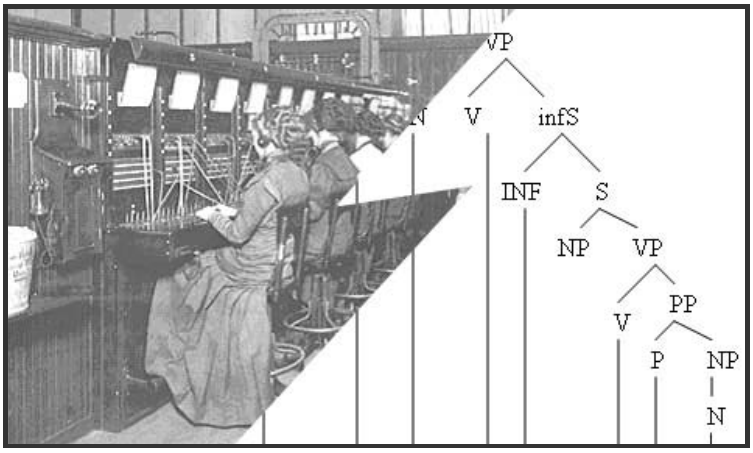
Det forståingsfulle opplysningssystemet



Eit studium av BussTUC, med fokus på ambiguitetsrelaterte problem

Bente Hole

Hovudfagsoppgåve i datalingvistikk og språkteknologi
Institutt for lingvistikk
Universitetet i Bergen
Februar 2004

Forord

Denne hovudfagsoppgåva i datalingvistikk og språkteknologi er utført og avlagt ved seksjon for lingvistiske fag, Institutt for lingvistikk og litteraturvitenskap, Universitetet i Bergen. Arbeidet vart påbyrja vårsemesteret 2003.

Eg vil først og fremst få takka vegleiarane mine for all hjelp og støtte: Takk til hovudvegleiar ved UiB, førsteamanuensis dr.art. Lars Johnsen, for å ha ofra mykje tid og krefter, kome med gode råd og vink og for å ha utvist akkurat den mengda med tålmod som skal til. Takk til bivegleiar ved NTNU, førsteamanuensis Tore Amble, for å ha avsett tid til både besøk og til å svara på diverse e-post og telefonar, samt – ikkje minst – for å ha gitt meg tilgang til heile BussTUC-systemet og det eg har tronge av korpusfiler.

Vidare fortener medstudentar, familie, sambuar og vener ein takk – for kantinestunder og samhald, for evna til å halda ut og for å vere der når det trengs. Ein spesiell takk til Magnus, Tom og ikkje minst Ingunn for å ha slitt seg gjennom diverse sider for dei heller lite spanande materiale.

Samandrag

Denne hovudfagsoppgåva tek utgangspunkt i bussruteopplysningssystemet BussTUC – eit databasesystem med eit grensesnitt basert på innputt i from av naturleg språk, det vil seie norske og engelske setningar. Systemet vart utvikla i tilknyting til NTNU i løpet av 1996 og er i dag tilgjengeleg via Internett og SMS. Oppgåva tek føre seg problemområde det framleis gjenstår å få bukt med i tilknyting til sjølve grensesnittet og språkprosesseringa; I første rekke problem relaterte til disambiguering. Ulike løysingstilnærmingar vert presenterte og enkelte av desse prøvde ut i praksis. Fokus vert i stor grad retta mot parsingdelen av systemet: Den opprinnelege parsaren, ein innebygd prologparsar som arbeider topp-ned med tilbakesporing, vert bytta ut med ein kartparsar. Målet er å finna ut om ein kartparsingmodul kan føra til betre resultat med omsyn til fleirtydige og ufullstendige innputt, samt om det er mogleg å få han like effektiv som den opprinnelege parsingmodulen.

Nøkkelord: Språkprosessering, ambiguitet, kartparsing, naturlegspråklege grensesnitt, ekstraposisjonsgrammatikk, kategorialgrammatikk, databasegrensesnitt baserte på naturleg språk

Abstract

The point of departure for this thesis is the bus route information system BusTUC – a database system with an interface based on input in natural language, i.e. sentences in Norwegian and English. The system was developed in connection to NTNU during 1996 and is now available to the public both on the Internet and per SMS. The thesis discusses problem areas tied to the interface and the language processing part of the system which still remain to be taken care of; first and foremost problems related to disambiguation. Different potential solutions are presented and some of these tried out in practice. The main focus is directed towards the parsing part of the system. The original parser, a built-in Prolog parser working top-down with backtracking, is replaced by a chart parser. The goal is to make a parsing module that leads to better results with regard to ambiguous input, and that works as effective as the original parser.

Key words: Natural Language Processing (NLP), ambiguity, chart parsing, Natural Language Interfaces (NLI), extraposition grammar, categorial grammar, Natural Language Interfaces to Databases (NLIBD)

Innhald

Innleiing

Endringar i brukarmiljø og -forventningar har ført til at formene for menneske-maskin-interaksjon har endra karakter. Blant anna har meir brukarvennlege og intuitive grensesnitt, som direkte manipulerbare og menybaserte grensesnitt, vorte introduserte. Dette har i sin tur resultert i at omfanget og bruken av meir tradisjonelle interaksjonformer, i første rekke grensesnitt baserte på bruk av kommandospråk, har minka. Språk i form av vanleg engelsk, norsk eller liknande har vorte teke i bruk i staden for dei mindre memorerbare og logiske kommandospråka som eit naturleg ledd i denne utviklinga. Grensesnitt baserte på naturleg språk, NLI (Natural Language Interfaces), fekk ikkje den pangstarten det på 80-talet vart predikert at dei skulle få og kjem kanskje aldri til å verta så utbreidde som mange på den tida trudde. Det må likevel kunna seiast at dei har vorte stadig fleire, til dels betre og noko meir vanlege i den seinare tid. I dag finn ein NLI for både munnleg og skriftleg innputt i tilknyting til alt frå teikneprogram og søkemotorar til VR (Virtual Reality)-system og robotar[1].

Denne hovudfagsoppgåva tek føre seg grensesnittet til BussTUC-systemet – eit database- og dialogsystem for bussruteopplysning[2]. Versjonen av BussTUC-systemet berekna på skriftlege innputt har vore tilgjengeleg og i bruk over Internett og per SMS sidan 1996 og fungerer rimeleg bra. Det finst likevel enkelte problem det gjenstår å løysa, ein del av desse typisk ambiguitetsrelaterte problem. Ambiguitet, som ein vesentleg og uunngåeleg bestanddel av naturleg språk, er ikkje til å komma utanom når ein jobbar med analyse av språk. Som oppgåva viser finst det eit utal ulike ambiguitetstypar og like mange løysingstilnærmingar. Oppgåva tek føre seg ulike ambiguitetsrelaterte problem og potensielle løysingar aktuelle for BussTUC. Løysingstilnærmingane som vert gjennomgått varierer frå meir direkte og spesifikke tiltak – som å leggja til eller endra reglar i grammatikken – til meir generelle, men hovudfokuset er via parsingdelen av systemet. Ein kartparsar (chart parser), det vil seie ein parsar som lagrar alle ufullstendige og fullstendige analyser han finn i løpet av parsingprosessen, vert integrert og testa ut. Testresultata vert så samanlikna med resultata den originale prologparsaren oppnår på same testsett. Målet er å finna ut om ein kartparsar kan tilføra systemet noko med omsyn til disambiguering, samstundes som effektiviteten vert oppretthalden eller – i beste fall – forbetra.

Oppgåva består av fem hovuddelar og er strukturert som følgjer: Første del er ein introduksjonsdel som inneheld ein kort historikk og bakgrunnsstoff, samt ei oversikt over systemet og teknologien. Deretter følgjer eit kapittel som skildrar behovsanalysen som vart gjennomført, først og fremst for å få eit klarare bilete av kva som faktisk er problematisk og ikkje, men òg for å få ei oversikt over systemet og innsikt i korleis det er bygd opp og fungerer. Påfølgjande kapittel tek føre seg ambiguitet, først på eit generelt, teoretisk grunnlag, deretter i høve til BussTUC-systemet. Arbeidet med parsingdelen vert så presentert i eit eige kapittel, som ei fortsetjing på det føregåande. Siste del tek føre seg fordeler og ulemper i tilknyting til grensesnitt baserte på naturleg språk, sett frå brukaren sin ståstad, og meir generelle løysingsmetodar i tråd med brukarbehov og -studium.

Avslutningsvis bør det nemnast at systemversjonen som ligg til grunn for oppgåva er ein gammal versjon. Grammatikkfila tilhøyrande denne versjonen vart endra av utviklarane siste

[1] Sjå *A Natural Language Interface for Virtual Reality Systems* på http://www.aic.nrl.navy.mil/~severett/vr-report/title.html og *KANTRA: Human-Machine Interaction for Intelligent Robots Using Natural Language* på http://www.dfki.uni-sb.de/vitra/papers/ro-man94/ro-man94.html

[2] TUC står for The Understanding Computer og er namnet på teknologien som ligg til grunn for systemet

gong i desember 2002. Systemet, og då først og fremst grammatikk-, leksikon- og semantikkfiler, har vorte utbetra ein god del sidan oppgåva vart påbyrja. Problema omtala i denne oppgåva er difor ikkje nødvendigvis representative for den utgåva av systemet som er i bruk i dag.

1 Bakgrunn

Første del av dette introduksjonskapitlet tek føre seg NLIDB - databasegrensesnitt basert på naturleg språk - på eit generelt grunnlag. Feltet vert først kort skildra og plassert innan det datalingvistiske området. Språkbaserte grensesnitt vert så samanlikna med andre typar databasegrensesnitt. Deretter følgjer ei kort, historisk oversikt som skildrar utviklinga av NLIDB, frå dei første prototypane vart introduserte seint på 60-talet og fram til dag. Ulike NLIDB-kategoriar vert så presenterte, og til slutt kjem eit avsnitt om kva som må til for å laga gode og effektive NLIDB, sett frå både brukar og utviklar sin ståstad. Mykje av stoffet brukt i første del er henta frå ein introduksjon til NLIDB skriven av Androutsopoulos, Ritchie og Thanisch [Androutsopoulos -95].

Andre del, frå og med delkapittel 1.4, er ei skildring av teknologien og systemet. Her vert TUC-teknologien og BussTUC-systemet presentert i grove trekk. Litt bakgrunnshistorie er teke med og enkelte av hovudkomponentane til systemet – grammatikken, parsaren og semantikkdatabasen – vert skildra i noko meir detalj. Systemet vert plassert i høve til dei ulike systemkategoriane nemnde i førre kapittel. Siste avsnitt tek føre seg hovudpunkta i prosessen frå innputt til utputt. Mykje av materialet i denne delen er henta frå brukarmanualen tilhøyrande BussTUC, 'The Understanding Computer: A Tutorial', skriven av Amble, Ranang og Sætre.

1.1 Natural Language Interfaces to Databases, NLIDB

Eit databasegrensesnitt basert på naturleg språk, NLIDB, er eit system tilrettelagt for at brukaren skal kunna få tilgang til informasjonen lagra i den aktuelle databasen ved hjelp av naturleg språk som til dømes norsk eller engelsk. Føremålet er først og fremst å forenkla interaksjonen mellom menneske og maskin, samt å kunne tilby ein alternativ måte å interagera med databasesystem på. Grensesnitt baserte på naturleg språk er meint å gjera det enklare å bruka system, spesielt kanskje for nye og lågfrekvente brukarar, ganske enkelt ved å tillata brukarane å dra nytte av den naturlege kommunikasjons- og språkevna dei allereie sit inne med. I systemet denne hovudoppgåva omhandlar, BussTUC, gir ein som innputt ei vanleg norsk eller engelsk setning, som "*Når går de 2 neste bussene fra Lade til sentrum?*", og får generert eit utputt basert på den underliggjande bussrutedatabasen (samt ut frå det dåverende tidspunktet): "*Holdeplassen nærmest Lade er LADE ALLE 80. Buss 3 passerer LADE ALLE 80 klokken 1535 og kommer til Munkegata M1 17 minutter senere. Buss 4 passerer LADE ALLE 80 klokken 1545 og kommer til Munkegata M1 18 minutter senere.*"

NLIDB er basert på språkprosessering, vanlegvis forkorta NLP (Natural Language Processing), som i sin tur står svært sentralt innan datalingvistikken. I tillegg til å verta brukte til NLIBD vert NLP-teknikkar nytta til blant anna ekstrahering av informasjon frå tekst, omsetjing, gjenkjenning av tale, samt gjenfinning og framhenting av relevante dokument frå dokumentsamlingar. NLP dekker prosessering av munnleg så vel som skriftleg språk, og systema kan vere opne eller lukka. Eit lukka system er eit system for eit avgrensa domene. Den semantiske interpretasjonen i eit slikt system er som oftast enklare enn for eit ope ettersom domenet, og dermed talet på moglege tydingar, er forholdsvis lite. Eit lukka system mogleggjer som oftast bruken av spesifikke, semantiske templat som forenklar tolkingsprosessen. I eit ope system, derimot, er den semantiske tolkinga normalt tentativ og hovudsakleg basert på leksikalsk semantikk. Det som på engelsk vert kalla QA-system, det vil

seie Question-Answering-system, er døme på opne system[3]. NLIDB-system er typisk lukka system. Dei fleste, inklusive BussTUC, er basert på tekstinnputt, men det finst òg døme på NLIDB-system basert på tale. Appendiks A gir ei oversikt over ulike språkprosesseringstypar. NLIDB, og NLP generelt sett, har ein klar relasjon til andre beslekta fag som menneske-maskin-interaksjon, HCI (Human Computer Interaction), og kunstig intelligens, KI. I følgje Zelkowitz finst det, sett frå eit HCI-perspektiv, to hovudmotivasjonar som ligg til grunn for datalingvistikk og NLP, ein teknologisk og ein vitskapleg: Den teknologiske motivasjonen ligg i å betra kommunikasjonen mellom menneske og maskin. Den vitskaplege er å forstå språkets natur [Zelkowitz -98, kap. 2.1.2]. Utvikling av NLIDB-system kjem først og fremst med under førstnemnde, men kan òg seiast å medverka til ei større forståing av språk.

1.1.1 NLIDB i høve til andre typar databasesystem

Tradisjonelt, det vil seie etter innføringa av den relasjonelle modellen (sjå nedanfor), har databasespørjingar gjerne vorte utførte ved hjelp av databasespråk som SQL (Structured Query Language). Enkelte, særleg mindre og enklare databasesystem, nyttar gjerne – eventuelt bare for delar av systemet eller i tillegg til databasespråket – eit brukargrensesnitt basert på menyval.

Førstnemnde databasespråksystem er raske og fleksible system. Ulempa med desse er at brukarane må læra seg både det aktuelle databasespråket og til dels sjølve databasen å kjenna før dei er i stand til å nytta systemet fullt ut. Brukarane må med andre ord læra seg eit formelt språk, ofte på linje med eit programmeringsspråk med omsyn til kompleksitet, noko som tek tid og krefter.

For menybaserte databasesystem er situasjonen omvendt: Menybaserte system er enkle å bruka, det vil seie brukarane treng minimalt med opplæring og kunnskap om systemet og sjølve databasestrukturen, men til gjengjeld er dei som oftast tregare og mindre fleksible. Skilnaden mellom dei to typane databasesystem, med sine respektive brukargrensesnitt, kjem særskilt til syne så snart det vert tale om meir kompliserte spørjingar.

Databasesystem med grensesnitt basert på naturleg språk, Natural Language Front End Systems som dei òg gjerne vert kalla [Gazdar –89], er meint å kombinera det beste frå begge dei føregåande. Denne typen system tek sikte på å vere fleksible og effektive, samstundes som det vert lagt vekt på – som nemnt ovanfor – at dei skal vere enkle å bruka. Brukarane skal i prinsippet ikkje trenga å ta særleg omsyn til verken databasestruktur eller måten systemet fungerer på.

1.2 Ei historisk oppsummering

Dei første databasesystema som vart tekne i bruk var oppbygde og strukturerte på ein heilt annan måte enn databasesystem av i dag normalt er. Som brukar måtte ein ikkje bare ha god kjennskap til databasestrukturen, ein måtte som oftast òg ha ein god del programmeringskunnskap for i det heile å kunne gjera seg nytte av systema. Det var med andre ord bare det ein kan kalla ekspertbrukarar, i all hovudsak programmerarar, som aksesserte dataa i databasane direkte. Omgrepet 'sluttbrukar', i den forstand det gjerne vert brukt i dag – om vanlege, ikkje-ekspertbrukarar – fanst ganske enkelt ikkje i tilknyting til databasesystem. Tilhøva endra seg kraftig i tida etter at Codd introduserte første utgåve av sin

[3] Sjå *Developing a Web-based Question Answering System* på http://www2002.org/CDROM/poster/203/ for eit døme på eit QA-system.

relasjonelle modell i 1969[4]. Den einaste forma for lagringsstruktur i den relasjonelle modellen er tabellen, noko til og med ikkje-ekspertar kunne forstå seg på. I kjølvatnet av den nye modellen dukka etterkvart dei relativt enkle, deklarative databasespørjingsspråka opp, noko som òg gjorde sitt for å gjera databasesystem tilgjengelege for andre enn ekspertbrukarar.

Brukarvennlege system er eit sentralt stikkord i tilknyting til den vidare utviklinga av databasesystem. Nettopp på grunn av denne store, nye brukarklassen – det vil seie gruppa av ikkje-ekspertbrukarane – som vaks fram frå 1970-talet av vart det nødvendig å fokusera meir på å leggja til rette for sluttbrukaren og gjera systema så enkle å bruka og lett tilgjengelege som mogleg. Det er her NLIDB kjem inn.

Dei første prototypane av NLIDB dukka opp mot slutten av 60-talet. Desse tidlege NLIDB var typisk laga med tanke på spesifikke bruksområde, det vil seie tilpassa ein gitt database. Ei av dei meir kjende utgåvene frå denne tida er Lunar, eit grensesnitt mot ein database bestående av kjemiske analysar av månestein. Mange av systema laga utover på 70-talet, som t.d. Ladder (òg kalla Lifer), nytta semantiske grammatikkar, ein teknikk som ikkje skilde mellom syntaktisk og semantisk prosessering. Ladder nytta til dømes etikettar som *SHIP* og *ATTRIBUTE* i staden for meir generelle, syntaktiske etikettar som *SUBSTANTIV* og *VERB*. Dette oppnådde ein imponerande resultat med, men teknikken gjorde det tungvint og vanskeleg å tilpassa systemet til nye bruksområde, i og med at ein måtte utvikla ein ny grammatikk for kvart nytt system. Ein gjekk etter kvart bort frå semantiske grammatikkar og byrja i større grad å fokusera på det å gjera systema meir portable og moglege å gjenbruka, blant anna ved å skilja den semantiske og syntaktiske prosesseringa frå einannan, samt å laga meir generelle komponentar. Planes og Philiqa1 er døme på to andre NLIDB som vart laga mot slutten av 1970-åra.

Pereira og Warren sitt NLIDB CHAT-80 er rekna som eit av dei mest kjente systema frå tidleg 80-tal[5]. CHAT-80 nytta mellomliggjande representasjonar (meir om dette i avsnitt 1.3 nedanfor) og var eit av dei første døma på prologbasert språkprosessering. Systemet omsette spørsmålsinnputt på engelsk til prologuttrykk, som så vart evaluerte mot ein prologdatabase. Mange av komponentane i CHAT-80-systemet er laga på ein slik måte at dei er relativt enkle å tilpassa andre bruksområde. Koden vart gjort tilgjengeleg og har i ettertid ligge til grunn for mange andre, eksperimentelle NLIDB, deriblant BussTUC (sjå delkapitla 1.4 og 1.5).

NLIDB utgjorde eit populært forskingsfelt på midten av 80-talet og svært mange prototypar vart laga i løpet av denne perioden. Mykje av forskinga fokuserte stadig på å utvikla metodar for å kunne laga meir generelle og portable system. Team-systemet er eit resultat av dette. Team vart designa slik at det enkelt kunne konfigurerast, sjølv av databaseadministratorar utan kjennskap til NLIDB.

I tillegg til fokuseringa på portabilitet finst det eit par andre utviklingslinjer som etterkvart gjorde seg gjeldande. For det første vart det laga ein del system som opna for ei større grad av brukarinvolvering. For det andre byrja ein å laga og prøva ut det ein kan kalla fleirapplikasjonssystem.

[4] Meir om Codd og den relasjonelle modellen finst i *The Birth of the Relational Modell* på http://www.aisintl.com/case/library/Date_Birth%20of%20the%20Relational%20Model-1.html

[5] Pereira, Fernando C. N.: *Fernando Pereira's old code*. Tilgjengeleg på http://www.cis.upenn.edu/~pereira/oldies.html

Ask skilde seg noko frå tidlegare system ved at sluttbrukarane, kva tid som helst under interaksjonen, hadde høve til å læra systemet nye ord og konsept. Ask var meir som eit informasjonsbehandlingssystem enn eit vanleg NLIDB, med eigen innebygd database, samt evne til å interagera med fleire eksterne databasar, e-postprogram med meir. Brukarane fekk tilgang til applikasjonane tilknytte systemet ved å tasta inn innputt i form av vanlege, engelske setningar. Janus er eit anna system frå same periode som på liknande vis grensa mot fleire ulike applikasjonar, som databasar, ekspertsystem, grafiske hjelpemiddel, osb. Dei underliggjande systema tok, når nødvendig, del i evalueringa av brukarinnputta. Andre system utvikla på 80-talet inkluderer Datalog3, Eufid, Ldc, Tqa, Teli og mange andre.

1.2.1 Fram mot i dag

Til tross for oppblomstringa i 1980-åra fekk ikkje databasegrensesnitta baserte på naturleg språk den kommersielle aksepten og utbreiinga som mange nok hadde venta. Ovum Ltd, eit av dei større rådgjevingsselskapa innan telekommunikasjon, programvare og IT-tenester i Europa, spådde at grensesnitt baserte på naturleg språk ville vere eit standard alternativ for brukarar av databasehandteringssystem innan 1987. I dag finst det kommersielt tilgjengelege og meir eller mindre suksessrike NLIDB på marknaden, men denne typen system vert framleis sett på som eksotiske system, gjerne forbundne med forsking. Den manglande populariteten skuldast mest sannsynleg utviklinga av vellukka alternativ til NLIDB, som grafiske og skjemabaserte grensesnitt, samt dei interne problema NLIDB er befengt med. Døme på nokre kommersielt tilgjengelege system er Intellect (Trinzic), Parlance (BBN), Q&A (Symantec), Natural Language (Natural Language Inc), Loqui (Bim) og English Wizard (Linguistic Technology Corporation).

