xsl og generere en 'transformer'
          try{

          Transformer transformer = filter1.newTransformer(xslSource); //xsl-stylesheet
          //transformerer xmlrmrs-kilden og sender output til et resultat objekt.
          transformer.transform(rmrsSource, result);

          }catch(TransformerConfigurationException e){
                 System.out.println("TransformerConfigurationException " + '\n');
                 e.printStackTrace();

          }catch(TransformerException e){
                 System.out.println("TransformerException " + '\n');
                 e.printStackTrace();
          }

          return rmrsResult.getBuffer().toString();

    }//XmlFilter
 }//RmrsFilter
```

StoreRmrs.java

```
package database;

/**
 *
 * @author stianjun(sstrand81@hotmail.com)
 *
*/

import java.sql.*;
import variabler.Atts;

public class StoreRmrs {

    public static void insertRmrs(String rmrs, String query) {
          Statement stmt = null;
          ResultSet rs = null;
          final Connection soccerDB;
          String output = "";
          boolean compare = false;
          boolean kamp = false;
          try {
                 //Laster inn JDBC driveren
                 Class.forName("com.mysql.jdbc.Driver").newInstance();
                 //Setter opp en forbindelse med databsen
                soccerDB = DriverManager.getConnection("jdbc:mysql://" + Atts.SqlHost, Atts.Bruker, Atts.Pass);
                 stmt = soccerDB.createStatement();

                 String leggInn = "INSERT INTO rmrs(rmrsData, setning) VALUES("+"'"+rmrs+"','"+"'"+query+"'"+");
                 stmt.executeUpdate(leggInn);

              System.out.println("RMRS-representasjonen blei lagt inn i databasen");
              soccerDB.close();
              if(stmt != null)
              try
                 {
                 stmt.close();
```

```
            }catch(SQLException e){
               e.printStackTrace();
            }
         }//try

         catch(Exception ex){
                System.err.println("Got an exception! ");
                System.err.println(ex.getMessage());
         }//catch
    }//InserData

    public static void insertData(String konstant, String ARG1, String ARG2, String ARG3) {
         Statement stmt = null;
         ResultSet rs = null;
         final Connection soccerDB;
         String output = "";
         boolean compare = false;
         boolean kamp = false;
         try {
                //Laster inn JDBC driveren
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                //Setter opp en forbindelse med databsen
               soccerDB = DriverManager.getConnection("jdbc:mysql://" + Atts.SqlHost, Atts.Bruker, Atts.Pass);
                stmt = soccerDB.createStatement();

                String leggInn = "INSERT INTO hendelse(Actor, utkom, hendelse) VALUES
                (" + "'" + konstant + "'," + "'" + ARG1 + " " + ARG2 + "'," + "'" + ARG3 +  "'" + "); ";

                stmt.executeUpdate(leggInn);

                System.out.println("Dataene ble lagt inn i databasen uten feil");
                soccerDB.close();
                if(stmt != null)
                       try
                       {
                              stmt.close();
                       }catch(SQLException e){
                              e.printStackTrace();
                       }
         }//try

         catch(Exception ex){
                System.err.println("Got an exception! ");
                System.err.println(ex.getMessage());
         }//catch
    }//insertData(constant2, ARG11_2, ARG12_2, ARG2_2);
}//StoreRmrs
```

DBQueries.java

```
package database;

/**
 *
 * @author Stianjun (sstrand81@hotmail.com)
 *
*/

import java.sql.*;
import variabler.Atts;
import variabler.Queries;

public class DBQueries {

  public static String Queries(String result, String result2) {

         Statement stmt = null;
         ResultSet rs = null;
         final Connection soccerDB;
         String output = "";
         boolean compare = false;
         boolean kamp, hendelse = false;

         try {
                //Laster inn JDBC driveren
                Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
                //Setter opp en forbindelse med databsen
              soccerDB = DriverManager.getConnection("jdbc:mysql://" + Atts.SqlHost, Atts.Bruker, Atts.Pass);
              stmt = soccerDB.createStatement();