I dei seinare år har det vore ein nedgang i talet på publiserte artiklar og oppgåver som omhandlar NLIDB. Utviklinga held likevel fram, blant anna dreg ein nytte av framskritt innan NLP og datalingvistikk generelt sett. Diskursteori har til dømes hatt stor innverknad i høve til utviklinga av nye NLIDB. Vidare tek ein nå i bruk arkitekturar som omformar NLIDB til resonneringsagentar og integrerer språk og grafikk for å utnytta fordelane frå begge modalitetar. Det har dukka opp generelle, lingvistiske grensesnitt, det vil seie system som omset språkinnputt til logikkuttrykk. Desse systema kan gjerast til NLIDB ved å leggja til modular som evaluerer dei logiske uttrykka mot ein database.

Mykje har endra seg på dei vel 40 åra NLIDB har eksistert og NLIDB-systema har måtta, og må, tilpassast og endrast for å halda følgje med den databaseteknologiske utviklinga. Som nemnt i innleiinga til dette kapitlet er det ein kalla ein database for 40 år sidan ikkje nødvendigvis det same som ein database av i dag. For det første har databasane vorte stadig større. For det andre vert det å laga ein database i dag rekna som eit forsøk på å modellera ein del av verda. Ein database er strukturert i tråd med ein datamodell og ein eventuell NLIDB er designa for å kunne fungera i høve til gjeldande modell. Ein database er med andre ord meir enn bare ei mengd data, og eit databasesystem er gjerne eit meir komplekst og prinsippfast system i dag enn det var for nokre tiår sidan. Androutsopoulos et al. nemner to store utviklingslinjer innan databaseteknologi som vil komma til å influera NLIDB-feltet i tida framover. Trenden innan relasjonell databaseteknologi går i retning meir komplekse lagringsstrukturar, for slik å leggja til rette for meir avansert datamodellering. Vidare vert objektorienterte databasesystem stadig meir sentrale. Begge desse trendane kan gjera det vanskelegare å utvikla passande NLIDB. Begge reflekterer eit fokus på nye, meir komplekse databaseapplikasjonsområde, som til dømes nettverksbehandling, der brukaren er av typen ekspertbrukar igjen og den direkte tilgangen til databasen gjerne vert utført av eit lag applikasjonsprogramvare [Androutsopoulos -95, s. 4].

1.3 Meir om NLIDB i dag

Androutsopoulos definerer fem hovudkategoriar av NLIDB-tilnærmingar:

Pattern-matching-system
Somme av dei tidlege NLIDB var baserte på mønstersamanlikningsteknikkar (pattern matching) og hadde som oftast ingen eigen parsingmodul. Mange av desse systema består ganske enkelt av reglar som seier at dersom det finst eit visst mønster i innputt, til dømes eit gitt ord følgt av eit anna, så skal ei viss handling utførast. Som oftast består denne handlinga i at ei gitt rad, kolonne eller celle frå den aktuelle databasetabellen vert skriven ut og presentert for brukar. Det finst òg meir avanserte mønstersamanlikningssystem, som SAVVY, som nyttar mønstersamanlikningsteknikkar liknande dei som vert brukte i signalprosessering. Pattern-matching-system er enkle å implementera, men har gjerne ein del kritiske avgrensingar på grunn av dei lingvistiske manglane.

Syntaksbaserte system
LUNAR er eit typisk døme på eit syntaksbasert NLIDB. Denne typen system analyserer, som namnet tilseier, innputt syntaktisk og omgjer så resultatet (parsetreet) til ei form for databasespråkuttrykk ved hjelp av spesielle omformingsreglar. Syntaksbaserte system nyttar med andre ord ein grammatikk som skildrar dei moglege syntaktiske strukturane brukaren sine spørsmål kan bestå av. Reine syntaksbaserte NLIDB, som LUNAR, grenser gjerne mot applikasjonsspesifikke databasesystem med eigne databasespråk laga spesielt med tanke på omforminga frå parsetre til databasespråkuttrykk.

System med semantisk grammatikk
System baserte på semantisk grammatikk, som Ladder, består òg av ein parsingmodul liknande den dei syntaksbaserte systema nyttar. Forskjellen ligg i at den semantiske informasjonen denne typen system nyttar er infiltrert i grammatikken og at kategoriane i denne semantiske grammatikken dermed ikkje korresponderer med syntaktiske konsept. Denne typen system fungerer bra og er spesielt nyttige med omsyn til rask utvikling av parsarar for avgrensa domene, men er vanskelege å tilpassa nye domene nettopp på grunn av den domenespesifikke semantiske informasjonen i grammatikken.

I mange av dei tidlege NLIDB-utgåvene var dei syntaktiske / semantiske reglane baserte på ad hoc-idèar. Dei nyare utgåvene baserer seg i større grad på lingvistiske teoriar og består som oftast av reglar uttrykte i variantar av velkjente formalismar.

System med mellomliggjande språkrepresentasjonar
Dei fleste nåverende NLIDB omset først språkinnputtet til ei logisk form, eit tydingsrepresentasjonsspråk, før det så vert omsett til eit databasespørjingsspråk og evaluert mot databasen. Dei mellomliggjande tydingsrepresentasjonane uttrykker kva spørsmålet brukaren stiller tyder ved hjelp av høgnivå verdskonsept uavhengige av databasestrukturen. Ein av fordelane med denne typen system ligg nettopp i det at dei mellomliggjande representasjonane er uavhengige av databasesystemet. Mange nyare NLIDB-system, som CLE, nyttar fleire av desse mellomliggjande språkrepresentasjonane.

System bestående av fleire underliggjande system
I somme tilfelle trengs det informasjon frå fleire databasar for å kunna svara på brukarinnputt. NLIDB tilrettelagt for slik bruk, som INTELLECT, omset den logiske representasjonen til eit sett av databasespørjingar, kvar enkelt spørjing på databasespråket tilhøyrande det

databasesystemet den aktuelle spørjinga skal evaluerast mot. Etter at delresultata er henta ut av dei aktuelle databasane vert dei samla og smelta saman til eitt utputt. Ask og JANUS er òg, som tidlegare nemnt, system av denne typen.

Det finst sjølvsagt mange system som nyttar andre teknikkar eller som kombinerer to eller fleire av tilnærmingane Androutsopoulos skildrar. Jung og Lee [Jung] nyttar til dømes lingvistisk prosessering basert på leksiko-semantiske mønster, samt fleirnivågrammatikkar til å omsetja innputt til SQL-spørjingar i sitt eksperimentelle hybridsystem. BussTUC-systemet er eit anna døme (sjå avsnitt 1.4 og 1.5 nedanfor).

1.3.1 Krav og brukarstudium

Carbonell gir, ut ifrå sine erfaringar i tilknyting til utviklinga og testinga av systema LanguageCraft og XCALIBUR, ei oversikt over kva for krav som bør oppfyllast for å gjera NLI brukbare, effektive og produktive. Sett frå sluttbrukaren si side er dekningsgrad, kor robuste systema er, responstid og tilbakemelding eller utputt viktige aspekt i så måte. Førstnemnte går på at alle signifikante domenekonsept, det vil seie alle sentrale objekt, relasjonar, tilstandar og handlingar, må vere inkorporerte i grammatikk og kunnskapsbase. Brukarar tilpassar seg som oftast eventuelle syntaktiske manglar, men finn det vanskelegare å tolerera eit system som rett og slett manglar informasjon om konsept eller operasjonar brukarane sjølve ser på som viktige. Eit robust system vil seie eit system som taklar innputtfeil, som stavefeil og diverse grammatiske feil gjort av brukar. Carbonell og hans medarbeidarar fann at så mange som ein tredjedel av alle setningar henta frå deira samling av fleire hundre analyserte brukarinteraksjonsdøme var ugrammatiske, noko som understrekar kor sentralt eit slikt krav om robuste system er. Rask responstid er essensielt, men grunna nye parsingteknikkar[6] og raskare maskinvare eit av dei mindre problematiske krava. Når det gjeld sjølve responsen frå systemet, det vil seie tilbakemeldingane eller utputta, er måten desse vert framstilt for brukaren på viktig. Generering av utputt på brukaren sitt eige språk, samt grafiske utputt dersom dette er aktuelt, er viktig for å kunna leggja til rette for ein god kommunikasjon mellom brukar og system.

Vidare er ellipse- og anaforløysing og tildels tolking av metaspråk viktige aspekt i tilknyting til system tilrettelagde for samanhengande diskurs. Dei førstnemnde tillet brukarane å nytta seg av kortare og mindre omstendelege innputt med referanse til tidlegare diskurs, noko som forenklar og effektiviserer kommunikasjonen. Metaspråk er ytringar om andre, tidlegare ytringar.

Sett frå utviklaren si side er oppdeling, utviklingsverktøy for grammatikkar, sporing og framstilling av prosesseringa, samt systematisk transformasjon viktige stikkord. Jo meir detaljert og forseggjort eit grensesnitt basert på naturleg språk er, jo vanskelegare er det som regel å tilpassa til nye domene. Dermed oppstår det gjerne ei spenning mellom sluttbrukar- og utviklarkrav. Prinsippa og utviklingsreiskapane nemnde ovanfor medverkar til å dempa denne spenninga noko. Prinsippet om oppdeling (decomposition på engelsk), går på syntaks- og semantikkdelen av systemet. Som nemnt tidlegare er det ein fordel, spesielt med omsyn til portabilitet, at desse modulane er avskilde. Problemet er at dette gjerne fører til at systemet vert tregare og analysane mindre nøyaktige. I den seinare tid har det derfor vorte teke i bruk ein ny teknikk som går ut på å prekompilera dei separate syntaktiske og semantiske kunnskapskjeldene til ein integrert grammatikk som så sørgjer for ei meir robust parsing. Spesialisert programvare medverkar til å gjera utviklinga av nye grensesnitt raskare og sørgjer

[6] t.d. on-line parsing, det vil seie at innputt vert parsa medan det vert tasta inn.

for at nye grammatikkar og leksika er velforma og at dei samsvarer. Med omsyn til feilsøking, -retting og kvalitetssjekking av nye grammatikkar bør det vere mogleg å spora og å framstilla parsingprosess og utputt grafisk. Etterkvart som det går raskare og lettare å laga grammatikkar vert det lagt meir vekt på det å tilpassa parsingmodulen til gjeldande databaseapplikasjon. Omsetjinga frå parsingresultat til databasespråk vert som oftast utført av systematiske transformasjonsspråk eller regelbaserte translatørar [Carbonell -86, s. 162-163].

I tillegg til å ta omsyn til dei meir generelle krava nemnde ovanfor er det sjølvsagt viktig at kvart enkelt system vert tilpassa den bruken og typen brukarar det er meint for. Eit system laga for offentleg bruk og dermed mange, meir tilfeldige brukarar møter naturleg nok andre utfordringar enn det eit system berekna for bruk innan eit eller fleire firma med relativt få og frekvente brukarar, gjer.

Omsynet til mennesket sin naturlege oppførsel, inklusive læring og tilpassing, kan vere av tyding i høve til NLIBD-system. Dette gjeld kanskje spesielt i tilknyting til system som vert brukte mykje av kvar enkelt brukar, samt system tilrettelagde for samanhengande diskurs. Karlgren har gjort ei undersøking som går på korleis brukarar tilpassar seg grensesnitt baserte på naturleg språk. Han tek utgangspunkt i menneskeleg oppførsel med omsyn til diskurs menneske imellom og prøver å finna ut om dei same tendensane opptrer i tilknyting til menneske-maskin-dialog. Studiet viser at brukarar har mykje den same oppførselen ovanfor NLI-system som det dei har i høve til ein menneskeleg motpart. Brukarane har til dømes ein tendens til å tileigna seg motparten sitt språk, det vil seie dei prøver å læra språket til systemet ut ifrå tilbakemeldingane det gir [Karlgren-92]. I følgje McKeown har det òg vorte vist[7] at brukarar ofte ønskjer å veta meir om motparten sin. Før dei byrjar å stilla spesifikke spørsmål om sjølve innhaldet i databasen spør brukarane typisk spørsmål for å gjera seg kjende med databasen, som ”*kva veit du?*” og ”*kva type data har du?*”. Vidare meiner McKeown brukarane bør få tilbakemeldingar frå systemet dersom det skulle visa seg at dei har feil oppfatning av korleis systemet fungerer, kva det har informasjon om og kan svara på og ikkje. For å kunna gje meir informasjon enn det det faktisk, det vil seie ordrett, vert spurt om og dermed svara tilstrekkeleg i høve til brukaren sine forventningar, kan det vere nødvendig å nytta brukarmodellering for å halda styr på brukaren sine intensjonar og mål [McKeown -83, s. 9-10]. Kapittel 5, ’Brukarstudium og bruksanalyse’, tek føre seg meir om dette.

1.4 TUC-teknologien

The Understanding Computer, TUC, er ein prototypisk prosessor for naturleg språk skriven i Prolog. Han er laga som ein generell prosessor meint å kunna tilpassast ulike system og bruksområde på ein enkel måte. Prosessoren består, i grove trekk, av ein generell grammatikk for eit subsett av aktuelle språk, ein semantisk kunnskapsdatabase, samt brukargrensesnittmodular. I dag finst prosessoren i to versjonar: Ein for norsk og ein for engelsk. Systemet avgjer kva for eit av språka det til ei kvar tid dreier seg om ved å samanlikna gjeldande innputt med dei respektive ordlistene.

Systemet tilrettelegg kort sagt for å kunna henta ut informasjon frå ein database ved hjelp av naturleg språk. På denne måten skal brukarane, ideelt sett, kunna spørja slik dei normalt ville ha gjort det i høve til ein menneskeleg kommunikasjonspart, utan å måtte ta omsyn til korleis datamaskinen faktisk lagrar og prosesserer informasjonen. Prosessen frå spørsmålsinnputt til

[7] A. Malhotra, 1975: *Design criteria for a knowledge-based English language system for management: an experimental analysis.* MAC TR-146, MIT, Cambridge, Mass.

svarutputt består av leksikalsk, syntaktisk og semantisk analyse, samt pragmatisk resonnering og databasespørjingsprosessering.

TUC starta som eit internt forskingsprosjekt ved NTNU i Trondheim, først og fremst for å vidareføra resultata frå eit tidlegare, nordisk prosjekt kalla HSQL. HSQL står for 'Hjelpesystem for SQL' og er eit spørjesystem for sjukehusadministrasjon. Dette bygg i sin tur direkte på PRAT-89, eit system laga av NTNU-studentar ut ifrå den eldre, engelske versjonen CHAT-80 (sjå avsnitt 1.2 ovanfor). Sistnemnde klassiske språkprosesseringssystem vart laga av Pereira og Warren og er eit geografispørjesystem[8]. CHAT-80 er klassisk i den forstand at det var – og er tildels framleis – eit effektivt og sofistikert system som gjorde sitt for å fremma Prolog som eit konkurransedyktig språk i høve til kunstig intelligens.

1.5 BussTUC

Nok eit universitetsprosjekt, kalla TABOR (Talebaserte grensesnitt og resonnerande system), vart starta i 1996. Målet var å laga eit automatisk ruteorakel for offentleg transport. Systemet var først meint å vere eit talebasert system tilgjengeleg per telefon, men i og med at Internett på denne tida såg ut til å verta eit stadig meir dominerande og vanleg medium vart det bestemt at det òg skulle lagast ein tekstbasert nettvariant.

Ein engelsk prototyp basert på TUC-teknologien, BusTUC, var klar i løpet av bare nokre månadar og vart lagt ut på nett sommaren -96. Den norske delen vart etterkvart lagt til og systemet vart tospråkleg same haust. Busselskapet i Trondheim, nå kalla Team Trafikk, sette opp ein server og tok i bruk systemet våren 1999.

Hovudkomponentane til systemet er:

- Eit parsingsystem bestående av ordbok, leksikalsk prosessor, grammatikk, samt sjølve parsaren.
- Ein kunnskapsdatabase delt inn i ein semantikk- og ein applikasjonsdatabase.
- Ein spørjeprosessor.

Då systemet er bilingvalt har det sjølvsagt dobbelt opp av både ordbok, morfologidel og grammatikk. Systemet har opplysningar om 42 bussruter og 590 stasjonar. Dette tilsvarer rundt 1900 avgangar og 60 000 busstasjonspasseringar per dag. Figur 1.1 nedanfor gir ei oversikt over systemstrukturen.

1.5.1 Grammatikken

Grammatikkformalismen som er brukt, eller rettare kombinasjonen av grammatikkformalismar som er brukte, er – så vidt underteikna veit – unik. Han er basert på ein enkel grammatikk for vanlege, deklarative setningar (statements). Spørsmål og kommandoar vert deriverte ved hjelp av flyttingar. Grammatikken vert kalla ’ConSensiCal Grammar’ som er eit akronym for ’Context Sensitive Categorial Attribute Logical Grammar’ (òg kalla Context Sensitive Compositional Grammar). Formalismen bygg i stor grad på logikk og vert rekna som ei enklare utgåve av Pereira sin ekstraposisjonsgrammatikk (Extraposition Grammar, XG), som i sin tur er ei generalisering av den kanskje meir kjente formalismen

[8] Sjå blant anna: David D. H. Warren og Fernando C. N. Pereira (1982): *An efficient easily adaptable system for interpreting Natural Language Queries*. Computational Linguistics, 8(3-4), s 110-122

'Definite Clause Grammar' (DCG). ConSensiCal-grammatikken er ein slags attributtgrammatikk – derav kontekstsensitiv – som produserer strenger i form av førsteordens hendingskalkulus (first order event calculus).

DCG er ei utviing av kontekstfrie grammatikkar (CFG) og har vist seg å vere nyttige med omsyn til skildring av språk, naturlege så vel som formelle. Ein av fordelane med DCG-ar er den nære tilknytinga til programmeringsspråket Prolog. DCG-reglar kan enkelt uttrykkast og eksekverest meir eller mindre direkte i Prolog. Den største skilnaden frå kontekstfrie grammatikkar ligg i at DCG-formalismen opnar for at ikkje-terminale symbol kan tileignast argument, noko som gjer det mogleg å leggja til detaljert, syntaktisk informasjon i form av samsvarstrekk. Deling av logiske variablar tillet informasjonsoverføringar mellom subfraser. Dømet nedanfor er henta frå TUC-grammatikken og viser eit predikat med mange slike argument:

```
statems0(S,Com,P,P and R) --->
    and1,
    statem(S,Com,Q),
    statems0(S,Com,Q,R).
```

Pereira nytta sin **ekstraposisjonsgrammatikk** i tilknyting til dialogsystemet CHAT-80. Denne grammatikktypen er laga spesielt for å kunna handtera langdistanseavhengigheit og flytting. Formalismen medverkar altså til omforminga av spørsmål og imperative uttrykk til vanlege, deklarative setningar.

Det spesielle med **kategorialgrammatikken** er at fraser vert kombinerte ved hjelp av spesielle operatorar som "/" og "\". Ei kategorialgrammatisk frase `P \ Q` vil seie ei frase `P` som startar med `Q`. Ei kategorialgrammatisk frase `P / Q` er ei frase `P` som manglar Q internt. Nedanfor følgjer nok eit døme henta frå TUC-grammatikken der førstnemnde operator er brukt:

```
ynq(P) --->
    lexv(iv,Rain,Pres,fin),
    [det],
    {testmember(Rain,[rain,snow])},
    !,
    statement(P) \ ([det], w(verb(Rain,Pres,fin))).
```

Dømet representerer ein relativt spesifisert regel for ein særskilt type 'ja-nei-spørsmål', valt først og fremst fordi han demonstrerer bruken av kategorialformalismen på ein enkel og klar måte. Eit spørsmål som "*regner det?*" vert omgjort til ein vanleg, deklarativ setning, "*det regner*". Kategorialgrammatikkar er attraktive med omsyn til semantikk fordi dei held greie på korrelasjonen mellom dei syntaktiske kategoriane og dei semantiske funksjonane (til uttrykka). Nettopp difor vert denne typen grammatikkar brukte innan logisk programmering. For meir stoff om kategorialgrammatikk, sjå til dømes Pereira og Shieber (1987): *Prolog and natural-language analysis*, s. 131-133.

ConSensiCal-grammatikken er meint å vere enklare og meir oversiktleg å lesa og forstå enn ein vanleg DCG-grammatikk. I originalsystemet vert han faktisk omsett til ein vanleg DCG-versjon før han vert brukt i sjølve parsingprosessen. ConSensiCal-grammatikken nyttar argument for å parameterisera konteksten og for å handtera semantiske definisjonar. I motsetnad til ein vanleg kontekstfri grammatikk tillet ConSensiCal-formalismen flyttingar.

1.5.2 Parseren

Parsingmetoden er, som vanleg når ein har med DCG og Prolog å gjera, topp-ned (top down) med tilbakesporing (backtracking). Parsingdelen har bydd på ein del problem, mykje grunna storleiken på og variasjonen forbunde med subsettet av språket som vert brukt. Disambiguering er eit av hovudproblema ein stadig slit med. Ei rekke tiltak er allereie gjennomførte for å løysa flest mogleg av fleirtydigheitane som kan oppstå. Ein semantisk typesjekkar vart integrert i parseren. Vidare nyttar ein seg av ein heuristikk som seier at lengste, moglege frase av ein kategori som er semantisk korrekt er den som, som oftast, er å føretrekke. Vidare har ein sett inn ein del kutt (cut) i grammatikken nettopp for å unngå ein del ambiguitetsrelaterte problem og for å gjera det heile meir effektivt ved å hindra unødvendig tilbakesporing. Meir om ambiguitetsproblem i kapittel 3 og parsing i kapittel 4.

1.5.3 Den semantiske kunnskapsdatabasen

TUC-teknologien er laga slik at det hovudsakleg er den semantiske kunnskapsdatabasen det må gjerast endringar i dersom teknologien skal tilpassast andre system og bruksområde. Ein har altså ein generell grammatikk for syntaksen, medan semantikken til orda er deklarert i tabellar for seg. Databasen er eit semantisk nett bestående av blant anna ISA- og AKO-tabellar og inneheld generell, semantisk informasjon – som kan gjenbrukast – så vel som meir domene- eller applikasjonsspesifikk informasjon. Sett bort ifrå filene innehaldande sjølve bussrutene består semantikkdatabasen av tilsaman ti filer. Nedanfor følgjer nokre døme frå tre av desse.

Dei to første døma er frå fila ’busdat’, ein del som er spesifikk for BussTUC og som fungerer som ei tilpassing til sjølve bussrutedatabasen. Blant anna tilrettelegg informasjonen her for variasjon i høve til stadnamn og liknande. På denne måten opnar altså systemet for at brukarane kan nytta andre namn enn nett dei som er brukte i bussrutedatabasen:

sameplace(blusevol,blussuvold).
synplace(studentby,studenthjem).