                //Finner hvem KampNR vi vil søke i
                if((kamp = result.startsWith("1")) == true)
                        Queries.KampNR = "1";
                else if ((kamp = result.startsWith("2")) == true)
                        Queries.KampNR = "2";

                //Finner hvem hendelse det er
                if((hendelse = result.endsWith("1")) == true)
                        Queries.hendelseNR = "1";
                else if ((hendelse = result.endsWith("2")) == true)
                        Queries.hendelseNR = "2";

                if((compare = (result2.equals(Atts.hendelse)))== true){
                rs = stmt.executeQuery(Queries.hendelseQuery + Queries.hendelseNR);

                if (rs != null){
                              while (rs.next()){
                                output += rs.getString("Actor") + "\n";
                                output += rs.getString("utkom") + "\n";
                                output += rs.getString("hendelse") + "\n";
                                //output += rs.getString(;
                                }//while
                              }//if
                soccerDB.close();
            }

            else if((compare = (result2.equals(Atts.antallYellowCard)))== true){
                System.out.println("Spørringen vi ønsker å kjøre er: " + result2 + " " + Queries.KampNR);
                rs = stmt.executeQuery(Queries.yellowCardQuery + Queries.KampNR);

                if (rs != null){
                              while (rs.next()){
                                output += rs.getString("rmrsNR") + "\n";
                                output += rs.getString("setning") + "\n";
                                //output += rs.getString(;
                                }//while
                              }//if
                soccerDB.close();
            }
            else if((compare = (result2.equals(Atts.antallGoal)))== true){
                System.out.println("Spørringen vi ønsker å kjøre er: " + result2 +" + "+compare);
                rs = stmt.executeQuery(Queries.goalQuery);

                if (rs != null){
                        while (rs.next()){
                                output += rs.getString("") + "\n";
                        }//while
                }//if
                soccerDB.close();
            }
            else if((compare = (result2.equals(Atts.antallRedCard)))== true){
                System.out.println("Spørringen vi ønsker å kjøre er: " + result2 +" + "+compare);
                rs = stmt.executeQuery(Queries.redCardQuery);

                if (rs != null){
                        while (rs.next()){
                                output += rs.getString("") + "\n";
                                }//while
                        }//if
                soccerDB.close();
                }
            else if((compare = (result2.equals(Atts.antallBytter)))== true){
                System.out.println("Spørringen vi ønsker å kjøre er: " + result2 +" + "+compare);
                rs = stmt.executeQuery(Queries.bytterQuery);

                if (rs != null){
                        while (rs.next()){
                                output += rs.getString("") + "\n";
                                }//while
                        }//if
                soccerDB.close();
                }
            else if(compare == false){
```

```
                System.out.println("Spørringen finnes ikke!"  +" + " +compare);
                rs = null;
                soccerDB.close();
            }//else

        }//try

        catch(Exception ex){
                System.err.println("Got an exception! ");
           System.err.println(ex.getMessage());
        }//catch

        if(stmt != null)
        try{stmt.close();
        }catch(SQLException e){
        e.printStackTrace();
        }
        if(output == null || output == "" || output == " "){
                output = "Det oppstod en feil, eller databsen er tom.";
                //return output;
        }//if
        else if (output != null){
                System.out.println("Resultatet fra spørringen er: " + output);
        }

        return output;

   }//Queries
}//DBQueries
```

Atts.java

```
package variabler;