Fila ’facts’ inneheld generelle, statiske fakta som vil vere meir eller mindre uavhengige av, og difor gjerne felles for, dei fleste applikasjonar og bruksområde. Denne delen representerer med andre ord ei form for generell verdskunnskap eller ein slags sunn fornuft, om ein vil:

tuc isa program.
ja isa answer.
tt isa company.

Fila kalla ’semantic’ er TUC sin leksikalsk-semantiske kunnskapsdatabase. Denne inneheld informasjon liknande den ein finn i ’facts’. Forskjellen ligg først og fremst i at faktaa her er mindre statiske og universale – dei er i større grad avhengige av gjeldande applikasjon (ein aktivitet, til dømes, er definert som det følgjande i høve til BussTUC-systemet og treng ikkje nødvendigvis ha akkurat denne definisjonen i ein anna kontekst):

activity ako thing.
activity has_a date.
activity has_a time.
activity has_a possibility.

1.5.4 Storleik og kapasitet

Utanom sjølve rutetabellane består systemet av over 35000 linjer programkode. Den norske delen består blant anna av 1300 grammatikkreglar, og det er lagt inn 420 substantiv, 150 verb, 165 adjektiv, 60 preposisjonar, osb. Det semantiske nettverket har rundt 4000 oppslag og ein tabell på godt over 3000 namn som kjem i tillegg til dei offisielle namna på busstopp og stasjonar på grunn av alle måtane ulike stadsnamn kan skrivast, og vert skrivne, på. Ein enkel stavekorreksjon, samt eigne reglar for omsetjing mellom parsingresultat og databasespråk og for generering av utputt på naturleg språk kjem i tillegg. Gjennomsnittleg responstid er vanlegvis mindre enn to sekund, men enkelte komplekse innputt kan kreva opp til ti sekund. Grensa for parsingprosessen er sett til ti sekund. Feilraten for fullstendige, grammatisk korrekte og relevante spørsmål og setningar (i høve til systemgrammatikken) har ligge på rundt 2 %. Tala ovanfor er sjølvsagt omtrentlege tal. Systemet har vorte oppdatert og utbetra heilt frå det først vart teke i bruk, noko som tilseier at stadig meir informasjon har vorte lagt til, spesielt med omsyn til stads-, busstopp- og stasjonsnamn og ulike versjonar av desse.

1.5.5 Frå brukarinnputt til systemutputt

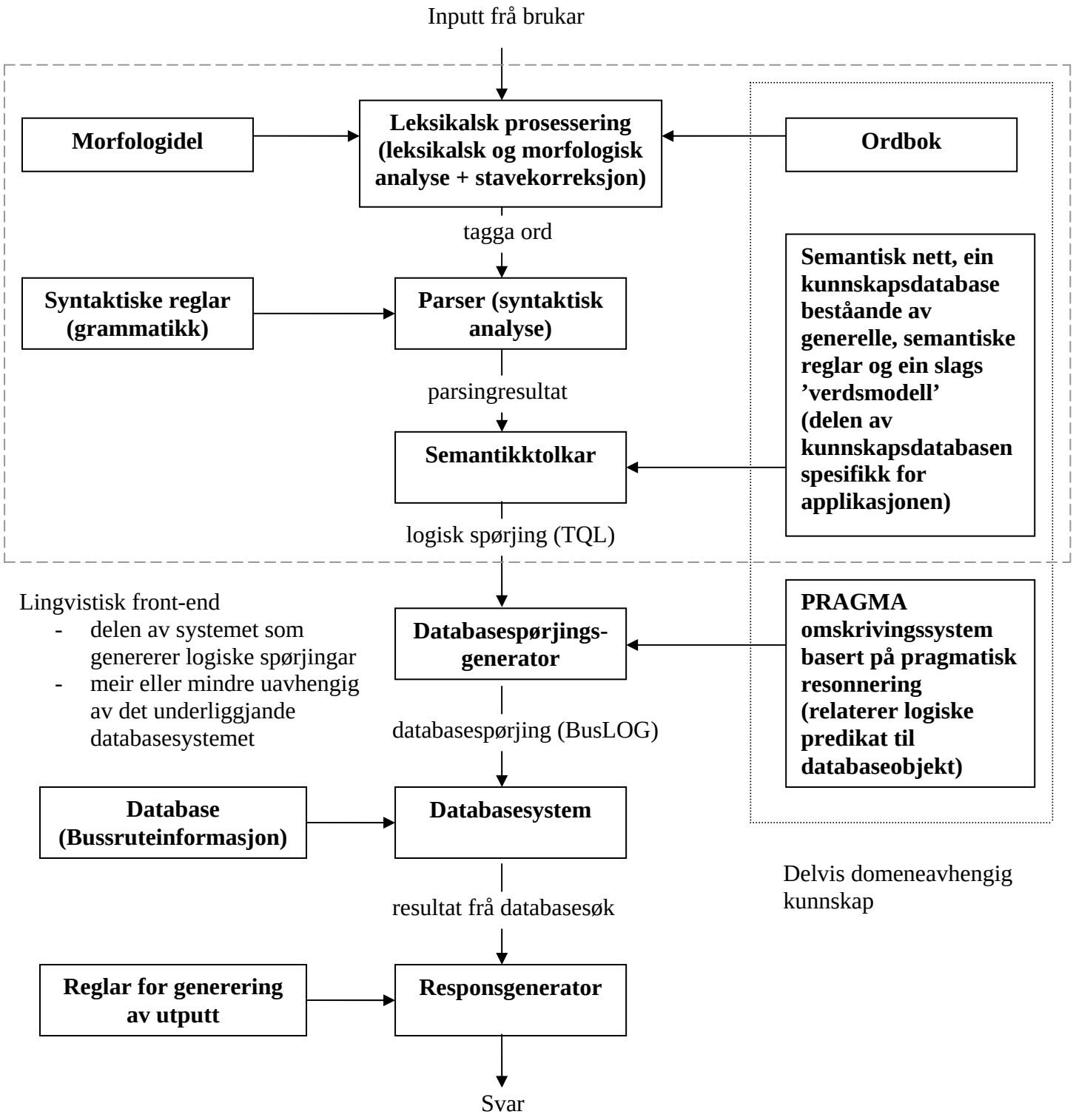
Følgjer ein Androutsopoulos si klassifisering (sjå kapittel 1.3) høyrer BussTUC til gruppa av system med mellomliggjande språkrepresentasjonar. Figur 1.1 nedanfor representerer ein abstraksjon av BussTUC og viser gangen i systemet, frå innputt til utputt.

Eit innputt, som ”*når går neste buss 4?*”, går først gjennom ein leksikalsk analyse. Bokstavane i innputt vert skanna og strengene samansette til ord. Kvart ord vert så analysert med omsyn til morfologiske endingar og det vert køyrd ein stavekontroll. Stavekontrollen består ganske enkelt i at det finst fleire ulike oppslag for somme ord. For ordet *neste* finst blant andre desse ordbokoppslaga:

synword(nestes, neste).
synword(nestse, neste).
synword(nete, neste).

Utputtet frå den leksikalske preprosessoren er ei liste bestående av aktuelle ord, samt kategoriane tilhøyrande kvart av dei:

w(når,[verb(reach,pres,fin),[når],name(når,n,0)])
w(går,[verb(go,pres,fin),[går],name(går,n,0)])
w(neste,[adj(next),name(neste,n,0)])
w(buss,[noun(bus,sin,u,n),name(buss,n,0)])
w(4,[nb(4,num),name(4,n,route),name(4,n,0)])
w(?,[[?],name(?,n,0)])



Figur 1.1 : Modell av BussTUC-strukturen

Modellen er laga med utgangspunkt i Androutsopoulos sin generelle modell for system med mellomliggjande språkrepresentasjonar [Androutsopoulos -95, s.18].

Parsaren analyserer denne omforma innputtstrengen i høve til dei grammatiske reglane og resultatet vert så sendt vidare til semantikktolkaren. Parsestrukturen kan skrivast ut i form av ein trestruktur (kun første del av parsetreet er vist her, resten er å finna i appendiks C):

sentence
onesentence
sentence1
greetings0
[]
question
whenq
when_adverbial
whenx
when
naar
[når]
andwhere0
...

Ved hjelp av semantiske avgrensingar bestemt av innhaldet i den semantiske kunnskapsdatabasen vert utputtet frå parsaren først omgjort til uttrykk på førsteordens logikk (FOL). Eventuelle anaforproblem vert så prøvd løyste før logikkuttrykket vert omsett til ei skolemisert[9] form kalla TQL (TUC Query Language):

[which(A:time)::true and((A isa time and(exists(B:event)::go/4/B and srel/in/time/A/B and event/real/B))and 4 isa bus)and adj/next/4/real]

[which(A)::(4 isa bus,A isa time,adj/next/4/real,go/4/B,srel/in/time/A/B,event/real/B)]

Her vert skråstrek, ”/”, brukt som ein generell, venstreassosiativ operator for generering av predikat (general predicate generating operator).

Før spørjinga kan nyttast til å henta ut aktuell informasjon frå bussrutedatabasen må spørjinga omformast til eit BussLOG-uttrykk, det veil seie eit uttrykk på det aktuelle databasespråket brukt i tilknyting til denne spesifikke, relasjonelle databasen.

Grunna den grådige parsingmetoden (greedy parse) og strategisk plasserte kutt (commit cut) vert kun første, moglege løysing brukt. Resten vert kutta bort. Svaret vert generert ut ifrå gjeldande tidspunkt:

”*Buss 4 passerer Munkegata M5 kl. 1836.*”

[9] Skolemisert vil seia at alle eksistensielle kvantifikatorar er fjerna

2 Behovsanalyse

Eit av formåla med denne hovudfagsoppgåva er å kartleggja kva typar innputt BussTUC-systemet eventuelt har problem med å handtera, og kvifor. I følgje utviklarane er problem knytte til ambiguitet noko av det som er mest framtredande i tilknyting til systemet og difor av særleg interesse. Det er ønskjeleg å finna ut kva for typar ambiguitet det er tale om, kva som ligg til grunn for problema og kva som eventuelt kan gjerast for å takla problema betre. For å få ei oversikt over kva for problem systemet strir med vert først det såkalla korpuset gjennomgått og behandla. I neste omgang vert grammatikken granska i høve til utvalde probleminnputt ved at parsingprosessen vert spora.

Dette kapitlet tek i all hovudsak føre seg gjennomgangen og kategoriseringa av korpusmaterialet og presenterer og kommenterer resultata dette medfører.

2.1 Behandling av korpus

2.1.1 Skildring

Det som vert kalla korpus i denne samanheng er ganske enkelt ei samling ubehandla innputtdata med tilhøyrande utputtdata som er samla inn i løpet av den tida BussTUC har vore i bruk, det vil seie tilgjengeleg via nettet og per SMS. Korpuset er fordelt over fire loggfiler, to med data henta frå nettutgåva, ei med SMS-data, samt ei med data av uidentifisert opphav. Filene med data komne inn via nettet, kalla WEBlog02 og -03, er samla inn over to ulike periodar. Den første av dei består av over 22 000 innputt med tilhøyrande utputt samla inn mellom 14. november og 18. desember 2002. Den andre består av rundt 17 000 inn- og utputt innsamla mellom 9. april og 9. mai 2003. Fil nummer tre, kalla SMSlog, består av litt i overkant av 600 inn- og utputt og vart samla inn mellom den 28. januar og 24. februar 2002. Den uidentifiserte fila inneheld data i form av vel 1700 inn- og utputt samla inn mellom 18. og 20. desember 2002. Innsamlingsdatoane, samt fordelinga av feilmeldingsinnputt (sjå tabell 2.1 nedanfor) kan tyda på at fila er ei fortsetjing på den eldste av WEBlog-filene, men dette er altså ikkje oppgitt.

Totalt består dei fire filene av 41 676 innputt med tilhøyrande systemutputt, dei fleste av desse på norsk (det finst òg nokre få på engelsk). 4121 av desse var innputt systemet, på dette tidspunktet, ikkje kunne svara på og som fekk generert feilmelding. Feilmeldingane er tilbakemeldingar systemet gir når det ikkje er i stand til å parsa innputt og er laga for å seie noko om kva type feil som finst i innputt, det vil seie om innputt er ugrammatisk, ufullstendig eller uforståeleg på noko vis, vel og merke i høve til systemgrammatikken. Systemet nyttar hovudsakleg fire ulike feilmeldingar: *Uforståelig ved *, Uforståelige ord: [ord], Vennligst bruk en fullstendig setning, Setningen er for vanskelig*. Det finst òg feilmeldingar på engelsk. Tabell 2.1 har ei oversikt over antalet feilmeldingar per loggfil.

I tillegg finst det ei mengd andre tilbakemeldingar, somme av dei meir spesifikke og informative enn dei føregåande feilmeldingane: ”*Jeg vet ikke*”, ”*Du må oppgi et sted i slike spørsmål*”, ”*Jeg har bare ruter for busser i Trondheim*”, ”*Stedet _ _ _ _ er flertydig. Vennligst bruk et mer presist navn*”, ”*Jeg klarte ikke å finne noen forbindelse i tide*”, osb. Desse tilbakemeldingane opptrer i tilknyting til innputt som kan parsast, men som er mangelfulle – det vil seie innputt som ikkje har oppgitt til dømes stad eller buss, irrelevante – det vil seie spør om ting systemet ikkje er meint å kunna svara på, eller liknande. Sjå appendiks B for ei oversikt over denne typen tilbakemeldingar.

Korpus / Feilmelding	WEB-02	SMS	Ukjent opphav	WEB-03	Samla
Uforståelige ord	998 (47,5%)	19	70	782 (44,9%)	1869
*Uforståelig ved **	724 (34,5%)	54	61	707 (40,6%)	1546
Vennligst bruk en fullstendig setning	290 (13,8%)	3	60	229 (13,1%)	582
Setningen er for vanskelig	72 (3,4%)	4	7	15 (0,9%)	98
Incomprehensible words	10 (0,5%)	1	-	7 (0,4%)	18
*Incomprehensible at **	6 (0,03%)	-	-	2 (0,01%)	8
Totalt antal feilmeldingsinnputt (*)	2100 (9,4%)	81 (13,1%)	198 (11,4%)	1742 (10,2%)	4121 (9,9%)
Totalt antal innputt (m/tilhøyrande utputt)	22 239	616	1740	17 081	41676

Tabell 2.1: Oversikt over feilmeldingsinnputt

(*) Prosentane i denne rada er prosent feilmeldingsinnputt i høve til totalt antal innputt. Dei resterande prosenttala er berekna ut ifrå det totale antalet feilmeldingsinnputt. *Uforståelig ord*-meldingane utgjer m.a.o. 47,5% av det totale antalet feilmeldingar i WEBlog-02, osb.

2.1.2 Tolking av og kommentar til tabell 2.1

Talet på feilmeldingar, samla sett, ser ikkje ut til å ha vorte lågare frå WEBlog-02 vart samla inn til WEBlog-03 vart samla inn. Talet på feilmeldingsinnputt i høve til det totale talet på innputt er tvert imot noko høgare for WEBlog-03. Om dette skuldast kvaliteten på systemet eller kvaliteten på innputt er uvisst, i og med at det er tale om ulike innputt i dei respektive korpusa. Mengda *Uforståelige ord*- og *Setningen er for vanskelig*-meldingar har gått ned med nokre få prosent for WEBlog-03 sett i høve til den samla summen av feilmeldingar. Feilmeldingar av typen *Uforståelig ved ** finst det fleire av i WEBlog-03. For dei resterande feilmeldingstypane er skilnadane små. Det er sannsynleg at mengda av uforståelege ord har gått ned på grunn av at det har vorte lagt til fleire ord i systemet sidan WEBlog-02 vart samla inn. Det er òg mogleg at systemet nå taklar fleire setningstypar og / eller fleire meir komplekse setningar enn det gjorde tidlegare. Sjå avsnittet ’Tolking av og kommentar til Tabell 2.2’ nedanfor for fleire kommentarar med omsyn til systemendring og -forbetring.

Det er interessant å merka seg at fordelinga av feilmeldingstypane er så og seie den same for begge WEBlog-korpus og for fila av ukjent opphav, medan tala for SMS-korpuset fordeler seg noko annleis. Tala er for små til at ein kan dra noko gyldig slutning ut frå dette, men kan kanskje seiast å tyda på at SMS-innputt og problema knytte til dei fortonar seg noko annleis enn nettinnputt og dei respektive problema som følgjer med desse.

2.1.3 Kategorisering

Målet var, som nemnt ovanfor, å kartleggja kva systemet har problem med og kva det eventuelt ikkje taklar i det heile. Dermed var det først og fremst dei av innputta som hadde gitt feilmelding, frå nå av kalla feilmeldingsinnputt eller bare innputt, som var av interesse. Systemet markerer somme feil ved hjelp av ein asterisk, '*', i det gjengjevne innputtet, slik:

*- - - Uforståelig ved * - - -*
*når og hvilke * busser går fra gløshaugen til moholt etter kl 18 . 0 .*

Denne typen feilmelding er på mange måtar ei av dei minst spesifikke typane feilmeldingar og nest etter *Uforståelige ord*-feilmeldinga den det finst flest av (sjå tabell 2.1). Eg ser på denne typen feilmeldingsinnputt, samt innputt med feilmeldingar av typen *Setningen er for vanskelig*, som dei potensielt viktigaste i høve til behovsanalysen. Dette fordi det, etter nærare ettersyn, ser ut til å vere størst sjans for å finna relevante, gjerne grammatikk- og ambiguitetsrelaterte, problemområde i tilknyting til desse innputta.

For å skilja ut førstnemnde vart det nytta eit lite AWK-program, laga for å plukka ut alle linjer innehaldande asterisk. På denne måten får ein altså ei fil bestående av kun aktuelle feilmeldingar, med respektive innputtsetningar (gjengjevne), som vist i dømet ovanfor. Innputt med generert *Setningen er for vanskelig*-melding, som det fanst langt færre av, vart plukka ut manuelt ved å søke etter feilmeldinga, for så å kopiera og lima.

Den resulterande fila med feilmeldingsinnputt vart så gjennomgått manuelt og feilmeldingsinnputta kategoriserte. Dei fleste av feilmeldingsinnputta vart kategoriserte i høve til feilmeldinga, det vil seie ofte i høve til kvar asterisken var plassert i det gjengjevne innputtet, og skjønn. Eg opererte med seks ulike, sjølvlaga kategoriar kalla 'Fleire i ein', 'Feilstavingar', 'Forkortingar / dialekt', 'Irrelevant'. 'Manglande ord / teikn' og 'Ugrammatisk / Ufullstendig'. I tillegg vart ein del av feilmeldingsinnputta plukka ut til vidare testing. Den første kategorien omfamnar dei av innputta som ikkje har gått / går gjennom fordi dei består av fleire enn ein setning, eller fordi innputtsetninga har vore for kompleks (systemet er laga for å ta imot ei setning om gongen og tek heller ikkje omsyn til kommateikn og liknande).

Døme, 'Fleire i ein':
jeg skal til byen på mandagen hvilken buss må jeg ta for og være i byen klokken 15?
når jeg skal være 1800 ved torvet i dag når er det 36 går da?

Systemet er utstyrt med ein enkel stavekontroll, men ein del feilstavingar vert likevel problematiske.

Døme på feilmeldingsinnputt som tilhøyrer feilstavingskategorien:
går går neste buss fra tillerbyen til sentrum ?
*når går bussen far gløshaygen * til tev ?*

Mange av dei innputta som hamna i kategorien 'Forkortingar / dialekt' er typiske SMS-innputt skrivne på det ein gjerne kan kalla SMS-språk. Systemet tek høgde for å kunna takla både slike og ein god del av innputta skrivne på trøndersk, noko det har vorte stadig betre til etterkvart som fleire ord og uttrykk av denne typen har vorte og vert lagt til. Mange av dei tidlege feilmeldingsinnputta går med andre ord greitt gjennom nå.

Døme på innputt med forkortingar og / eller dialekt:
kossn buss ska æ ta fra Jakobsli for å kom te stranda?
nr går neste buss fra gløs til sentr?

Innputt det i utgangspunktet ikkje er aktuelt at systemet skal kunna svara på, det vil seie som har lite eller ingenting med bussruter å gjera, hamnar i kategorien 'Irrelevant'. (Sjå kapittel 5, *Brukarstudie og bruksanalyse*, for ein vidare diskusjon og kommentar med omsyn til irrelevante innputt).

Døme på irrelevante innputt:
Hvor bor Nils Arne Eggen?
Forstår du deg på kvinner?

'Manglande ord / teikn' inneheld innputt som består av ord eller teikn som ikkje er lagt inn i systemet. Igjen er det slik at systemet i mange tilfelle har endra seg sidan desse dataa vart samla inn. Om dei aktuelle orda og teikna som framleis ikkje finst i systemet burde vore lagt inn eller ikkje er eit anna spørsmål og er ikkje teke omsyn til her.

Døme på innputt innehaldande ord eller teikn som ikkje ligg i systemet:
*når går neste 5 busser til loholtbakken 11 fra sildråpevn 35 * b?*
når kommer neste 7 reppe / vikåsen til sentrum ?[10]

Feilmeldingsinnputt som vert kategoriserte som ugrammatiske eller ufullstendige er innputt som er ugrammatiske eller ufullstendige i høve til systemgrammatikken (normalt er desse òg ugrammatiske i høve til vanlege grammatikkstandardar). Feilmeldingsinnputta som hamnar i denne kategorien er som oftast innputt som har fått *Setningen er for vanskelig*-feilmelding eller *Uforståelig ved**-melding med asterisken plassert til slutt, det vil seie etter det gjengjevne innputtet.

Døme på ugrammatiske og / eller ufullstendige setningsinnputt:
når må jeg ta fra skovgård til sentrum til kl 1 på søndag?
når går fra jakobsli i mårra t semtrum?

Ein del av innputta, særskilt dei det var vanskeleg å kategorisera eller som kunne virka interessante på andre måtar, vart testa igjen, enten i nettversjonen av systemet eller i den lagra, lokale versjonen. Mange av innputta går greitt gjennom nå, på grunn av at systemet har vorte forbetra. Andre er framleis problematiske. Det er nettopp desse det er ønskjeleg å sjå nøyare på og som derfor er plukka ut til vidare testing.

2.1.4 Feilmarginar

Det er viktig å påpeika, nok ein gong, at det hovuddelen av behovsanalysen, det vil seie korpusbehandlinga, fører til av resultat ikkje nødvendigvis representerer tilstanden til systemet slik det er i dag. Som nemnt vert systemet stadig utbetra og endra.