/**
 *
 * @author stianjun (sstrand81@hotmail.com)
 *
*/

public class Atts {

  public static final String URL_HOG = "http://edvarda.hf.ntnu.no:8434/";

  public static final String SqlHost = "localhost:3306/SoccerFinder"; //host og database
    public static final String Bruker = "stian";
    public static final String Pass = "tottenham";

  public static final String antallYellowCard = "Antall gule kort";
    public static final String antallGoal = "Antall mål";
    public static final String antallRedCard = "Antall røde kort";
    public static final String antallBytter = "Antall bytter";
    public static final String kampID1 = "1 Skottland - Norge";
    public static final String kampID2 = "2 Italia - Norge";
    public static final String hendelse1 = "Hendelse 1";
    public static final String hendelse2 = "Hendelse 2";
    public static final String hendelse = "Vis Hendelse";

    public static final String XslSource = "C:/…/SoccerFinder/WEB-INF/rmrsFilter.xslt";
    public static final String XslSourceNames = "C:/…/SoccerFinder/WEB-INF/rmrsFilterNames.xslt";
    public static final String XslSourceARG2 = "C:/…/SoccerFinder/WEB-INF/rmrsFilterARG2.xslt";
    public static final String XslSourceARG11 = "C:/…/SoccerFinder/WEB-INF/rmrsFilterARG11.xslt";
    public static final String XslSourceARG12 = "C:/…/SoccerFinder/WEB-INF/rmrsFilterARG12.xslt";
    public static final String XslSourceAlt1 = "C:/…/SoccerFinder/WEB-INF/rmrsFilterAlt.xslt";

 }
```

web.xml

```
<?xml version="1.0" ?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-
app_2_3.dtd">

<web-app>

  <servlet>
        <servlet-name>Menu</servlet-name>
        <display-name>Menu</display-name>
        <description>The menu of SoccerFinder</description>
        <servlet-class>servlet.Menu</servlet-class>
        <load-on-startup></load-on-startup>
    </servlet>

    <servlet>
        <servlet-name>XmlRpcClient</servlet-name>
        <display-name>XmlRpcClient</display-name>
        <description>Parses form data into xml and sends xml-rpc to HoG</description>
        <servlet-class>servlet.SendXmlRpc</servlet-class>
        <init-param>
          <param-name>home</param-name>
          <param-value>C:\Documents and Settings\stianjun\Desktop\Hovedprosjekt\SoccerFinder\</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

     <servlet>
        <servlet-name>DbMenu</servlet-name>
        <display-name>DbMenu</display-name>
        <description>Dispays Db query menu</description>
        <servlet-class>servlet.DBMenu</servlet-class>
        <load-on-startup>5</load-on-startup>
    </servlet>

    <servlet>
          <servlet-name>ResultDb</servlet-name>
          <display-name>ResultDb</display-name>
          <description>Displays results from query</description>
          <servlet-class>servlet.ResultDb</servlet-class>
          <load-on-startup>6</load-on-startup>
    </servlet>

     <servlet-mapping>
        <servlet-name>XmlRpcClient</servlet-name>
        <url-pattern>/send</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>XmlRpcClient</servlet-name>
        <url-pattern>/get</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>Menu</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>DbMenu</servlet-name>
        <url-pattern>/showDB</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
          <servlet-name>ResultDb</servlet-name>
          <url-pattern>/displayDB</url-pattern>
    </servlet-mapping>

</web-app>
```

12.0 Vedlegg E – XSLT-kode

rmrsFilter.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xalan="http://xml.apache.org/xslt">

  <xsl:output method="xml"
              encoding="UTF-8"
              indent="yes"
              xalan:indent-amount="2"/>


  <xsl:template match="/"> <!--hele xml-fila-->
         <SoccerFinder_result>
         <xsl:for-each select="/pet/rmrs/ep"> <!--Elementet vi vil hente data fra-->
                <ep>
                <xsl:copy-of select="gpred"/>
                <xsl:copy-of select="label"/>
                <xsl:copy-of select="var"/>
                <xsl:copy-of select="realpred"/>
                </ep>
         </xsl:for-each>


         <xsl:for-each select="/pet/rmrs/rarg[rargname != 'BODY' and rargname != 'RSTR']">
                <rarg>
                <xsl:copy-of select="rargname"/>
                <xsl:copy-of select="label"/>
                <xsl:copy-of select="var"/>
                <xsl:copy-of select="constant"/>
                </rarg>
         </xsl:for-each>


         </SoccerFinder_result>
  </xsl:template>

</xsl:stylesheet>
```

rmrsFilterNames.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xalan="http://xml.apache.org/xslt">