Poenget med behandlinga er først og fremst å finna fram til typar av innputt som framleis byr på problem. Det er interessant å sjå kva type feil det finst flest av og, ikkje minst, om dei ulike feiltypane er noko ein som utviklar kan gjera noko med. Ved å gå gjennom korpus på denne måten får ein ei oversikt over kva innputt som typisk finst i tilknyting til eit slikt system som

[10] I dette dømet er det skråstreken som ikkje er lagt inn i systemet og som difor skapar problem.

det her er tale om – noko som er interessant i seg sjølv – samt at ein får ein peikepinn i retning kva som, generelt sett, er problematisk i høve til system av denne typen. Vidare kan det vere interessant å samanlikna resultata frå dei første korpusinnsamlingane med nyare resultat og på den måten kanskje finna ut noko om korleis systemet fungerer i dag i høve til tidlegare.

Resultata er naturleg nok prega av metoden som er brukt. I og med at kategoriseringa vert gjort manuelt og i stor grad er basert på skjønn er mange av tala nedanfor ikkje meir enn indikasjonar å rekne. Ofte finst det til dømes fleire feilkategoriar i eitt og same innputt. Dermed må ein prøva å bedømma kva for feil som har størst relevans, det vil seie kva feil som mest sannsynleg er hovudårsaka til at innputtet ikkje går gjennom. I somme tilfelle er det enkelt å rangera feila: Dersom eit innputt er irrelevant speler det til dømes inga rolle om det òg inneheld stavefeil eller liknande. I andre tilfelle må ein ganske enkelt gjera eit val etter beste evne.

Ofte finst det fleire av same eller svært like feilinnputt i korpusa. Brukarar prøver typisk fleire gonger med same eller liknande innputt dersom dei ikkje får generert eit svar frå systemet på første forsøk. Dersom det finst fleire identiske innputt etter einannan (skrivne av same brukar) vert som ein hovudregel bare eitt eksemplar lagra og kategorisert, medan innputt som skiljar seg frå einannan – om enn bare med eitt enkelt ord eller ei litt anna ordstilling – vert tekne med.

2.1.5 Resultat

Tabell 2.2 viser ei oversikt over kor mange feilmeldingsinnputt som vart tekne ut, kor mange av desse som vart valde ut til vidare testing og kor mange som hamna i kvar av dei seks resterande kategoriane.

Korpus / Kategori	WEB-02	SMS	Ukjent opphav	Samla
Fleire i ein	43	21	1	65
Feilstavingar	63	4	5	72
Forkortingar / dialekt	12	4	-	16
Irrelevant	208	3	11	239
Manglande ord / teikn	214	16	11	241
Ugrammatisk / Ufullstendig	128	18	14	160
Feilmeldingsinnputt tekne ut til vidare testing	75	15	20	93
Antal feilmeldingsinnputt tekne ut til saman (*)	743	81	62	886

Tabell 2.2: Kategorisering av feilmeldingsinnputt

(*) Innputta som vart valde ut var kun innputt som hadde fått feilmeldinga Uforståelig ved * eller Setningen er for vanskelig.

2.1.6 Tolking av og kommentar til tabell 2.2

Igjen er fordelinga for nettinnputt-tala noko annleis enn for SMS-innputt-tala, men kategoriane 'Manglande ord / teikn' og 'Ugrammatisk / Ufullstendig' skiljer seg ut som to av dei større kategoriane i alle korpusa.

Manglande ord og teikn
I WEBlog-02 finst det ti ulike innputt (av 214) innehaldande skråstrek. Det finst òg eit innputt med kolon. Åtte av ti av dei innputta som mottok feilmelding på grunn av stadnamn eller namn på busshaldeplassar som ikkje låg i systemet går greitt gjennom nå. Gateadresser, som *35b*, har vore problematiske tidlegare, men ser nå ut til å skapa færre problem. Seks av dei 18 døma funne i 'Manglande ord / teikn'-kategorien henta frå WEBlog-02 går gjennom nå. Buss- / rutenummer bestående av ein kombinasjon av siffer og bokstav, som *6a* eller *3b*, skiljer seg òg ut som vanskelege. Mange av døma henta frå WEBlog-02 går gjennom nå, slik som dei er, eller så fungerer det som oftast dersom bokstaven vert fjerna.

Ugrammatisk / Ufullstendig
Ugrammatiske og / eller ufullstendige innputt finst det òg mange av i alle korpusa. Dei aller fleste av innputta plasserte i denne kategorien inneheld feil gjort av brukar og er dermed vanskelege, frå systemutviklaren si side, å gjera noko med. Mange av innputta inneheld typisk eitt ord for mykje, gjerne i form av eit oppatteke ord, eller dei manglar ord: *når går må jeg ta bussen fra dokkparken når jeg skal være i munkegt kl. 20.0?* og *hvilke er de tre neste busser går fra ila til sentrum?* eksemplifiserer dette. Ein del av innputta er ugrammatiske på grunn av feil ordstilling: *når går buss fra østre berg til byen klokken etter halv åtte?* og *når de to går neste bussene til gløs?* er typiske døme. Felles for døma ovanfor er at brukar truleg gjer desse feila ubevisst.

Andre innputt er meir kommandoliknande og brukaren er truleg bevisst på korleis ho skriv. Med kommandoliknande meiner eg korte, ugrammatiske (i høve til standardgrammatikkar) innputt som går i retning av databasespråkkommandoar eller menyvalliknande strenger. Eit par typiske døme er *gløshaugen ila no neste.* og *5 går fra?*
Mange brukarar utelet gjerne det dei verkar til å sjå på som sjølvsagte ord , som til dømes *buss* i *når går fra strindheim til fossegrenda?* og *fra buran som er på lerkendal 745.*

Systemet verkar til å kunna handtera fleire av denne typen innputt nå enn tidlegare, men så lenge det i stor grad er feil frå brukaren si side det her er tale om vil denne typen problem aldri kunna få noka endeleg og altomfattande løysing. Spørsmålet i tilknyting til denne typen innputt er kor grensa skal gå. Grammatikken systemet består av er i utgangspunktet konstruert for å kunna takla fullstendige, grammatiske setningar. Dermed byr det på problem å skulla handtera alle moglege former for avvik frå dette. Skal ein prøva å "oppdra" brukarane og få dei til å skriva grammatisk, eller bør ein i større grad jenka seg etter og tilpassa systemet til måten folk faktisk uttrykker seg på? Kva skal ein godta og ikkje? Hovudpoenget med NLIDB-system er at dei skal vere enkle og effektive å bruka. Korte, kommandoliknande innputt – gjerne ugrammatiske i høve til standardgrammatikkar – er ofte det brukarar ser på som mest effektivt og enklast og burde derfor kunna handterast av systemet.

Irrelevant
Irrelevante innputt finst det langt fleire døme på i WEB-korpus enn i SMS-korpus. Mest sannsynleg har dette å gjera med det faktum at nettversjonen av systemet er gratis å bruka, medan SMS-ar koster pengar. Innputt som ikkje har noko med bussruter å gjera vert i stor grad ignorert frå utviklaren si side, men BussTUC er faktisk i stand til å svara på enkelte

innputt av denne typen. Innputt som stadig dukker opp, spørsmål som har med sjølve systemet å gjera, eller spørsmål som vert sett på som signifikante eller interessante på andre måtar, som *hva er meningen med livet?* og *hvem har laget deg?*, vert gjerne tekne omsyn til. Systemet har vorte rusta til å kunna handtera stadig fleire av desse (meir eller mindre) irrelevante spørsmåla sidan det først vart teke i bruk. Fleire kommentarar og ein vidare diskusjon rundt dette og resultata elles finst i kapittel 5, 'Brukarstudie og bruksanalyse'.

2.2 Behandling av dei utvalde probleminnputta

I utgangspunktet talde probleminnputta som vart plukka ut til vidare testing rundt 60 stykk. Desse er tekne med i appendiks B. Neste steg var å organisera innputta og fordela dei i hovudgrupper. Eit par av dei meir prominente gruppene probleminnputt vart så studerte nøyare. Kvart enkelt probleminnputt frå desse vart grundig testa ved at dei aktuelle delane av parsingprosessen vart spora og gjennomgått. Vidare vart det konstruert nye, liknande strenger som òg vart testa på same måte, for slik å kunna fastslå meir nøyaktig kva problem ein hadde med å gjera. Deretter var tida inne for å prøva å endra på aktuelle grammatikkreglar, samt – i somme tilfelle – ordbokoppslag og morfologireglar, for så å testa igjen. Som nemnt i innleiinga til dette kapitlet er ambiguitet eit av problemområda det er særleg ønskjeleg å få sett nøyare på. Dei fleste av dei problemutputta som vart arbeidd vidare med er problematiske nettopp grunna ulike former for ambiguitet. Skildringa av utvalde problem og prosessen vidare er å finna i kapitlet om ambiguitet, kapittel 3.

3 Ambiguitet

Dette kapitlet tek føre seg ambiguitet, først på eit generelt, teoretisk plan, og deretter på eit meir spesifikt, det vil seie i høve til NLIDB-system og BussTUC-systemet spesielt. Som det vert påpeika i kapittel 2 utgjer ambiguitet og problema denne medfører store deler av grunnlaget for denne hovudfagsoppgåva. Ulike typar ambiguitet og vanlege løysingsmetodar er difor skildra i noko meir detalj. Den meir spesifikke delen skildrar kva som er gjort for å prøva å betra BussTUC-systemet si evne til å få bukt med enkelte utvalde ambiguitetsrelaterte problem.

Ambiguitet, eller fleirtydigheit, er ein essensiell og på mange måtar verdifull bestanddel i naturleg språk.[11] Ambiguitet illustrerer kor komplekse og rike menneskelege språk er. Samstundes er ambiguitet eit av dei mest sentrale og vanskelegaste problema ein har å stri med innan språkprosessering. Ein kallar noko – eit ord, ei frase, ei setning eller liknande – fleirtydig dersom det kan tolkast på meir enn ein måte, det vil seie har fleire enn ei mogleg tyding. Jurafsky og Martin definerer ei frase som fleirtydig dersom ho har fleire, alternative, lingvistiske strukturar [Jurafsky -00, s. 358]

3.1 Ulike typar ambiguitet

Det finst svært mange typar ambiguitet. I litteraturen presenterer ulike forfattarar emnet på noko ulike vis, nyttar tildels ulike namn på somme typar ambiguitet og deler ambiguitet inn og organiserer det heile på noko ulik måte. Hovudtypane og –skilja er likevel stort sett dei same. Normalt skiljer ein mellom ulike typar ambiguitet ut ifrå kva språkleg nivå det er tale om, det vil seie om ambiguiteten opptrer på til dømes ord- eller setningsnivå. To hovudtypar ambiguitet er følgjeleg det som gjerne vert kalla leksikalsk og syntaktisk ambiguitet.[12]

3.1.1 Leksikalsk ambiguitet

I tilknyting til førstnemnde har ein, som namnet tilseier, å gjera med enkeltord med ulike tydingar. I setninga *Kari mista posten* utgjer ordet *post* eit døme på leksikalsk fleirtydigheit. Utan noko meir informasjon om kontekst og situasjonen rundt hendinga kan ein ganske enkelt ikkje veta om det Kari mista var ei bunke med brev eller ei stilling. ”*hvilken linje på nattbussen går * fra granåsen*”, henta frå BussTUC-korpuset, kan tyda på å vere fleirtydig på grunn av ordet *går*.

Somme gjer eit finare skilje mellom det ein kan kalla ordtydingsambiguitet og kategorialambiguitet. I tilknyting til språkprosessering og parsing definerer ein ord som ordtydingsfleirtydige dersom dei har eitt terminalsymbol, men kan referera til ulike konsept. Begge døma ovanfor kan kanskje kategoriserast som denne typen ambiguitet. Kategorialambiguitet opptrer i tilknyting til ord med meir enn eitt terminalsymbol. Eit døme er ordet *lærer* som kan vere både eit nomen og eit verb. Leksem er òg eit viktig omgrep i dette høvet, og nært relatert til kategorialambiguitet. Eit leksem er ein slags abstraksjon som viser til innhaldssida til ord. *Å fiska*, *fiskar*, *fiska* er instansar av same leksem – det dreier seg om same handling og dermed same tyding – men er like fullt ulike grammatiske ord. Forma *fiska* kan vere tvetydig med omsyn til kva grammatiske ord det er tale om, infinitivs- eller

[11] Naturleg språk vil her seie språk slik menneske bruker det seg imellom – skriftleg eller munnleg – i motsetting til det ein kan kalla kunstige språk som logikk- og programmeringsspråk.

[12] Leksikalsk ambiguitet vert òg kalla tydings- (sense) og ordtydingsambiguitet (word sense ambiguity). Andre ord for syntaktisk ambiguitet er strukturell, grammatisk eller konstruksjonell (constructional) ambiguitet.

preteritumsform. Ordet *fiskar*, derimot, kan seiast å vere fleirtydig på leksemnivå, i og med at det kan stå både for fleirtal av *fisk*, for ein som fiskar fisk, samt for handlinga *å fiska*.

Omgrepa polysemi og homonymi er sameleis omgrep ein gjerne treff på i høve til leksikalsk ambiguitet. Polysemi vil seie at ein har fleire ord på same form med relatert innhald. *is* kan til dømes tyda både *iskrem* og *frose vatn*. Homonymi har òg å gjera med ord på same from, men med ulikt innhald. Eit døme er ordet *høy* som på bokmål kan tyda både *høg*, som i *lang*, og *tørka gras*. I tillegg kan omgrep som metafor, metonym, homofon, homograf, paradoks, med fleire relaterast til denne forma for ambiguitet (sjå *Semantics* i [Saeed -00, s. 63-71] for definisjonar av desse).

3.1.2 Syntaktisk ambiguitet

Syntaktisk ambiguitet er knytt til syntaktiske konstituentar, det vil seie setningar eller fraser, med ulike tydingar. Her er det altså grammatikken som gir opphav til ulike tolkingar. Setninga *Me reiser til Sverige og Danmark eller Finland* eksemplifiserer ei form for syntaktisk fleirtydigheit: Tyder setninga at ein reiser til Sverige og i tillegg til enten Danmark eller Finland, eller reiser ein enten til både Sverige og Danmark eller bare til Finland?

Baustad skildrar i si avhandling fire av dei, i følgje han, vanlegaste typane syntaktisk ambiguitet, nemleg analytisk ambiguitet, tilknytings-, konjunksjons- og lukelokaliseringsambiguitet (gap finding ambiguity). Analytisk disambiguering dreier seg om å bestemma konstituentane sin natur. Baustad nemner mange ulike døme på denne typen ambiguitet. Somme døme er kun relevante i høve til engelsk, men enkelte gjeld òg for norsk. I setninga *eg vil ha boksen på bordet* er til dømes preposisjonsfrasa problematisk – det er uklart om setninga tyder noko slikt som *eg vil ha tak i boksen som står på bordet* eller om ho tyder noko i retning *eg vil ha boksen plassert på bordet*

Tilknytingsproblem oppstår i situasjonar der ein syntaktisk konstituent, ein preposisjon til dømes, kan knytast til fleire noder.[13] Eit klassisk døme på dette er setninga *han såg mannen med kikkerten* der det er uklart om preposisjonsfrasa *med kikkerten* modifiserer nominalfrasa *mannen* eller verbfrasa.

Reisedømet ovanfor inneheld dei til tider ambiguitetsframkallande konjunksjonane *og* og *eller*, og er eit døme på den forma for syntaktisk ambiguitet som vert kalla konjunksjonsambiguitet. Konjunksjonsambiguitet kan seiast å vere ei form for grupperingsambiguitet.

Lukelokaliseringsambiguitet har med flyttingar å gjera. Flyttingar er transformasjonsoperasjonar involverte i danninga av passive setningar, relativsetningar og spørsmål. Ambiguiteten oppstår dersom ein flytta konstituent kan ha høyrt heime meir enn ein stad i utgangssetninga. Setningane *Per såg <u>boka</u> som Erik gav jenta _* og *Per såg <u>jenta</u> som Erik gav _ boka* inneheld relativsetningar som eksemplifiserer fenomenet. Det einaste som skiljer dei to setningane er plasseringa av dei to substantivfrasene *boka* og *jenta*, men dei føretrekte lukene (representerte med "_") er på ulike stadar. Begge relativsetningar har to moglege luker, ei i direkte objekt-posisjon og ei i indirekte objekt-posisjon [Baustad-92, kap. 2.2].

[13] Sett i høve til parsingtreet til aktuelle setning.

3.1.3 Andre skilje

I tillegg til leksikalsk og syntaktisk ambiguitet skildrar Baustad det han kallar anafor- og kvantifikatorspennambiguitet (quantificator scope ambiguity). I ei setning som *Eli og Janne diskuterte breva ho hadde fått frå Olav* er det uklart kven, Eli eller Janne, anaforen *ho* refererer til – ein har med andre ord ein anaforambiguitet. Kvantifikatorspennambiguitet har, som namnet tilseier, med kvantifikatorar å gjera. Setninga *alle professorane skreiv ei bok* er eit typisk døme på denne forma for fleirtydigheit. Kvantifikatoren *alle* gir opphav til to ulike tolkingsspenn – om ein kan kalla det det – eit smalt og eit vidt: Setninga kan tolkast som at alle professorane arbeidde saman om ei og same bok, eller som at kvar og ein av professorane skreiv kvar si bok [op.cit. s. 19-23].

Referanseambiguitet er nok eit omgrep som dukkar opp i ambiguitetslitteraturen. Denne typen kan kanskje seiast å omfatta anaforambiguitet. I følgje Inman har ein med referanseambiguitet å gjera når ei nomenfrase kan referera til meir enn eitt objekt [Inman -97, kap. 4.4]. Eit døme er *Ibsen fekk henne alltid til å gråte* der *Ibsen* kan referera til personen Ibsen, eller til verk skrivne av Ibsen.

Eit anna vanleg hovudskilje går mellom semantisk og pragmatisk ambiguitet. Eit uttrykk er semantisk fleirtydig når uttrykket kan tilleggjast ulike tydingar, som i dei fleste av døma ovanfor. Både leksikalsk og syntaktisk ambiguitet, med fleire, kan med andre ord reknast som ulike typar semantisk ambiguitet. Ei formulering er pragmatisk fleirtydig når ho i ein og same situasjon kan nyttast til å utføra fleire enn ein type talehandling. Ei ytring som *det er kaldt her* kan til dømes vere ei konstatering eller ei rein informasjonsformidling, men det kan òg vere ein implikasjon, ei oppfordring – om enn indirekte – om å lukke vindauga eller liknande.

Vidare vert det ofte gjort eit skilje mellom lokal og global ambiguitet. Lokal ambiguitet vil seie at ein del av ei setning – eit ord, ei frase eller ein større setningskonstituent – har fleire enn ei tolking, medan setninga som heilskap kun har ei tolking. I setninga *ein fiskar fiskar fiskar* er kvart av orda *fiskar*, isolert sett, fleirtydige. Heile setninga har derimot kun ei tyding[14]. Dersom ei setning kan tolkast på meir enn ein måte kallar ein det global ambiguitet. Setninga *eg kjenner fleire vakre kvinner enn Ingunn* eksemplifiserer fenomenet. (Kjenner eg eit større antal vakre kvinner enn det Ingunn gjer, eller kjenner eg Ingunn – som er vakker – og fleire andre som òg er vakre?).

3.2 Disambiguering

Ein vart tidleg klar over kor sentralt løysing av ambiguitetsproblem står i høve til å oppnå gode resultat innan NLP. Ambiguitet vart lenge sett på som eit svært komplisert og nærast uløyseleg problem, og det var først for nokre tiår sidan at forskarane byrja å faktisk ta tak i problemet og komma fram til ulike løysingsmetodar.

Ein har etterkvart innsett at det å få ein maskin til å prosessera språk i heile sin fylde er bortimot umogleg. Språket, generelt sett, vert for komplekst og kunnskapsflaskehalsen ganske enkelt for smal.[15] Innanfor snevrare område derimot, fungerer NLP relativt greitt. Syntaktisk,

[14] Setninga kan vel og merke analyserast på i alle fall to måtar, men tydinga er den same.
[15] Gale et al (1992) definerer, i følgje Lyse [Lyse -03, s. 5] 'the knowledge acquisition bottleneck' som problemet med å finna tilgjengelege og anvendbare kunnskapsressurser for å utføre automatisk orddisambiguering.

semantisk og pragmatisk analyse løyser ein god del av potensielt problematiske former for ambiguitet. Ei form for det ein kan kalla sunn fornuft, samt ein modell av (den aktuelle delen av) verda er andre viktige bestanddelar eit skikkeleg NLP-system bør ha.

Baustad nemner assosiasjonar mellom ord, syntaktiske karakteristikkar ved ord, seleksjonsrestriksjonar, kontekst og inferens som nøkkelomgrep i høve til det å skulle løysa ambiguitetsproblemet. Han skildrar tre hovudtilnærmingar til disambiguering, nemleg ordboksbaserte metodar, korpusbaserte metodar og kvalitative metodar [Baustad -92, kap. 3].

3.2.1 Ordboksbaserte metodar

Førstnemnde metode nyttar maskinleselege ordbøker og det av informasjon som finst i desse for å disambiguera. Denne tilnærminga går typisk ut på å finna semantiske assosiasjonar mellom orda i setningane og ein dreg gjerne nytte av syntaktiske teikn, seleksjonsrestriksjonar og kontekst. Lesk er ein av pionerane innan leksikalsk disambiguering med hjelp av ordbøker[16]. Lesk nyttar tydingsdefinisjonane i ordboka og ei algoritme som leitar etter ord i desse som overlappar ord i definisjonane til nærståande ord i innputteksten. Eit av dei meir kjente døma hans er disambigueringa av det engelske uttrykket for furukongle, *pine cone*. Ordboka Lesk opererer med inneheld fire tydingar av *pine* og tre for *cone*. Orda *evergreen* og *tree* finst i ein av tydingsdefinisjonane for kvart av orda. Ut ifrå dette kan eit program komma fram til kva for tyding som er mest aktuell når *pine* og *cone* opptrer saman.

Becker står for ei alternativ, men liknande, algoritme som i staden for å nytta tydingsdefinisjonane nyttar sitata i ordboka, det vil seie døma på bruk av aktuelle ord.

Fordelen med denne metoden er at ein nyttar informasjon som allereie finst. Sånn sett vert arbeidsmengda langt mindre enn ho gjerne er i tilknyting til andre metodar, som korpusbaserte metodar. Spørsmålet er bare om ordbøkene, som jo ikkje er laga med tanke på denne typen arbeid, faktisk inneheld nok nøyaktig og relevant semantisk informasjon i høve til det disambigueringsprosessen krev, samt om informasjonen er på ei slik form at den er tilgjengeleg for ei datamaskin.

Det finst døme på kombinasjonar av ulike tilnærmingar til disambigueringsproblemet: Veronis og Ide kombinerer til dømes bruken av digitale ordbøker med nevrale nettverk for å finna relasjonar mellom ord, noko som ofte resulterer i meir robuste resultat enn det metodar kun baserte på ordboksinformasjon gjer [op.cit., kap. 3.3.1].

3.2.2 Korpusbaserte metodar

Korpusbaserte metodar er probabilistiske metodar og nyttar, som namnet tilseier, store korpus som eksempeldatabasar. Korpusa er normalt markerte med syntaktisk og gjerne semantisk informasjon som i sin tur vert brukt i disambiueringsprosessen. Innputt vert ganske enkelt samanlikna med ord og setningar i korpus, med omsyn til kontekst, struktur, eller liknande. Den mest plausible tolkinga av innputt vert berekna ut ifrå dette. Baustad nemner Uramoto og Nagao som begge presenterer kvar sin eksempelbaserte disambigueringsmetode i tilknyting til dei respektive systema the Sentence Analyzer og the Dependency Analyzer.

[16] Sjå Lesk M. (1986): *Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to tell a Pine Cone from an Ice Cream Cone*, Proceedings of SIGDOC, s. 24-26. Ordboka nytta av Lesk heiter Oxford Advanced Learner's Dictionary of Current English.

Eksempeldatabasen Uramoto nyttar består av disambiguerte setningar, samt informasjon om synonymi-, taksonymi- og avhengigheitsrelasjonar, og fungerer som ein kunnskapsbase brukt til å kalkulera preferanseverdiar.

Gale, Church og Yarowsky nyttar parallel tekst i staden for handannoterte korpus som eksempelbase. Ord som er fleirtydige på eitt språk er gjerne ikkje det på eit anna. Ta til dømes det engelske ordet *sentence* som innan jusen kan tolkast i ein forstand og innan lingvistikken i ein ganske annan. I ei norsk eller fransk omsetjing ville ein derimot finna to ulike ord for dei to tolkingane, nemleg *dom* og *setning* på norsk og *peine* og *phrase* på fransk. Ved å samla eksempelmateriale på denne måten er det altså mogleg å løysa ein god del av den leksikalske ambiguiteten. Nå skal det nemnast at enkelte ord kan vere fleirtydige, på same måte, på fleire språk, noko som vanskeleggjer bruken av denne metoden. Likevel oppnår Gale, Church og Yarowsky gode resultat [op.cit., kap. 3.3.3].

Ein av fordelane med denne metoden, forutan gode resultat, er at disambiguerte setningar kan leggjast til i korpus og dermed medverka til å auka og forbetra kunnskapsbasen ytterlegare.[17] Ulempa med metodar som dette er at dei krev svært store, manuelt tagga korpus for å kunne fungera tilfredsstillande. Det vert arbeidd med metodar for å automatisera korpustagginga. Lyse skildrar i si hovudfagsoppgåve ein metode som ekstraherer finitte, tydingstagga korpus automatisk basert på omsetjingskorrespondansar i eit parallellkorpus [Lyse -03].

3.2.3 Kvalitative metodar

I motsetnad til dei to føregåande er kvalitative metodar ikkje baserte på ytre kjelder som ordbøker og korpus. Den nødvendige kunnskapen vert bygd opp manuelt. Hirst løyser leksikalsk ambiguitet ved hjelp av ein metode basert på det som vert kalla markøroverføring (marker passing). Markøroverføring er ein generell mekanisme for å finna semantiske assosiasjonar og har vore mykje brukt innan kunstig intelligens. Systemet til Hirst består blant anna av ein kunnskapsbase oppbygd av rammer[18]. Dei såkalla rammene er organiserte i eit isa- og instanshierarki. Markøroverføringa går ut på at taggar, eller markørar, vert overførte mellom og innan rammene i tråd med spesielle reglar. Kvart ord i innputt vert representerte som semantiske objekt, såkalla polaroid-ord (Hirst kalla systemet sitt Polaroid Words), og vert brukte som utgangspunkt for markøroverføringa. Andre mekanismar, som seleksjonsrestriksjonar og syntaktiske teikn, medverkar òg i disambigueringsprosessen.

McCord skildrar eit logikkbasert maskinomsetjingssystem, LMT (Logic-based Machine Translation), der kjeldeanalysen er basert på det som kanskje kan kallast lukegrammatikk på norsk (*slot grammar* på engelsk). LMT løyser leksikalsk ambiguitet blant anna ved hjelp av semantiske typar. Kvar konstituent vert tildelt ein semantisk type som er med på å bestemma kva for semantiske typar argumenta deira godtek og ikkje. Pereira sin NLIDB CHAT-80 løyser leksikalsk ambiguitet på liknande måte. Her vert substantivfrasane tildelte semantiske typar som i sin tur må stemma overeins med dei semantiske mønstra verba er assosierte med.

Mange av tilnærmingane baserte på kvalitative metodar har fokusert på svært avgrensa domene og har vist seg å vere svært vanskelege å skalera opp. Ein av fordelane med denne løysingsmetoden er at kunnskapsdatabasen er oppbygd manuelt og at utviklaren dermed får bestemma kva kunnskap som skal takast med og ikkje, samt på kva form kunnskapen skal representerast.

[17] Fenomenet vert på engelsk kalla knowledge bootstrapping.
[18] Ei ramme representerer ein entitet eller eit konsept i verda, i form av ein heil klasse (t.d. alle bilar) eller ein instans (t.d. ein særskilt bil), og består av ei samling attributt (slots) med tilhøyrande verdiar (fillers).

3.3 Ambiguitet i høve til BussTUC

I høve til språkprosessering og parsing vert definisjonen på ambiguitet noko meir spesifikk: Ein konstituent er fleirtydig dersom han gir opphav til fleire enn ein måte å parsa han på. Det kan ofte vere skilnad på kva me menneske ser på som fleirtydig og kva som faktisk er fleirtydig for ein maskin og eit system. Eit system laga for eit gitt, avgrensa domene finn gjerne kun ein måte å parsa ein for oss menneske fleirtydig streng på. Ordet *gryta* er til dømes eintydig i BussTUC-systemet i kraft av kun å vere ført opp som eit stadnamn. Som oftast finn ein òg døme der situasjonen er den motsette, nemleg at systemet slit med ting me menneske vanlegvis ikkje ein gong tenkjer over at kan vere problematiske. Alt beror på kva informasjon systemet har tilgjengeleg og ikkje. Eit døme som kanskje kan illustrera dette, henta frå feilmeldingsdelen av korpuset tilhøyrande BussTUC-systemet, er *"går 4 klokka 20?"* I motsetnad til det liknande *"går 4 klokken 20?"* gir dette innputtet feilmelding. Systemet får, overraskande nok, problem med innputtet på grunn av ordet *klokka*, eit ord som elles verkar til å fungera greit og på lik linje med *klokken*. Det finst nemleg ingen informasjon om kjønn i ordboka til systemet og i følgje informasjonen i morfologidelen kan a-endinga difor tolkast som ei mogleg fleirtalsending (som det ville vore om det var eit inkjekjønnsord i bestemt form fleirtal, som *barna*, det dreidde seg om). Dermed får ein ei overgenerering, og i dette tilfellet ei feiling på grunn av av ein cut i grammatikken.[19]

Vidare kan ein trekka eit skilje mellom faktisk ambiguitet og ambiguitet som skuldast mangel på kontekst-, situasjon- og / eller verdskunnskap. Det eg kalla reisedømet ovanfor (konjunksjonsambiguitetsdømet) må kunna reknast som genuint fleirtydig – det er fleirtydig same kva for samanheng det måtte bli sett inn i. Andre setningar er kun fleirtydige tekne ut av kontekst. Genuint fleirtydige innputt, som òg er fleirtydige i høve til det aktuelle systemet, er det lite ein kan gjera med kva sjølve prosesseringa vedkjem.

Ambiguitet er problematisk i høve til NLIDB-system, og NLP-system generelt, av fleire grunnar. Dersom innputtstrengen som heilskap har fleire enn ein mogleg parse er det sjølvsagt problematisk, for ikkje å seie umogleg, for systemet å velja kva for tolking det skal gå vidare med utan vidare rettleiing frå brukar. Dersom ein eller fleire av konstituentane innputtstrengen er oppbygd av er fleirtydig(e) vil systemet kanskje komma fram til rett løysing, men det kan ta tid – noko som er uakseptabelt særskilt med omsyn til online-applikasjonar. Inman kallar det ein kombinatorisk eksplosjon: La oss seie at ein innputtstreng består av ti ord og at kvart av orda kan tolkast på tre ulike måtar. Då vil talet på moglege kombinasjonar av aktuelle streng komma opp i godt over 50 000, noko det tek tid å gå gjennom [Inman -97, kap. 2]. Problemet i høve til NLIDB-system som BussTUC er at alle innputt må kunna omsetjast til eintydige uttrykk – databasespråkuttrykk – for i det heile å kunna gå vidare og henta ut informasjon frå databasen. Disambiguering er med andre ord ein føresetnad for at systemet i det heile skal fungera.

3.3.1 Disambiguering i BussTUC

Som nemnt i delkapittel 1.5.2 i bakgrunnskapitlet er BussTUC i stand til å løysa, eller rett og slett unngå, ein god del av potensielt problematiske ambiguitetstilfelle. Den syntaktiske, semantiske og pragmatiske analysen står for ein god del av disambigueringa. Vidare nyttar systemet ein type grammatikk liknande den Pereira nytta i CHAT-80, basert på flytting og semantiske typar. Disambigueringsmetoden er med andre ord kvalitativ.

[19] Årsaka til problemet hjalp bivegleiar Tore Amble til med å finna. Problemet verkar til å vere løyst i den nyaste utgåva av BussTUC.

ConSensiCal-grammatikken har sine klare fordelar, men òg sine ulemper. Ulempene har først og fremst med parsingmetoden, samt storleiken på domenet, å gjera. Meir om dette følgjer i kapitlet om parsingdelen, kapittel 4. Ein semantisk typesjekkar integrert i parsaren medverkar til å verta kvitt semantisk ukorrekte parsingar tidleg i prosessen og har difor ein sentral plass når det gjeld diambiguering. BussTUC-systemet er vidare konstruert for å følgja ein heuristikk som seier at lengste moglege frase for ein gitt kategori som er semantisk korrekt som oftast er den som er å føretrekke. I tillegg er det satt inn ein god del såkalla kutt (cut) på strategiske stadar i grammatikken. Dette for å hindra parsaren i å tilbakespora, noko som er meint å resultera i sparing av tid og kraft og dermed ei effektivisering av diambiguerings- og sjølve parsingprosessen, men som i enkelte tilfelle òg kan føra til at eventuelle gale val som vert gjort i løpet av prosessen ikkje vert retta opp (som i *klokka*-dømet nemnt ovanfor).

Til tross for desse disambigueringsfremmande tiltaka finst det framleis fleirtydige omgrep og uttrykk som skapar problem. Nokre av desse er skildra nedanfor.

3.3.2 Talambiguitet

Innputt innehaldande tal utgjer eit av dei ambiguitetsrelaterte problema som opptrer hyppigast i høve til BussTUC, og er dermed noko av det denne oppgåva tek føre seg i noko meir detalj. I BussTUC-verda kan tal vere alt frå klokkeslett og datoar, til rute- og bussnummer, del av gateadresser og tal på bussar eller avgangar, med meir. I tillegg opererer ulike brukarar med ulike måtar å skriva tala på. Mange skriv tal i form av siffer, andre nyttar talord. Nokre skriv datoar og klokkeslett fullt ut, altså som *14.02.02* og *09.45*, medan andre nyttar andre format, som *1402* og *945*. Dei fleste brukarar tek med *kl* eller *klokka/-en* framfor klokkeslettet, men somme gjer det ikkje. Systemet verkar til å handtera dei ulike formata rimeleg bra, men i somme tilfelle kjem det ikkje tydeleg nok fram kva det faktisk er tale om.

Kombinasjonsproblem

Innputtet ”*når går neste buss etter 1001 bussen fra lundåsen til sentrum?*” er eitt døme der tal, i dette tilfellet mest sannsynleg i form av eit klokkeslett, skapar problem. Her kan klokkeslettet sjå ut til å vere brukt som ei slags spesifisering, som eit namn om ein vil, på ein særskild buss. Dette er systemet ikkje laga for å kunna handtera. Talet vert prøvd parsa som både rutenummer og klokkeslett, men ingen av tolkingane går gjennom. Når klokkeslettet, feilaktig, vert tolka som eit rutenummer feilar det ganske enkelt fordi det ikkje finst noko rutenummer 1001. Når systemet faktisk tolkar klokkeslettet som eit klokkeslett vert setninga, grunna det påfølgjande ”*bussen*” feil syntaktisk sett, det vil seie i høve til systemgrammatikken. Spørsmålet ”*når går bussen fra sentrum nr 15 til st olavs hospital etter kl 1600?*” er eit anna døme der det er gateadressa, ”*nr 15*”, som skapar problem. Hovudproblemet ser ut til å vere at slike adresser kan uttrykkast på mange ulike vis. Dei fleste skriv gatenummeret etter gatenamnet, men det finst òg brukarar som skriv nummerert framfor namnet, som i ”*når går bussen fra lerkendal gård til 36 thoning owesens gate klokken 14.30?*”. Somme nyttar ”*nr*”, som i ”*sentrum nr 15*”, andre gjer det ikkje. I tillegg kan det finnast ulike variantar av sjølve gateadressene. Enkelte av dei kan til dømes innehalda bokstavar i tillegg til tal, som i *34a*. Det vert med andre ord mykje som må leggjast inn i systemdatabasen og mange ulike kombinasjonar som må leggjast til rette for dersom ein skal ta høgde for alt av variasjon. Noko av det same finn ein i tilknyting til spesifisering av bussar / ruter. Her finn ein òg nok av døme på at ulike brukarar nyttar ulike uttrykk og uttrykk av ulik kompleksitet. Eitt og same spørsmål kan variera frå ”*når går buss rute nr 21 fra sentrum?*” til ”*når går 21 fra sentrum?*” Som aller oftast gjenkjenner systemet kvart og eit av orda i desse og liknande innputt; Det er igjen alle kombinasjonsmoglegheitene, det vil seie ei form for syntaktisk ambiguitet, som utgjer problemet. Nok ein gong dukkar spørsmålet om kor grensa

skal gå opp. Skal ein halda fram med å utrusta systemet slik at stadig fleire kombinasjonar går gjennom, eller bør ein i større grad heller gå inn for å læra opp brukarane og spesifisera kva som bør brukast og ikkje? Meir om dette i kapittel 5.

Andre former for feiltolking
Eit anna problem som går att fleire gonger i korpusinnputta har nok ein gong med klokkeslett å gjera. Det kan sjå ut til at systemet tolkar alle tider som ønska framkomsttid. Spørsmålet ”*når kommer buss nr 44 til Adressa når den går fra sentrum kl 1515?*” får generert eit svar innehaldande informasjon om ein buss som er *framme* ved Adressavisa klokka 15.15. Eit fabrikkert innputt vert brukt for å testa fenomenet: *”Når kommer en til lade når en tar bussen fra sentrum kl 14?*” gir svaret ”*Holdeplassen nærmest Lade er Lade allé 80. Buss 3 passerer Munkegata M5 klokken 1335 og kommer til Lade allé 80 klokken 1351*.”[20] Når spørsmålet så vert gitt inn på nytt med klokkeslettet frå det genererte svaret, 1335, i staden for det originale klokkeslettet, 14, vert neste systemutputt sjåande slik ut: *”Holdeplassen nærmest Lade er Lade allé 80. Buss 3 passerer Munkegata M5 klokken 1315 og kommer til Lade allé 80 klokken 1331*.”

”*når går de 3 neste 48 fra kattem?*” og ”*når går de 3 første 5 bussene fra østre berg etter kl 18 til sentrum?*” er andre, typiske døme på brukarinnputt systemet opphavleg hadde problem med og som fekk generert feilmelding. Testsetningar med kun eitt siffer involvert, sett bort frå eventuelle klokkeslett, som *”når går de neste 48 fra Kattem?*”, *”når går de første 5 bussene fra østre berg etter kl 18 til sentrum?*” og ”*når går de 3 første bussene fra østre berg etter kl 18 til sentrum?*” viste seg, i motsetnad til dei føregåande døma, å gå gjennom og få generert eit svar. Problemet var at systemet konsekvent tolka dei respektive tala, med unntak av klokkeslett, som antal (det vil seie antal bussar). Dermed fekk ein altså ikkje nødvendigvis svar på kva tid ein bestemt buss, det vil seie ei bestemt rute, vert køyrd. I tilfelle der ein kan velja mellom fleire, ulike ruter når ein skal frå ein stad til ein annan burde det vere mogleg å få informasjon om kvar enkelt av dei ved hjelp av namnet, det vil seie talet, på den aktuelle bussen / ruta.

3.3.3 Løysingstilnærmingar
Talkonstituentar kan altså, som det føregåande viser, vere leksikalsk fleirtydige og difor opphav til feile tolkingar som kan føra til generering av meir eller mindre gale svar eller feilmelding. Ofte fører den leksikalske ambiguiteten til syntaktisk ambiguitet som i sin tur kan føra til generering av gale svar eller feilmelding. Problema skildra ovanfor er, til tross for at dei alle har med tal å gjera, såpass ulike at det neppe finst noko felles løysing for dei. Kvart einskild problem vert prøvd løyst for seg, ved ganske enkelt å gå inn og leggja til eller endra på aktuelle grammatikkreglar. I somme tilfelle vart det òg nødvendig å endra eller leggja til ordliste- og morfologioppslag.

Sistnemnde problemtype, der buss-/rutenummer feilaktig vert tolka som antal, er den som har vorte arbeidd mest med. I det høvet vart det først gjort ei undersøking, ut ifrå korpus, med omsyn til bruk av bindestrek. Enkelte, inklusive meg sjølv, finn det meir naturleg å skriva til dømes ’5-bussen’, med bindestrek, enn ’5 bussen’ når det er tale om å spesifisera kva buss ein meiner. Skulle dette visa seg å vere ei allmenn oppfatning blant brukarane av Bussoraklet ville ei innføring av bindestrek som eit eige tydingsberande teikn kunna vere ei potensiell løysing på dette spesifikke feiltolkingsproblemet. Ved hjelp av bindestrek ville det til dømes vere mogleg å tileigna innputt som *Når går de neste X-bussene fra ... til ...?* og *Når går de*

[20] Munkegata er ein av sentrumsstasjonane

neste X bussene fra ... til ...? ulike tydingar. Det opphavlege systemet tek ikkje omsyn til det ein kan kalla ikkje-alfabetiske teikn eller spesialteikn – som bindestrek, komma og skråstrek – i det heile. Slike teikn er karakteriserte som støy og er lista som såkalla støyord (noisew) i systemordlista. På denne måten vert dei ganske enkelt oversett under parsing.

Bindestrekundersøkinga gjekk ut på å henta ut alle innputt innehaldande bindestrek frå kvar av korpusfilene, for deretter å sortera desse i høve til kva tyding bindestrekteiknet ser ut til å vere brukt i i dei ulike tilfella. På den måten får ein ei oversikt over kva i alle fall eit visst utval brukarar faktisk nyttar bindestrek til. Som tabell 3.1 viser ser teiknet ut til å verta brukt hovudsakleg i tyding ’frå - til’, som i "*moholt - sentrum?*", i tyding 'og' eller 'til' i tilknyting til klokkeslett, som i "*går det noen buss mellom klokken 03- 05?*", samt i tilknyting til namn, som *Amo-senteret*. Innputt innehaldande bindestrek brukt i tilknyting til bussnamn, som i ”*Når går neste 9-buss fra voll studentby?*”, finst det kun eit par av per korpusfil (desse er rekna med under kategorien ’Andre’ i tabell 3.1).

	WEB02	**Ukjent**	**WEB03**	**SMS**	**Tilsaman**
’frå - til’	63	2	43	0	108
Særnamn	65	13	15	3	96
Klokkeslett	68	5	20	0	93
Andre	13	6	18	0	37
Irrelevant*	(2)	(1)	(2)	0	(5)
Samla	209	26	96	3	334

Tabell 3.1: Bindestrekbruk i korpus
* Medrekna i ’Andre’

Å leggja til rette for bruk av bindestrek på lik linje med andre ord fører rett nok til at innputt av sistnemnde type går gjennom og får generert eit passande svar, men det fører òg til ein del ekstraarbeid, særskilt i høve til eit par av dei resterande – og største – kategoriane for bindestrekbruk. Ved å leggja til ’-’ som eit synonym både for ’og’ og ’til’ i ordboka, som oppslag på forma *synword(-,og).* og *synword(-,til).*, går dei aller fleste innputta tilhøyrande kategorien ’Klokkeslett’ gjennom. Innputta tilhøyrande ’frå - til’-kategorien gjekk òg gjennom, men gav derimot gale svar. Systemet byter rett og slett om på stadane det er tale om, det vil seie det tek ønska avreisestad som ønska framkomststad, og omvendt. For ’Særnamn’-kategorien sin del må det leggjast til nye oppslag i ordboka for kvart samansette namn der brukarar nyttar, eller kan tenkjast å nytta, bindestrek. For å kunna bruka eit uttrykk som ’KBS-senteret’ må ein til dømes ha lagt til eit oppslag på forma ”*cmpl(kbs,[-,senteret], kbs_senteret).*” i applikasjonsdatabasen (*busdat.pl*).

Med omsyn til at bindestrek vert brukt så lite som det verkar til å verta i tilknyting til bussnamn ser vinninga ut til å gå opp i spinninga. Ein kjem mest truleg like langt, og unngår utan tvil ein del ekstra problem og meirarbeid, ved å behalda systemet slik det er i høve til ikkje-alfabetiske teikn. Det verkar til å vere ein tendens i skriftleg norsk at bindestrekteiknet vert teke i bruk stadig meir. Dette kunne tenkjast å vere eit argument for å ta omsyn til teiknet i tilknyting til system som BussTUC, men bruken verkar til å vere av ein såpass uregelmessig og varierande karakter at det ville ikkje ha ført til anna enn ei komplisering av det heile. Det

viser seg dessutan å fungera overraskande bra å ganske enkelt ignorera teiknet. Dei aller fleste av innputta grupperte ovanfor er innputt som går gjennom og får generert eit tilfredsstillande svar i det originale systemet. Det er òg eit viktig poeng at systemet uansett må kunna handtera innputt der det i utgangspunktet ville vore naturleg å nytta bindestrek, men der brukaren likevel har valt å utelata teiknet. Vidare tyder undersøkinga på at bindestrek er svært lite brukt i innputt sendt som SMS. Mest truleg skuldast dette at bindestrek er eit forholdsvis vanskeleg tilgjengeleg og dermed arbeidsamt teikn å skriva på dei fleste mobilar. I staden for å endra på systemet med omsyn til ikkje-alfabetiske teikn burde ein kanskje heller informera brukarane og oppfordra dei til å unngå teikna (med unntak av punktum og spørjeteikn, sjølvsagt).