  <xsl:template match="/">
       <xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'CARG' and not(contains(constant, '-rel'))] ">

                <xsl:value-of select="constant" />

       </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

rmrsFilterArg1.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">

  <xsl:for-each select="/SoccerFinder_result">

  <xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'ARG1']">

          <xsl:variable name="vid">
                  <xsl:value-of select="label/@vid"/>
          </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $vid]">

                  <xsl:if test="realpred/@lemma">
                          <xsl:value-of select="realpred/@lemma" />
                  </xsl:if>

          </xsl:for-each>
  </xsl:for-each>
  </xsl:for-each>

        </xsl:template>
</xsl:stylesheet>
```

rmrsFilterArg2.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">

  <xsl:for-each select="/SoccerFinder_result">

  <xsl:for-each select="rarg[rargname = 'ARG2']">

          <xsl:variable name="vid">
                  <xsl:value-of select="var/@vid"/>
          </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/ep[var/@vid = $vid]">

                  <xsl:if test="realpred/@lemma">
                          <xsl:value-of select="realpred/@lemma" />
                  </xsl:if>

          </xsl:for-each>
        </xsl:for-each>
  </xsl:for-each>

  </xsl:template>
</xsl:stylesheet>
```

rmrsFilterTime.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">

          <xsl:for-each select="/SoccerFinder_result">

          <xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'ARG1']">

                  <xsl:variable name="vid2">
                          <xsl:value-of select="label/@vid"/>
                  </xsl:variable>

                  <xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $vid2]">

                  <xsl:value-of select="gpred" />

                  </xsl:for-each>

  </xsl:for-each>
  </xsl:for-each>

  </xsl:template>
</xsl:stylesheet>
```

rmrsFilterAlt.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xalan="http://xml.apache.org/xslt">

  <xsl:template match="/">

  <xsl:for-each select="/SoccerFinder_result/rarg[rargname = 'CARG' and not(contains(constant, '-rel'))] ">

          <xsl:value-of select="label/@vid" />
          <xsl:value-of select="constant" />

          <xsl:variable name="labelvid1">
                  <xsl:value-of select="label/@vid" />
          </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $labelvid1]">

                  <xsl:variable name="varvid1">
                          <xsl:value-of select="var/@vid"/>
                  </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/rarg[rargname='ARG1' and var/@vid = $varvid1]">

                  <xsl:variable name="labelvid2">
                          <xsl:value-of select="label/@vid"/>
                  </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/ep[label/@vid = $labelvid2]">

                  <xsl:if test="realpred/@lemma">
                          <xsl:value-of select="realpred/@lemma" />
                  </xsl:if>

          </xsl:for-each>

          <xsl:for-each select="/SoccerFinder_result/rarg[rargname='ARG2' and label/@vid = $labelvid2]">

                  <xsl:variable name="varvid2">
                          <xsl:value-of select="var/@vid" />
                  </xsl:variable>

          <xsl:for-each select="/SoccerFinder_result/ep[var/@vid = $varvid2]">

                  <xsl:if test="realpred/@lemma">
                          <xsl:value-of select="realpred/@lemma" />
                  </xsl:if>
```

```
                <xsl:variable name="vidx">
                        <xsl:value-of select="var/@vid" />
                </xsl:variable>

                <xsl:for-each select="/SoccerFinder_result/rarg[(rargname = 'ARG1' and var/@vid = $vidx)]">

                <xsl:variable name="labelx">
                        <xsl:value-of select="label/@vid" />
                </xsl:variable>

        <xsl:for-each select="/SoccerFinder_result/ep[realpred = 'lemma' and (label/@vid = $labelx)]">

                <xsl:if test="realpred/@lemma">
                        <xsl:value-of select="realpred/@lemma" />
                </xsl:if>

        </xsl:for-each>
        </xsl:for-each>

        </xsl:for-each>
        </xsl:for-each>

        </xsl:for-each>

        </xsl:for-each>
  </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```