Bindestrekprosjektet skulle likevel visa seg å medverka til ei løysing av enkelte antal- og bussnamnproblem. Ein av reglane som vart lagt til for å kunna nytta bindestrek fungerer nemleg òg utan at det vert teke omsyn til bindestrekteiknet og kan difor nyttast til å løysa ein del av problema. Regelen er ein nominalfraseregel av typen'noun_complex' og sørgjer for at det let seg gjera å handtera innputt innehaldande meir enn eitt siffer, som "*når går de 3 neste 4 bussene?*" og "*når går de neste 3 4 bussene?*":

```
noun_complex(N:Bus,N isa Bus,plu) --->
    plausibleno(Bus,N),
    %bindestrek,
    w(noun(Bus,_,_,n)),
    {subtype0(Bus,vehicle)}.
```

Parsetreet for (deler av) nominalfrasa i innputtet "*når går de 3 neste 3 bussene?*" ser slik ut:

```
…
 np_head
  determiner0
   theplu0
    all0
     []
    theplu
     [de]
    !
    accept
     []
  number
   only0
    []
   num
    nb(3,num)
    !
   {}
 preadjs0
  adj(next)
  preadjs0
   []
 {}
 noun_complex
  plausibleno
   nr0
    []
```

```
  num
    nb(3,num)
    !
  {}
bindestrek
  []
noun(bus,plu,def,n)
…
```

I løpet av løysingsprosessen kom det fram at innputt som var identiske, kun med unntak av tal, som *"når går de neste 3 bussene fra lade?*" (fleirtal) og *"når går den neste 3 bussen fra lade?*" (eintal), vart parsa ulikt. Grunnen til at *de* og *den* vart parsa på ulike måtar viste seg å vere at orda *de* og *det* opphavleg var sette opp som synonym, truleg på grunn av at det er ein vanleg stavefeil å skriva *de* i staden for *det*. Grunna oppslaget "*thereit ---> [det],!,accept.*" førte dette til at predikatet for midlertidig subjekt, 'thereit', vart køyrd, i staden for predikatet for fleirtalsartikkel, 'theplu', når innputt innehaldande *de* vert parsa. Vidare viste det seg at innputt innehaldande *de* vart parsa annleis enn same innputt med *dei*, sjølv etter at det tidlegare nemnde synonymoppslaget i ordboka vart kommentert ut. Kun *dei* var oppført som ein fleirtalsartikkel, medan *de* vart assosiert med eintalsartikkelen, 'the', grunna grammatikkoppslaget "*the ---> [de].*" Oppslaget vart kommentert ut, noko som gav resultat i høve til parsingstrukturen. Endringane hadde derimot ingen effekt i høve til sjølve resultatet av parsinga og generering av svarutputt. Dermed bør kanskje stavekontrollen prioriterast framfor parsingstruktur. Ei mogleg løysing kunne vore å plassera stavekontrollen i ei eiga fil, avskild frå resten av grammatikkreglane, slik at han ikkje får noko innverknad på parsingstrukturen.

Eit testsett bestående av 24 innputt innehaldande ulike kombinasjonar av antal og buss- / rutenummer vart konstruert (sjå appendiks B) og brukt for å testa systemet og dei nye reglane, først med bindestrek, deretter utan. I det opphavelege systemet går alle testinnputta, med unntak av par 2 og 3, gjennom. Innputt med bindestrek vert handterte på same måte som dei same innputta utan bindestrek og alle tal vert konsekvent tolka som antal (bussar). Når bindestrekteiknet i den endra utgåva av systemet vert behandla som eit tydingsberande teikn, fungerer alle testinnputta med unntak av det første (1a), som etter parsing gir ein 'existence-error'-melding. I og med at denne typen innputt manglar både avreise- og framkomststad, og dermed reiseretning, hamnar dei strengt tatt utanfor det systemet er meint å kunna handtera og burde difor fått ei passande feilmelding (som "*Du må oppgi et sted i slike spørsmål*"). Innputt med bindestrek på forma "*Når går de neste X-bussene fra ... (til ...)?*", vert tolka annleis enn same innputt utan bindestreken, "*Når går de neste X bussene fra ... til ...?*"; I førstnemnde er det tale om ein spesifikk buss / ei spesifikk rute, medan i den siste vert talet tolka som antal (bussar). Innputt på same form som testpar 2 og 3 får derimot same svar, vilkårleg om dei vert skrivne med bindestrek eller ikkje. For å oppsummera fungerer den nye utgåva av systemet, utan å ta omsyn til bindestrekteiknet, på lik linje med den opphavelege utgåva, bortsett frå at sistnemnde type innputt – det vil seie innputta innehaldande to siffer – òg går gjennom og får generert passande svar. Innputt på same form som testinnputt 4a og 4b er, og vert, fleirtydige så lenge bindestrekteiknet ikkje vert teke omsyn til. Det viktigaste vert i denne situasjonen at systemet er konsekvent: Dersom det finst fleire tolkingsmoglegheiter må det enten alltid tolka siffera som antal, eller alltid som buss-/rutenamn. I og med at antaltydinga er den som er mest utbreidd er det logisk å behalda denne som eit slags utgangspunkt eller ei slags standardtolking om ein vil. Det er òg viktig å framheva at ein har høve til å få ut ønska informasjon ved å skriva om spørsmålet: Både ved å nytta "*buss 4*" eller "*bussene nr 4*" i

staden for "*4 bussene*" og ved å ta med antal kan ein få svar på kva tid ein særskilt buss går. "*når går de neste buss 4 / bussene nr 4?*" og "*når går de 2 neste 4 bussene?*" gir med andre ord svar på kva tid rute 4 vert køyrd.

Neste kapittel tek føre seg parsingdelen av systemet og skildrar arbeidet med å få ein kartparsar tilpassa BussTUC-grammatikken. Formålet er å testa om denne måten å parsa på kan føra til betre resultat, spesielt med omsyn til fleirtydige inputt. Ein kartparsar tek omsyn til og lagrar alle moglege parsar, både for delkonstituentar og for heile innputtstrengen. For genuint fleirtydige innputt, som det sistnemnde dømet ovanfor, kunne ei mogleg løysing vore å presentera dei ulike tydingane for brukaren, for så å la brukaren velja kva for tyding systemet skal gå vidare med. Ei slik løysing krev nettopp at det vert laga og teke vare på eventuelle alternative parsar.

Kapittel 5 presenterer andre, alternative løysingstilnærmingar som blant anna dekker nokre av dei problema som er skildra og prøvd løyste her. Parafrasing er ein av desse tilnærmingane som kanskje kan vere til hjelp òg i høve til ambiguitet. Ved at systemet gjentar alle brukarinnputt på ein eintydig måte får brukaren, for det første, innsikt i korleis systemet tolkar innputta og fungerer. For det andre kan brukaren så velja om han vil gå vidare med gjeldande innputt, eller om han vil gå tilbake og omformulera spørsmålet sitt.

4 Parsingdelen

Den opphavlege tilbakesporingsparsaren tilhøyrande BussTUC-systemet er effektiv og rask, men det finst – som det føregåande viser – framleis ambiguitetsproblem det står att å løysa. Kutta som vert brukt for å effektivisera parsinga kan i somme tilfelle føra til at feil parse vert valt, i og med at kun første moglege alternativ vert teke omsyn til. Ein kartparsar finn derimot alle moglege parsar og kan i mange tilfelle oppnå betre resultat enn ein vanleg topp-ned-parsar, spesielt med omsyn til disambiguering. Det er difor interessant å testa om dette kan vere tilfellet i høve til BussTUC-systemet. Storleiken og kompleksiteten til grammatikken gjer det til ei heller vanskeleg, for ikkje å seie umogleg, oppgåve å skulla bedømma kva utfall dette vil få på førehand. Einaste måten å finna det ut på er å implementera ein kartparsar, prøva han ut, for så å samanlikna resultatet ein då får med resultatet den originale parsarmodulen gir.

Første delen av kapitlet tek føre seg tilbakesporings- og kartparsing i noko meir detalj, med vekt på kva for fordelar og ulemper som er forbunde med kvar av desse metodane. Deretter følgjer nokre avsnitt om val og samanlikning av metodar, før så arbeidet med å tilpassa kartparsaren til Buss-TUC-systemet vert skildra. Til slutt vert resultata diskuterte og andre metodar vurderte.

4.1 Tilbakesporingsparsing

Bakdelen med å nytta den innebygde prologparsemekanismen er at denne kun arbeider topp-ned, djubde-først med tilbakesporing. Ein vanleg parsar basert på tilbakesporing kan vere ineffektiv. Ofte utfører denne typen parsar mykje overflødig arbeid i form av å gjenoppdaga fraser som allereie er funne og å utforska hypotesar som allereie har feila, på ny. Parsaren utfører eit såkalla blindt søk. Topp-ned-parsarar med tilbakesporing har i verste fall eksponensiell tidkompleksitet, men denne teoretiske grensa har ikkje nødvendigvis noko særleg relevans i høve til parsing av setningar på naturleg språk. Pereira meiner at i praksis er storleiken på grammatikken av mykje større tyding i høve til yteevne enn det lengda på innputt er. Ein vanleg innputtstreng er vanlegvis relativt kort, medan det å leggja til ein ny konstruksjon til grammatikken aukar talet på alternative reglar for somme ikkje-terminalar og kompleksiteten i parsetreet for andre. Vidare er den implementerte tilbakesporinga i Prolog såpass effektiv at dette i stor grad veg opp for kostnaden forbunde med å utforska ein eller fleire blindvegar, det vil seie parsingalternativ som ikkje fører nokon stads hen [Pereira -83, s. 96-97]. Når det gjeld parsaren tilhøyrande BussTUC-systemet har denne, som nemnt, vist seg å vere både rask og effektiv. På grunn av dei mange kutta unngår ein mykje av tilbakesporinga, parsaren vert spart for ein god del unødig ekstraarbeid og ein kuttar dermed ned på tidsforbruket. Ei av ulempene med ei slik løysing er likevel at ein del parsealternativ som burde vorte tekne omsyn til, ikkje vert prøvde i det heile. Kun første, moglege parse vert altså teken vare på.

Eit anna argument mot vanleg topp-ned-parsing – og botn-opp-parsing for den del – går på at parsingresultatet, ofte framstilt i form av ein trestruktur, er utilstrekkeleg som representasjon når det finst lokal ambiguitet og eventuelle ufullstendige strukturar. Ein får ganske enkelt ingen informasjon om desse fenomena – eit innputt går enten gjennom, eller så feilar det; Informasjon om det som har skjedd i løpet av prosessen vert ikkje teken vare på i noko tilfelle. Grammatikken til BussTUC er laga for å kunna handtera enkelte typar ufullstendige strukturar, og parsaren er dermed i stand til å analysera desse. Argumentet held like fullt med omsyn til ufullstendige strukturar som ikkje vert dekka av grammatikken.

Venstrerekursjon er eit velkjent problem i tilknyting til parsarar som arbeider topp-ned. Ein topp-ned-parsar vil gå i løkke dersom det finst reglar på forma `X → X Y.` eller ein serie reglar som `X → Y Z.`, `Y → W.` og `W → X.`. Ein kan få bukt med problemet ved å ta i bruk attributt, sjølv om uttrykka for enkelte konstruksjonar på denne måten vert noko meir komplekse enn dei elles ville vore. Regelen `np → np pp` kan til dømes skrivast `np(np(NP,PP)) → np(np(NP)),pp(PP)` [Gazdar -90, s. 158]. I TUC-grammatikken vert problemet unngått ved at ein opererer med ulike namn på predikat av liknande typar, som til dømes `noun_phrase`, `noun_phrases` og `noun_phrase1`, og ved at ein nyttar attributta på liknande vis som i dømet ovanfor (døme på grammatikkreglar finst i delkapittel 1.5.1 og i appendiks D).

4.2 Kartparsing

Innføringa av kartparsing er vanlegvis kreditert til Kay som nytta metoden som grunnlag for å vurdera og samanlikna ulike parsingstrategiar [Arnold][21]. I motsetnad til DCG-parsarar er kartparsarar svært fleksible. Dei fungerer nærast som eit generelt rammeverk som kan nyttast med ulike søkestrategiar, det vil seie både topp-ned og botn-opp så vel som djubde- og breidde-først, med fleire. Ulike søkestrategiar kan òg kombinerast til diverse hybride variantar.

Ein av dei store fordelane med kartparsarar er at dei ulike parsane – dei analyserte strengene – deler felles konstituentar. Ein gitt delstruktur kan nyttast av alle subsumerande strukturar. Ein konstituent vert såleis kun parsa ein gong, same kva for søkestrategi som er brukt. Både uferdige og fullstendige parsar vert lagra. Dette gjer det mogleg å prøva å tolka ugrammatiske strenger, det vil seie strenger som ikkje kan tildelast ein fullstendig, syntaktisk struktur, på eit semantisk og pragmatisk grunnlag. Kartparsarar har vanlegvis ingen problem med venstrerekursjon, ikkje ein gong når ein parsar topp-ned.

Når det gjeld lokal ambiguitet er det framleis slik at det vert lagra konstituentar som ikkje vert brukte i det endelege parseresultatet, men ein unngår altså den dupliseringa som kan finnast i tilknyting til andre typar parsarar. Til forskjell frå andre typar parsarar som gjerne produserer ei form for trestruktur gir ein kartparsar – grunna kartet (chart), eller databasen der parsane vert lagra – ein slags kompakt representasjon av den lokale ambiguiteten som måtte finnast. Kartparsarar representerer vidare ein effektiv måte å finna og presentera fleire, alternative parsar på, noko som er fordelaktig i høve til global ambiguitet. I eit NLIDB-system burde desse eigenskapane kunna nyttast til å få presentert dei ulike tolkingane av eit fleirtydig innputt for brukaren, slik at brukaren sjølv kan spesifisera kva han eller ho meinte ved ganske enkelt å velja kva for alternativ systemet skal gå vidare med. Ei slik tilnærming er spesielt aktuell i høve til genuint fleirtydige uttrykk.

Ulempa med kartparsing er først og fremst at ”alt” som er mogleg å gjera vert gjort, sjølv om mykje er unødvendig – det vil seie ikkje vert del av det endelege resultatet. Kva som faktisk vert prøvd og ikkje er sjølvsagt avhengig av valt søkestrategi. Ein kartparsar som arbeider topp-ned vil finna og leggja til kantar ein kartparsar som arbeider botn-opp ikkje vil leggja til, og omvendt. Med ein stor grammatikk vil kartet uansett kunna verta svært omfattande, uavhengig av kva for søkestrategi som er valt.

[21] Arnold refererer til Kay, Martin (1986): *Algorithm schemata and data structures in syntactic processing*. I Grosz et al. s 35-70.

Tida det tek å søka gjennom det på utkikk etter passande kantar kan faktisk representera eit problem. Meir om dette i avsnitt 4.4.2 nedanfor.

4.3 Samanlikning og val av metode

Tommelfingerregelen for val av parsingmetode seier at dersom det finst mange reglar for ein og same kategori er det ein fordel å nytta ein botn-opp-parsar. Dersom det derimot finst få reglar per kategori, men kategoriane på høgresida i desse reglane dannar utgangspunkt for mange nye reglar, det vil seie utgjer venstresida i mange andre reglar, er ein topp-ned-metode å føretrekka. Denne regelen fungerer vanlegvis bra i høve til mindre, meir oversiktlege grammatikkar. Noko verre er det i høve til store grammatikkar utan noko typisk mønster med omsyn til kva type reglar dei består av. BussTUC-grammatikken har typisk mange reglar per kategori, men har òg eit utal høgresidekategoriar som i sin tur dannar venstresida i andre reglar. I og med at grammatikken er skriven i Prolog og med utgangspunkt i DCG-formalismen er det naturleg at den innebygde Prologparsaren vert brukt. På denne måten slepp ein å skriva og leggja til ein eigen parsingmodul. Topp-ned-parsarar vert rekna som gode i høve til langdistanseavhengigheit, til dømes i tilknyting til relativsetnningar, men er ikkje særleg minneeffektive (memory efficient). Med botn-opp-parsarar er det omvent: Dei er som oftast meir minneeffektive, men mindre gode på langdistanseavhengigheit.

Pereira samanliknar i si doktoravhandling kort ein vanleg tilbakesporingsparsar med ein kartparsar, begge topp-ned-parsarar, i tilknyting til CHAT-80-systemet. Spørsmålet i høve til effektivitet vert, i følgje Pereira, om kostnaden forbunde med lagring av partielle analysar vert rettferdiggjort på grunnlag av elimineringa av duplisert arbeid. Resultata han kjem fram til gjennom si samanlikning tyder på at dei teoretiske fordelane kartparsarar har ikkje vert realiserte i høve til CHAT-80-grammatikken. Bare 16% av arbeidet tilbakesporingsparsaren i CHAT-80 utfører er bortkasta arbeid, og han undersøker bare 75% av dei analysestiane kartparsaren utforskar. Talet på delanalysar kartparsaren lagrar i tilknyting til dei innputtdøma Pereira undersøker er seks gonger talet på ikkje-terminalar som faktisk vert brukte. Pereira konkluderer med at kostnadane med omsyn til både tid og lagringsplass dette krev overgår det bortkasta arbeidet til tilbakesporingsparsaren [Pereira -85, s.98]. Covington har gjort ein liknande test og finn, i likskap med Pereira, at vanlege tilbakesporingsparsarar er raskare enn kartparsarar. ”The sad news is that as the algorithms get ”better” the parsing gets slower and slower.” uttalar han og oppgir mangel på optimalisering, storleiken på grammatikkane og kostbare subsummeringssjekkar som moglege grunnar for dei dårlege kartparsingsresultata [Covington -94, s. 191-192].

Pereira, blant andre, meiner like fullt at det finst andre argument for bruk av kartparsingmetodar, og dette er noko av grunnen til at det vart satsa på å testa nettopp ein kartparsar i tilknyting til BussTUC. Kartparsarar er normalt betre enn vanlege tilbakesporingsparsarar når det gjeld feilkorreksjon og gjenoppretting (recovery). System baserte på naturleg språk berekna på praktisk bruk har vanlegvis eit relativt stort behov for feilkorreksjon, noko til dømes CHAT-80-systemet manglar. Vidare er det tenkjeleg at kartparsingsalgoritmar, i eigenskap av å lagra partielle analyser, kan representera ein måte å handtera ufullstendige innputt – som elliptiske fraser – på, samstundes som grammatikken ein nyttar i utgangspunktet er laga kun for fullstendige setningar [Pereira -85, s.109-110]. Elliptiske fraser er setningar der grammatisk viktige ord er uteletne fordi dei ikkje er nødvendige for å få fram kva som er meint. Om slike fraser i realiteten er ufullstendige kan diskuterast, men sett i høve til ein gitt grammatikk kan dei altså reknast som det. Eit døme på norsk kan vera ”Tina et eple og Ola pærer” der verbet *et* er utelete i siste del av setninga (det vil seie i delsetninga etter konjunksjonen). Pereira gir inga skildring av korleis handteringa av

denne og liknande typar fraser kan utførast i praksis, men det burde la seg gjera å nytta seg av det som måtte finnast av kontekst – det vil seie byrjinga av aktuelle setning, eller eventuelt tidlegare innputt – for å løysa denne typen problem. Ei elliptisk frase er gjerne – som dømet ovanfor viser – ei frase med lik struktur som den føregåande, med unntak av den uteletne konstituenten. Dermed burde frasene kunna samanliknast og manglande konstituent – funne med utgangspunkt i første, fullstendige frase eller setning – gjenbrukast i den elliptiske frasa[22].

Det er verdt å merke seg at avhandlinga til Pereira vart skriven i 1983. Maskinane ein opererer med i dag er langt større, kraftigare og raskare enn dei var tidleg på 80-talet. Høgt tidsforbruk og liten lagringsplass er gjerne ikkje det som utgjer hovudproblemet lenger, til tross for at kravet til fart er høgt – spesielt i tilknyting til online-applikasjonar. Vidare har grammatikken mykje å seie i høve til kva utfall bruk av ulike parsingmetodar får. BussTUC-grammatikken er både større og bygd ut ifrå ein noko annan formalisme enn det CHAT-80-grammatikken og grammatikken Covington arbeidde med er. I tillegg er leksikonet av ein annan storleik. Dette er aspekt som kan føra til andre resultat enn dei som vart presenterte ovanfor.

4.4 BussTUC-kartparsar

Målet med å byta ut den opphavelege BussTUC-parsaren med ein kartparsar var i utgangspunktet å oppnå ein parsar som kunne handtera og løysa fleire av ambiguitetsproblema enn det tilbakesporingsparsaren gjer. Den nye parsaren bør òg vere like effektiv som den originale. Bratseth, som skreiv hovudfagsoppgåva 'Bustuc – A Natural Language Bus Traffic Information System' i 1997, nemner parsaren som ei av avgrensingane systemet har: "TUC has reached a level of sophistication where an improved parser is the single feature that will contribute most to improving the performance of the system […]" [Bratseth -97, s.11]. Ein god, robust parsar bør i følgje Bratseth kunna handtera meir enn bare setningar som kan tildelast ein komplett analyse. Parsaren bør prøva å gjetta seg fram til manglande, men sannsynlege ord og fraser, samt å ordna på ord som ikkje passer inn med resten av setninga. Ein probabilistisk parsar vert difor foreslått som det mest ideelle [ibid.]. Bratseth meiner at ein kartparsar potensielt vil kunna bøta på ein del av dei gjenståande problema knytte til BussTUC-systemet og viser til Allen sitt TRAINS-prosjekt der kartparsing fører til gode resultat [Allen -95 i Bratseth -97, s.12 og 22]. Han argumenterer med at kartparsing er ein meir akseptert og standard parsemetode enn probabilistisk parsing. I høve til dette bør det poengterast at probabilistisk parsing og kartparsing ikkje nødvendigvis utelukkar einannan: Kartparsarar kan, som dei fleste typar parsarar, utvidast til ein probabilistisk parsar dersom det skulle vere behov for det.

Parsaren som vert integrert og testa tek utgangspunkt i ein enkel topp-ned-kartparsar henta frå Gazdar og Mellish [Gazdar -90, kap. 6.9]. Algoritma kan oppsummerast slik:

1. Start med tomt kart / database
2. Ta neste ord i innputtstrengen og slå det opp i ordboka. Tildel ordet leksikalsk(e) kategori(ar) ut ifrå oppslaget.

[22] Meir om handtering av ufullstendige og ugrammatiske innputt finst blant anna i Lee, Yoon-Hee 1990: *Handling Ill-Formed Natural Language Input for an Intelligent Tutoring System*, doktorgradavhandling, Illinois Institute of Technology.

3. Legg til ein kant frå gjeldande node til neste node innehaldande sjølve ordet, samt informasjonen funnen i ordboka.
4. Gjenta steg 2 og 3 til det ikkje er fleire ord igjen, det vil seie til innputtstrengen er tom.
5. Start analysen med utgangspunkt i første grammatikkregel på høgaste nivå – normalt setningsregelen – og legg på ein aktiv kant for denne frå og til startnoda.
6. Sjekk om det finst inaktive kantar denne aktive kanten treng, det vil seie inaktive kantar med kategorien som ligg først, som merkelapp (LABEL), i TOFIND-lista til den aktive kanten og som startar i den noda der den aktive kanten sluttar.[23]
7. Dersom det ikkje finst nokon passande, inaktiv kant, gå vidare med utgangspunkt i kategorien som ligg først på TOFIND-lista i gjeldande, aktive kant og gjenta frå steg 5 (det vil seie med grammatikkregelen / -reglane som stemmer overeins med gjeldande kategori).
8. Dersom det finst ein passande, inaktiv kant, legg til ein ny kant som spenner frå startnoda til den aktive regelen og til sluttnoda til den inaktive regelen. Merkelappen (LABEL-en) til den inaktive kanten vert fjerna frå TOFIND-lista til den nye kanten og i staden lagt til i FOUND- og PARSE-lista. Før denne nye kanten faktisk kan leggjast til vert det sjekka om han finst i kartet frå før.
9. Dersom denne nye kanten er inaktiv, søk etter aktive kantar som treng denne, det vil seie aktive kantar som ender der den inaktive startar og som har aktuelle kategori først på TOFIND-lista.
10. Dersom den nye kanten framleis er aktiv, ta utgangspunkt i første kategori på TOFIND-lista og gjenta frå steg 5.

Dette held fram til ein har ein eller fleire fullstendige parsar, det vil seie inaktive kantar med merkelapp i form av startsymbolet `sentence` (i dette tilfellet) som spenner frå start- til sluttnode. Figur 4.1 nedanfor illustrerer algoritma.

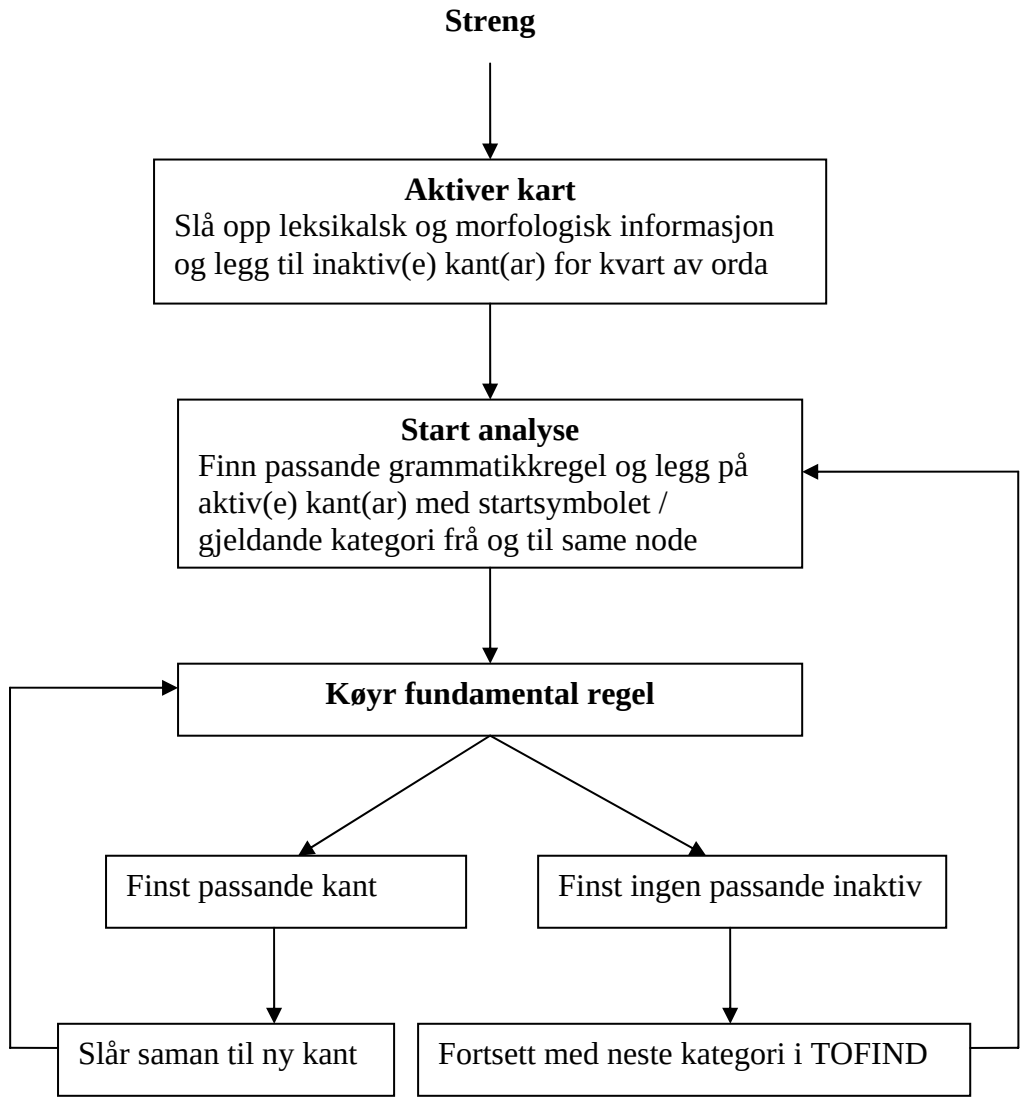
Predikata BussTUC-grammatikken består av er organiserte med tanke på topp-ned-søk, og det var difor naturleg å ta utgangspunkt i ein slik søkemetode. Den standard kartparsarkoden brukt som utgangspunkt har vorte endra og utvida, litt etter litt, for å kunna handtera BussTUC-grammatikken. Dette vart i første omgang gjort ut ifrå ei særskilt testsetning, *når går neste buss fra Lade?*. Ein av grammatikkreglane som vert brukte under parsing av denne setninga består blant anna av eit kategorialgrammatisk uttrykk innehaldande operatoren `'-'`. For å gjera sporingsprosessen så enkel og oversiktleg som mogleg vart ein stor del av dei grammatikkreglane som normalt ville vorte prøvde, men ikkje brukte, kommenterte ut. Etter å ha tilpassa kartparsarreglane og oppnådd ein fullstendig parse for aktuelle testsetning vart så regelgruppe for regelgruppe lagt til att og parsaren testa fortløpande. Dei påfølgjande testsetningane, blant andre *når i dag går bussen fra lade til sentrum?* og *når er det første buss går fra Lade?*, vart prøvde pressa gjennom på liknande måte. Nokre av grammatikkreglane som vart brukte i tilknyting til parsinga av desse setningane inneheld ein eller fleire av dei kategorialgrammatiske operatorane `'-'`, `'\'` og `'/'`. Somme av desse kategorialgrammatiske sekvensane er rimeleg komplekse, det vil seie høgresida i det kategorialgrammatiske uttrykket består av fleire element – som i regelen for når-spørsmål:

```
whenq(which(Y) :: P) --- >
      when,
      adverbial1(AA,BB,CC,DD),
      !,
      adverbial1(_,Y,Subj,P3) \
            (prep(in),the(Y),w(noun(time,sin,u,n))),
```

[23] LABEL, TOFIND, FOUND og PARSE er argument i kantpredikatet `edge`. Sjå appendiks E.

```
      !,
      question0(P) –
            (adverbial(in,Y,Subj,P3),adverbial(AA,BB,CC,DD)).
```

Meir om dette i avsnitt 4.5 nedanfor.



Figur 4.1: Kartparsingalgoritma

Ein del av kartparsarreglane som er endra eller lagde til er tekne med i teksten nedanfor. Heile koden er å finna i appendiks E. Den generelle algoritma ovenfor kan spesifiserast med utgangspunkt i kartparsarpredikata på denne måten:

1. Hovudpredikatet `Parse` tek imot innputtstrengen, sørgjer for at eventuelle kantar frå tidlegare parsingar vert fjerna frå kartet / databasen, set i gang `start_chart` som initialiserer kartet, definerer startsymbolet ved hjelp av `initial` og startar sjølve analysen ved hjelp av `start_active`.

2. `start_chart` tek føre seg innputtstrengen (på listeform), ord for ord, og sørgjer for at det vert lagt på ein eller fleire inaktive kantar for kvart av orda på rett plass i kartet. Eit ord som tilhøyrer tre kategoriar fører til tre inaktive kantar plasserte på same plass i kartet, ein for kvar kategori. `Start_chart` held fram heilt til innputtstrengen er tom. `morfological_word` får tilgang til dei ulike morfologiske kategoriane gjennom `lcodes` – ei slags kategoriliste tilhøyrande filene 'lex.pl' og 'morph.pl' – og nyttar `member` for å gå gjennom denne lista.

3. `start_active` sørgjer for at grammatikkreglane vert tekne i bruk og startar, som definert i `parse`-regelen, frå node 0 (kalla `V0`) og med grammatikkreglane på øvste nivå, nemleg setningsreglane. `Start_active` går gjennom alle aktuelle grammatikkreglar (for gjeldande kategori) og legg på kantar i kartet ved hjelp av dei ulike `add_edge`-reglane. Predikatet `p2list` sørgjer for at høgresida i reglane, døtrene om ein vil, vert gjorte om til listeform slik kartparsaren krev.

4. Gruppa av `add_edge`-reglar består først og fremst av regel 12 og 13 som representerer den fundamentale regelen[24]. Regel 12 er ein fundamental topp-ned-regel som handterer aktive kantar og som sørgjer for at det alltid vert leita etter passande, inaktive kantar for desse. Regel 13 handterer inaktive kantar tilsvarande, det vil seie sørgjer for at det alltid vert søkt etter passande, aktive kantar for kvar inaktiv. `add_edge`-regel 1 til og med 6 sørgjer for handtering av diverse kategorialgrammatiske uttrykk, medan regel 7 o g 8 handterer blant anna prologuttrykk (typisk semantiske testar og liknande) og andre uttrykk som ikkje skal leggjast til, eller som ikkje skal leggjast til direkte, som kantar i kartet (desse ekstra `add_edge`-reglane er viste nedanfor). Før ein kant eventuelt vert lagt til sjekkar regel 9 og 10 om han ligg i kartet frå før.

5. `add_edge`-reglane sørgjer for at det vert lagt på ein ny kant for kvar gong ein aktiv kant finn ein passande inaktiv kant, eller omvendt. Dersom det ikkje finst nokon passande, inaktiv kant for ein aktiv kant vert det teke utgangspunkt i kategorien først på TOFIND-lista til gjeldande, aktive kant og `start_active` vert køyrd på ny.

4.4.1 Skildring av reglar som er lagt til eller endra

Som nemnt i delkapittel 1.5.1 er grammatikkformalismen nytta i BussTUC-systemet noko spesiell. Dette skapte ein del utfordringar i høve til det å få integrert kartparsaren. Kartparsaren måtte blant anna lagast slik at alle kutt vert sett bort ifrå og at reine prologuttrykk (semantiske sjekkar og liknande) ikkje vert lagt på som kantar i kartet, men køyrde undervegs. Dette vert gjort med hjelp av to ekstra `add_edge`-reglar, samt predikatet `rejectclause`:

```
add_edge(V2,V1,V2,Cat,[{PrologClause}|Cats],Parse,_):-
     PrologClause,
     (true -> add_edge(_,V1,V2,Cat,Cats,Parse,_))
     ;!.
```

Regelen ovanfor sørgjer for at eventuelle prologuttrykk som finst i ein del av grammatikkreglane ikkje vert lagde på som kantar i kartet, men vert køyrde undervegs.

[24] Reglane er nummererte i appendiks E.

Dersom ein sjekk gir ”true” går parsaren vidare med neste kategori på `TOFIND`-lista. Dersom han feilar vert det køyrd eit kutt.

```
add_edge(V2,V1,V2,Cat,[Funny|Cats],Parse,_):-
     rejectclause(Funny),
     add_edge(_,V1,V2,Cat,Cats,Parse,_).


rejectclause(!).
rejectclause(accept).
```

I BussTUC-grammatikken finst det vidare ein del terminalar på forma `w(verb(go,pres,fin))` som i det opprinnelege systemet stemmer overeins med element i listene som er utputt frå den leksikalske analysen (sjå parsingdømet i appendiks C). I og med at kartparsaren legg til ord som kantar utan `w`-predikatet, det vil seie på forma `verb(go,pres,fin)`, må `w`-predikatet fjernast før den aktuelle terminalen vert prøvd lagt til eller samanlikna med aktuelle kant i kartet. Nok ein ekstra `add_edge`-regel vart lagt til i kartparsarkoden for å handtera dette:

```
add_edge(V2,V1,V2,Cat, [w(Cat1)|Cats], Parses,Fillers):-
     add_edge(V2,V1,V2,Cat,[Cat1|Cats],Parses,Fillers).
```

Dei mange kategorialgrammatiske uttrykka utgjorde ei av dei største utfordringane. Dei kategorialgrammatiske operatorane ”`/`”, ”`\`” og ”`-`” vert brukte til flyttingar og overføringar av delkonstituentar under deriveringa frå spørsmål eller kommando til deklarative setningar. Eit ja-nei-spørsmål som ”*går neste buss fra Lade kl. 2*?” vert gjort om til ”*neste buss fra Lade går kl. 2*”. Det resulterande deklarative uttrykket vert så sjekka mot faktaa i databasen. For å kunne flytta verbet som vert funne først i spørsmålsinnputtet til rett plassering i den deklarative setninga må all leksikalsk informasjon om verbet takast vare på og gjerast tilgjengeleg der han trengs. I kartparsaren vert dette løyst ved at kantane vert utvida med eit ekstra argument, `FILLER`. Ein kant består dermed av seks argument og vert sjåande slik ut: `edge(START,FINISH,LABEL,TOFIND,FOUND,FILLER)`. `FILLER`-argumentet representerer ei differanseliste der informasjon om alle konstituentar som skal flyttast vert lagde til. Ein av grammatikkreglane for ja-nei-spørsmål demonstrerer bruken av operatoren ”`-`”:

```
ynq(P) --->
    w(verb(Speak,Pres,fin)),
    saa0,
    {Speak \== have},
    negation(_N),
    !,
    st2(P) - w(verb(Speak,Pres,fin)).
```

Operatoren tyder her at ein skal gå ut ifrå at det finst eit verb på forma `verb(Speak,Pres,fin)` og prøva `st2` først med og deretter eventuelt utan dette. `st2` er ein av ’statement’-reglane. I dette tilfellet går `st2` til `st`, som i sin tur går til `noun_phrase, verb_phrase`. Det aktuelle verbet, *går* – dersom ein framleis tek utgangspunkt i ja-nei-spørsmålsdømet ovanfor – vert lagt til som element i `FILLER`-lista idet siste dottera i `ynq`-regelen vert køyrd gjennom ein `start_active`-regel laga spesielt for kategorialgrammatiske uttrykk. Deretter vert det teke i bruk i verbfrasa tilhøyrande `st`. `start_active`-regelen laga for å kunna handtera dei kategorialgrammatiske uttrykka ser slik ut:

```
start_active(V0,Cat1 – Cats4):-
```

```
    foreach(rule1(Cat1,Cats),
     add_edge(V0,V0,V0,Cat1,Cats,[Cat1],Cats4)).
```

Variabelen `Cats4` står for ei differanseliste bestående av element som til dømes `verb(Speak,Pres,fin)`, det vil seie element som skal nyttast seinare i parsingprosessen og som difor vert lagde på som `FILLER`-element i dei følgjande kantane. Differanselista som variabelen `Cats4` står for vert til i `add_edge`-regelen for kategorialgrammatiske uttrykk:

```
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers):-
       asserta(edge_aktiv(V2,V1,V2,Cat,[Cat1|Cats],Parses,Fillers)),
       p2dlist(Cats2,Cats3),
       (member_dl(w(Cat3),Cats3) -> p2dlist(Cat3,Cat4),
            append_dl(Cat4,Cats3,Cats4)
       ; start_active(V2,Cat1-Cats3)),
        start_active(V2,Cat1 - Cats4).
```

Grunnen til at `FILLER`-elementa vert gjorde om til differranselisteform allereie i `add_edge`-regelen er at kategorialgrammatiske fraser kan vere komplekse, det vil seie høgresida kan bestå av ei heil liste. `member_dl` vert brukt for å gå gjennom og fjerna eventuelle `w`-uttrykk som kan finnast som del av dei kategorialgrammatiske uttrykka. Predikatet som vert nytta for å gjera vanlege paranteslister om til differanselister, `p2dlist`, er å finna i appendiks E.

Dei tre kategorialgrammatiske operatorane brukte i grammatikken vert behandla på same måte av kartparsaren ved at ”/” og ”\” vert gjorde om til ”-”:

```
add_edge(V2,V1,V2,Cat,[Cat1 / Cats2|Cats],Parses,Fillers):-
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers).

add_edge(V2,V1,V2,Cat,[Cat1 \ Cats2|Cats],Parses,Fillers):-
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers).
```

For å kunne sjekka og nytta `FILLER`-listeelement der desse trengs vart den fundamentale topp-ned-regelen utvida med ein medlemsfunksjon, `member_dl`, for handteringa av differanselister:

```
add_edge(V2,V1,V2,Cat1,[Cat2|Cats],Parses,Fillers):-
 asserta(edge_aktiv(V2,V1,V2,Cat1,[Cat2|Cats],Parses,Filler)),
      foreach(edge_inaktiv(V2,V2,V3,Cat2,[],Parse,Filler),
      append_dl(Filler,Fillers,Ny_filler),
      add_edge(_,V1,V3,Cat1,Cats,[Parse|Parses],Ny_filler),
      (member_dl(Cat2,Fillers) ->
      add_edge(V2,V2,V2,Cat2,[],Cat2,X--X))
      ;
      start_active(V2,Cat2,Fillers).
```

Dersom det ikkje vert funne nokon passande, inaktiv kant ved hjelp av `foreach` vert `member_dl` og tilhøyrande `member_var` brukte for å sjekka om kategorien ein er ute etter finst i differanselista `FILLER`. `member_var` sørgjer for at eventuelle variablar i `FILLER` ikkje fører til ein match:

```
member_var(X,Var):- var(Var),!,fail.
member_var(X,[V|_]):- \+ var(V), X=V.
member_var(X,[_|Y]):- member_var(X,Y).
```

Dersom aktuelle kategori finst i `FILLER` vert han lagt på som inaktiv kant frå og til aktuelle node slik at han kan nyttast av den aktive kanten som treng han. Dersom kanten det vert leita

etter verken finst som passande kant i kartet frå før eller som element i `FILLER` køyrast `start_active` som vanleg.

`add_edge`-regelane som sjekker om kantar som er klare for å leggjast til i kartet finst frå før (regel 9 og 10 i appendiks E) skiljer seg frå den opprinnelege varianten brukt i Gazdar og Mellish blant anna ved at det er brukt ein fri variabel for argumentet `Parse`. Utan ein fri variabel fekk BussTUC-kartparsaren nemleg problem med rekursive reglar som til dømes `verb_complement`-reglane. Dette utdraget henta frå appendiks D viser dei potensielt problematiske reglane:

```
verb_complements0(V,X,S,Com1P1,Com12P3) --->
    verb_complements(V,X,S,Com1P1,Com12P3).

verb_complements(V,X,S,Com1P1,Com12P3) --->
    verb_complement(V,X,S,Com1P1,Com12P2),
    verb_complements10(V,X,S,Com12P2,Com12P3).

verb_complements10(V,X,S,Com1P1,Com12P3) --->
    verb_complements0(V,X,S,Com1P1,Com12P3).
```

Venstrerekursjon utgjer vanlegvis ikkje noko problem i høve til kartparsing, men ein vanleg `Parse`-variabel i `add_edge`-regel 9 og 10 gjer at òg `PARSE`-listene vert samanlikna når to kantar vert sjekka med omsyn til likskap. For `verb_complement`-reglane sin del fører dette til at kantar vert lagde på kartet feilaktig, nettopp fordi `PARSE`-listene gjer at likskapstesten feilar. Under følgjer eit utdrag frå parsetreet i appendiks C som kan vere med på å demonstrera dette:

```
do_phrase
     .
     .
  verb_complements0
    verb_complements
      verb_complement
        adverbial1
          nil
        {}
      verb_complements10
        verb_complements0
          {}
           .
           .
```

Dette viser at første `verb_complements0`-predikat (på høgaste nivå i treet) høyrer med inn under `do_phrase`. Når `verb_complements0`-kanten vert inaktiv skal han med andre ord finna ein aktiv `do_phrase`-kant. I staden finn han den aktive `verb_complements10`-kanten lagt på tidlegare (to nivå under, men frå og til same node). Dersom `PARSE`-listene vert samanlikna i `add_edge`-regel 5 eller 6 fører dette til at nok ein ny, inaktiv `verb_complements10`-kant feilaktig vert lagt på i kartet:

```
edge(0,0,verb_complements10(…),[],[[[[verb_complements0(…)],
verb_complements10(…)],[[complement1(…)],adverbial1(…)],
verb_complement(…)],verb_complements(…)],verb_complements0(…)],
verb_complements10(…)],X--X)
```[25]

[25] Argumenta er tekne bort og erstatta med ”(...)” for å gjera dømet meir leseleg.

Denne passar så inn i den aktive `verb_complements`-kanten som finst frå før, som igjen passar inn i den aktive `verb_complements0`-kanten, osb. Den inaktive `verb_complement10`-kanten som ligg på kartet frå før, og som den nye vert likskapssjekka mot, ser slik ut:

```
edge(0,0,verb_complements10(…),[],[[verb_complements0(…)],
verb_complements10(…)], X))
```

Ved ikkje å ta omsyn til `PARSE`-listene vert kantane derimot sett på som like og den nye `verb_complement10`-kanten ovanfor vert altså forkasta slik han burde.

4.4.2 Effektivisering

Ei undersøking av kartet etter at ein mogleg parse for testsetninga *"når går neste buss fra lade?*" er funnen viste at det fanst heile 10 941 kantar alt i alt, 1507 av desse inaktive. Bare frå og til startnoda fanst det tilsaman 614 kantar. Så lenge parsaren må søka gjennom heile den samlinga av lagra kantar som til ei kvar tid finst i kartet – aktive som inaktive – for kvar gong han leiter etter ein spesiell kant, seier det seg sjølv at parsinga ikkje kan verta vidare effektiv. Første utgåve av BussTUC-kartparsaren brukte over tre minutt(!) på å parsa eit innputt før effektiviseringstiltak vart sette i verk (vel og merka før kompilering). Ei gruppering av kantane i henhaldsvis aktive og inaktive kantar reduserte parsingtida med omlag eit minutt. For å oppnå ei slik gruppering vart det opprinnelege `edge`-predikatet omgjort til to nye predikat, `edge_aktiv` og `edge_inaktiv`. To nye `add_edge`-reglar, 9 og 10, vart lagde til i staden for den opprinnelege regelen for likskapssjekk. Sidan aktive kantar leiter etter inaktive som startar der dei sjølve ender og inaktive tilsvarande leiter etter aktive som ender der dei sjølve startar, vart det òg lagt til eit ekstra argument i form av start- og sluttnode (`START` og `FINISH`), respektivt, først i `edge`-predikata. `add_edge`-reglane vart utvida tilsvarande. Kantane vert nemleg indekserte etter første argument. Ved å utvida som skildra ovanfor, og dermed gruppera kantane endå finare, sørgjer ein difor for at søket etter passande kantar går raskare. Dette siste tiltaket fekk tidsforbruket ned med nok eit lite minutt. Gjeldande testsetning vart til slutt parsa i løpet av litt over 30 sekund.

4.5 Testing og resultat

Kun sjølve parsingdelen vert prøvd ut. Til dette trengs filene *busdat.pl*, *dagrun_n.pl*, *dict_n.pl*, *dictn.pl*, *evaluate.pl*, *gram_n.pl*, *lex.pl*, *morph_n.pl*, *topreg.pl*, *tuc.pl*, *main.pl*, *utility.pl*, *fernando.pl*, *semantic.pl* og *namehashtable.pl*.

Som det føregåande viser var kartparsaren på eit tidspunkt i stand til å gje ut fullstendige analysar av den typen innputt som den første testsetninga representerer, som til dømes *når går neste buss til lade?* og *går buss 4 til lade?*. Det vidare arbeidet med å få parsaren til å handtera andre typar setningar – i framste rekke innputt som kravde parsing av komplekse kategorialgrammatiske sekvensar – førte derimot ikkje til noko endeleg resultat innanfor dei tidsrammene som var sette for dette hovudfagsarbeidet. Dermed var det heller ikkje mogleg å få testa parsaren på testsettet bestående av problematiske innputt, slik planen opprinneleg var.

Sjølve behandlinga av enkle, så vel som komplekse, kategorialgrammatiske uttrykk ser ut til å fungera greit: Element henta frå kategorialgrammatiske uttrykk vert lagde på `FILLER`-lista og tekne i bruk seinare slik dei skal. Parsinga elles ser òg stort sett normal ut. Problema som nå står att kan sjå ut til å vere relaterte til enkelte av prologuttrykka i grammatikken. Somme av dei semantiske sjekkane, nokre av dei tilhøyrande den semantiske typesjekkaren, verkar ikkje til å verta instansierte, noko som gjer at heile parsingsprosessen stoppar opp.

I og med at kartparsarkoden ikkje er fullendt har han heller ikkje vorte kompilert, noko som i sin tur medfører at ein ikkje eigentleg får noko klart inntrykk av kor rask parsaren faktisk er, eller ville vorte. Utan eventuelt å gjera inngrep som ville ha redusert kvaliteten på parsinga er det vel heller tvilsomt om kartparsaren nokon gong kunne vorte like rask som, eller raskare enn, den opprinnelege BussTUC-parsaren. Som dei fleste innan feltet òg rapporterar er det som oftast slik at kartparsarar kjem dårlegare ut med omsyn til tid enn det vanlege tilbakesporingsparsarar gjer. Kvaliteten på analysen, derimot, samt handteringa av fleirtydige og ufullstendige innputt, skårar gjerne kartparsarar høgare på. Dei få resultata eg var i stand til å få ut av den uferdige karparsaren verka til å sjå fine ut sett i høve til resultata originalsystemet gir for same innputt. Sjå appendiks C for døme på eit parseresultat.

4.6 Diskusjon og gjenståande arbeid

Ut ifrå arbeidet som er utført i tilknyting til denne oppgåva er det framleis vanskeleg å seie om kartparsing verkeleg kunne vore eit bra alternativ til den nåverande BussTUC-parsaren. Med omsyn til tidsforbruk er det – som nemnt – noko usikkert, spesielt med tanke på at BussTUC er ein online-applikasjon der kravet til fart er særskilt høgt. På den andre sida talar innputtkorpusa sitt klare språk med tanke på kva for utfordringar BussTUC-parsaren har å stri med. Som me har sett uttrykker BussTUC-brukarane seg til dels svært ulikt og urovekkande ofte ugrammatisk og ufullstendig i høve til den grammatikken systemet har å forholda seg til. Dette er altså aspekt som kan seiast å styrka argumentasjonen for bruk av ein kartparsar. Fordelen med kartparsinga i så måte er jo nettopp – som eg har vore inne på tidlegare i kapitlet – at det vert teke vare på delanalysar som eventuelt kan hentast inn frå kartet og nyttast dersom det ikkje skulle la seg gjera å komma fram til ein full parse.

I tillegg til å få kartparsaren ferdig og i stand til faktisk å parsa heile setningar bør det lagast eigne reglar for omsetjinga frå parsingresultat til TQL slik at sjølve databaseoppslaget og svargenereringa òg kan undersøkast.

4.6.1 Optimalisering

Andre alternativ

Med tanke på ei optimalisering av kartparsaren kunne det vore interessant å prøvd ut andre metodar, som botn-opp, eller kanskje helst ein kombinasjon av fleire metodar. Å kombinera topp-ned- og botn-opp-algoritmer har i mange tilfelle vist seg å føra til meir effektive parsarar og gode resultat. Ein såkalla venstrehjørne-parsar (left-corner parser), til dømes, startar – som botn-opp-parsarar gjer – med å bestemma kva type konstituent første innputtord utgjer første del av. Resten av aktuelle frase vert så parsa topp-ned [Covington -94, s.158-165].

Vidare kan det vere interessant å prøva å endra søkemetoden frå djubde- til breidde-først. Djubde-først vil seie at ein utforskar ein moglegheit om gongen. Kun dersom første valde sti til slutt feilar sjekker ein andre alternativ (det vil seie neste regel av same type). Eit standard problem med denne metoden er at han i verste fall kan resultera i infinitte søk dersom greinene i søketreet er infinitte. Søker ein breidde-først prøver ein ut alle moglege val på ein gong, det vil seie ein jobbar seg fram, eitt steg om gongen, i alle gjeldande reglar av same type. Fordelen med å nytta breidde-først er at ein unngår å følgja eventuelle feile val heilt til endes. Dermed unngår ein samstundes ein god del tilbakesporing. Breidde-først-søking vil alltid finna ei løysing og stoppa, så lenge det *finst* ei løysing. Den største ulempa med metoden er det faktum at ein må halda styr på alle vala. Dersom det finst mange like predikat, det vil seie venstresider med same symbol, vert det raskt svært mykje data på ein gong.

Endringar i parsemetode og / eller –søk ville mest truleg gjort det nødvendig òg å endra på rekkefølgja grammatikkreglane nå står i.

Restriksjonar

Det kan òg kanskje vere aktuelt å prøva ut ulike restriksjonsmetodar som går ut på å redusera mengda arbeid parsaren må gjera. Ranang skildrar i si semesteroppgåve 'Sentence Generation' ein metode for å avgrensa grammatikk og leksikon – på grunnlag av aktuelle innputtstreng – før sjølve parsinga, for slik å avgrensa søkeområdet for parsaren. Ranang konsentrerer seg – som tittelen på oppgåva hans tilseier – først og fremst om setningsgenereing, men diskuterer òg metoden med utgangspunkt i DCG-parsing. Ranang meiner at nokre av hovudideane i genereringsmetoden bør kunna nyttast i tilknyting til denne typen parsing. Blant anna bør leksikonet kunna avgrensast i høve til dei leksikalske elementa i innputtstrengen, ei avgrensing som kan utførast under sjølve parsingprosessen. Grammatikken kan avgrensast blant anna ut ifrå POS-markørar (Part-of-Speech tags) dersom innputt vert markert (tagga), til dømes ved hjelp av ein HMM (Hidden Markov Model), før sjølve parsinga. Andre metodar, som bruk av seleksjonsrestiksjonar eller andre trekk som er tekne med i leksikon og reglar frå før, burde òg kunna nyttast på tilsvarande måte. Grunnprinsippet går ut på at grammatikken skal vere avgrensa til reglar med kroppar (høgresider) som kun består av enten POS-kategoriar som finst i det avgrensa leksikonet eller av ikkje-terminalar som utgjer hovudet i andre reglar som er i tråd med denne såkalla relevansrestriksjonen [Ranang -03, s.6-10]. Når det gjeld kartparsing skjer det ei avgrensing i høve til leksikon idet kartet vert initialisert, i og med at initialiseringa går føre seg på grunnlag av nettopp dei aktuelle leksikalske oppslaga som vert funne ut ifrå kvart enkelt ord i innputt. Ranang samanliknar faktisk avgrensingsmetoden sin (knytt til DCG-parsing) med initialiseringa i kartparsing. Det er med andre ord først og fremst grammatikkavgrensinga som kan vere av interesse i høve til kartparsing.

Covington nemner ein liknande metode der parsaren vert utvida med eit såkalla orakel eller ein tabell som hindrar at grammatikkreglar som ikkje fører til suksess vert prøvde ut. Ein parsar bestående av ein slik orakeldel går gjennom ein serie nummererte tilstandar. I kvar av desse sjekkar oraklet neste ord i innputtstrengen og gir ut frå dette tilbakemelding til parsaren om kva grammatikkregel som skal nyttast, samt kva for neste tilstand han skal i. Covington viser til Pereira for ei meir detaljert innføring i bruk av orakel[26] [Covington -94, s. 192].

[26] Pereira, F. C. N. (1985): *A new characterization of attachment preferences*. I Dowty et al., s. 307-319.

5 Brukarstudium og bruksanalyse

Denne delen tek føre seg eigenskapar, fordelar og ulemper ved NLIDB-system generelt og ved BussTUC-systemet spesielt. Det vert fokusert på brukarane og korleis dei ser ut til å oppleva og reagera på denne relativt nye forma for interaksjon. Kapitlet seier litt om korleis brukarar handterer denne typen system, kva som fungerer og ikkje, samt kva som kan gjerast for å optimalisera interaksjonen mellom brukar og system. Den teoretiske delen er i stor grad basert på Jussi Karlgren sin artikkel 'Discourse Modality and User Expectations, The Interaction of Discourse Modality and User Expectations in Human-Computer Dialog' [Karlgren -92], medan dei fleste meir systemspesifikke skildringane og døma er baserte på BussTUC-systemet og henta frå det tilhøyrande korpuset. Avsnitt 5.3 nedanfor tek føre seg ein generell feiltypologi for feil som kan oppstå ved interaksjon mellom menneske og NLP-system og knyt feila i dei sjølvlaga feilmeldingskategoriane skildra i kapittel 2 opp mot denne. Ei slik klassifisering av feiltypar seier meir om kva feila skuldast og dermed noko om kva for feil som faktisk kan utbetrast og ikkje. Klassifiseringa kan vere nyttig for å kartleggja fordelinga av feiltypar med omsyn til eit gitt system, og for å finna fram til det løysingsalternativet og den -metoden som er mest aktuell i høve til aktuelle feilfordeling.

5.1 Positive sider ved NLIDB-system

Samanlikna med andre typar grensesnitt – som databasespråk-, meny- og skjemabaserte grensesnitt – finst det klare fordeler forbunde med grensesnitt baserte på naturleg språk. Nokre få av dei vart kort skisserte i første del av oppgåva, men det finst altså fleire: Enkelte spørsmål er utan tvil enklare å uttrykka på naturleg språk enn i noko anna form for databasegrensesnitt. Spesielt gjeld dette spørsmål som inneheld negasjon og kvantifikasjon, som til dømes "*kva for avdelingar har ingen (*eller *ikkje) programmerarar?*" (negasjon) og "*kva selskap leverer til kvar avdeling (*eller *alle avdelingar)?*" (kvantifikasjon)[27]. Spørsmål som dette kan sjølvsagt uttrykkast i databasespråk, som SQL, men då som oftast i form av relativt kompliserte uttrykk. I meny- og skjemabaserte grensesnitt lar det seg ganske enkelt ikkje gjera å bruka innputt av denne typen enkeltvis; Ein må gjerne gå omvegar, det vil seie fordela spørsmålet over fleire innputt, og få ut tilsvarande informasjon på den måten. Karlgren gir eit anna døme på innputt som vert rekna som enklare å uttrykka i naturleg språk, nemleg eit slags komparativt spørsmål som "*Who exports more oil than Norway?*" [op.cit., s. 4]. Når det gjeld BussTUC-systemet finst det òg enkelte typar innputt som ville vore vanskelege å uttrykka i andre typar grensesnitt, nemleg innputt som "*når må eg ta bussen fra Lade for å vere i byen før kl 13?*" og "*når er eg framme ved jernbanestasjonen dersom eg tek buss fra Byåsen kl 14.12?*" Vidare er naturleg språk unikt dersom ein treng å kunna identifisera objekt som ikkje visast på skjermen eller spesifisera temporale relasjonar. Identifisering og handsaming av store sett og subsett av entitetar vert òg enklare. Dessutan har ein i mange former for NLI-system høve til å nytta interaksjonskonteksten, noko andre typar grensesnitt ikkje gjer rom for.

Enkelte NLIDB laga for samanhengande diskurs har den fordelen at dei kan handtera anaforiske og elliptiske uttrykk. Desse systema tillet bruk av svært korte, lite spesifikke spørsmål der tydinga av kvart spørsmål vert komplementært av diskurskonteksten. Andre typar grensesnitt støttar normalt ikkje denne forma for diskurskontekst. BussTUC, slik det er i dag, er i utgangspunktet ikkje eit diskurssystem meint for samanhengande diskurs. Det er likevel mogleg å nytta korte, kommandoliknande innputt der sjølve bruksområdekonteksten – om ein kan kalla han det – spelar inn. Skriv ein til dømes inn eit innputt på forma "*sentrum* –

[27] Døma er fabrikkerte og er ikkje konstruerte ut ifrå noko bestemt NLIDB-system.

lade." tek systemet det for gitt at det er informasjon om bussruter mellom sentrum og Lade brukaren er interessert i og genererer eit svar ut ifrå det, samt gjeldande tidspunkt.

Det finst fleire fordelar med å nytta eit grensesnitt basert på naturleg språk i tilknyting til eit bussruteopplysningssystem som BussTUC: Systemet er rimeleg fleksibelt og enkelt å bruka, noko som er særskilt fordelaktig med tanke på at brukarane er mange og til dels svært ulike, samt at systemet vert brukt relativt lite av kvar enkelt brukar. Brukarane treng lite eller inga opplæring i bruk av systemet før dei kan nytta det. Vidare treng ein ikkje veta verken start- eller endestasjon for dei ulike rutene for å kunne bruka systemet, noko ein gjerne må om ein slår opp i vanlege rutetabellar på papir eller i digital form. Ein treng faktisk talt ikkje veta kva eit einaste busstopp heiter, så lenge ein kjenner namnet på gata eller område ein skal til. Systemet gjenkjenner dei fleste, vanlege stads- og gatenamn i Trondheim og finn næraste stasjon eller busstopp ut ifrå informasjonen lagra i applikasjonsdatabasen.

Napier et. al har samanlikna bruk av eit grensesnitt basert på naturleg språk med bruk av eit menybasert grensesnitt for same domene. I dette studiet kom grensesnittet basert på naturleg språk best ut. Langt fleire brukarar klarte å løysa problema dei vart bedt om å løysa i grensesnittet basert på naturleg språk, og læringsrata verka til å vere høgare for dette grensesnittet enn for det menybaserte. Det finst mange samanlikningsstudium der det motsette er tilfellet, men eit anna og vel så viktig moment i Napier sitt eksperiment var at brukarane var meir nøgde og treivst betre med å bruka grensesnittet basert på naturleg språk. Dette siste poenget er eit viktig eit i høve til dei fleste NLIDB-system. Long slår fast at ein av fordelane med grensesnitt baserte på naturleg språk, generelt sett, er at dei fleste brukarar faktisk liker å bruka dei og er meir nøgde med dei enn med andre typar grensesnitt [Napier i Long -94, s. 5].

Reeves og Nass har forska på interaksjon mellom menneske og media og på sosiale responsar på kommunikasjonsteknologi. Forskinga har blant anna gått ut på å utføra eksperiment der menneske interagerar med ulike media – alt frå ulike programvare til filmklipp. Resultata viser at menneske responderer på ein sosial og naturleg måte i høve til media. Sjølv menneske som hevdar at dei ikkje ser på ein datamaskin som ein sosial aktør, viser seg å respondera som om PC-en var nettopp ein sosial aktør. Reeves og Nass meiner at menneske oppfører seg ovanfor media på same måte som dei oppfører seg ovanfor andre menneske, rett og slett fordi det er enklare å oppføra seg naturleg [The Net Net -03 og Connecticut -96]. Ut ifrå dette burde system baserte på naturleg språkinteraksjon kunna medverka til å gjera det enklare for menneske å interagera med system og maskinar slik det er naturleg for dei å gjera det. For kva er vel meir naturleg enn å nytta eins eige språk for å kommunisera, enten det er andre menneske eller eit datasystem ein interagerer med?

Ei slik oppfordring til brukarane om å oppføra seg naturleg kan likevel ha sine bakdeler, noko enkelte av avsnitta nedanfor skildrar i meir detalj.

5.2 Negative sider ved NLIDB-system

Eit av poenga med grensesnitt baserte på naturleg språk er altså at dei skal vere så og seie sjølvforklarande og enkle å bruka. Som gjenteke ovanfor skal brukarar av NLIDB-system i teorien sleppa å måtta læra noko som helst om databasestruktur eller systemspråk før dei kan ta systemet i bruk. Dette er meint å vere særskilt fordelaktig med tanke på system som først og fremst vert brukte av det ein kan kalla tilfeldige og lågfrekvente brukarar, det vil seie brukarar som nyttar systemet sjeldan og lite. I praksis utartar det heile seg gjerne noko annleis: I og med at ein har med eit avgrensa subsett av aktuelle språk å gjera må tross alt

brukarane læra seg – samt huska – kvar avgrensingane går, det vil seie kva for typar spørsmål eit gitt system kan handtera og ikkje. Dette gjeld i høve til dei reint lingvistiske, så vel som dei konseptuelle eigenskapane til systemet. I somme tilfelle kan dette vere vel så vanskeleg som – og definitivt meir forvirrande enn – å læra seg eit databasespørjingsspråk. Det kan faktisk diskuterast om slike subsett av eit språk som NLIDB-system nyttar eigentleg kan kallast naturleg språk[28].

5.2.1 Usikkerheit i høve til systemkompetanse

Brukarane risikerer å enda opp med feile forventningar i høve til systemet, gjerne i form av falske, positive eller falske, negative forventningar. Androutsopoulos gir døme på dette ut ifrå NLIDB-systemet MASQUE. MASQUE kan svara på det engelske innputtet ”*What are the capitals of the countries bordering the Baltic and bordering Sweden?*”. Ut ifrå dette kan brukar få ei positiv forventning om at systemet òg kan handtera andre innputt innehaldande konjunksjonar. Vidare testing viser at forventninga er falsk: Ei liknande setning som ”*What are the capitals of the countries bordering the Baltic and Sweden?*” går nemleg ikkje gjennom. Når innputt som dette feilar kan brukar lett gå ut ifrå at andre, like ’vanskelege’ spørsmål òg vil feile og difor prøva å unngå desse – brukaren får med andre ord falske, negative forventningar til systemet [Androutsopoulos -95, s.7]. For BussTUC-systemet sin del kan problematikken eksemplifiserast med døma ”*gløshaugen – Byåsen*” og ”*når går neste fra gløshaugen til byåsen?*” der førstnemnde innputt – til tross for at det ikkje er ei fullstendig setning – gir svar, medan sistnemnde (heller ikkje ei fullstendig setning) gir feilmelding. Det er sjeldan opplagt for brukaren kva for lingvistiske eigenskapar eit gitt system har, og ikkje har. Ofte finst det heller ingen dokumentasjon på dette, i alle fall ikkje i form av dokumentasjon som er lett tilgjengeleg for brukarane. Når det gjeld Bussoraklet[29] får brukarane kun informasjon om at dei må nytta fullstendige, norske setningar, at dei bør ha med haldeplassen dei skal frå og til, samt at dei kan nytta gateadresser. I tillegg finst det døme på ein del gyldige innputt. Det finst altså ingen informasjon om eller forklaring på at enkelte, tilsynelatande ugrammatiske innputt faktisk kan handterast av systemet, medan andre ikkje kan det. Det kan vere heller vanskeleg å definera kompetansen til BussTUC-systemet ved å prøva og feila. Når eit innputt som ”*når går buss nr 4 fra lade til sentrum?*” går gjennom skulle ein tru at òg ”*når går buss rute nr 4 fra lade til sentrum?*” ville gå, noko det ikkje gjer.[30]

Det faktum at det ofte er uklart for brukaren kvifor NLIDB-systemet han eller ho brukar ikkje taklar somme spørsmål kan i seg sjølv vere problematisk. Androutsopoulos skiljer mellom det han kallar lingvistiske og konseptuelle feil. Lingvistiske feil oppstår grunna manglar i den lingvistiske dekningsgrada til systemet, det vil seie manglar i grammatikken, morfologidelen eller liknande. Konseptuelle feil oppstår grunna den konseptuelle dekningsgrada [ibid.]. Dersom ein brukar av BussTUC-systemet nyttar uttrykk som systemet ikkje er laga for å kunna handtera, eller spør om noko som ikkje har direkte med buss og bussruter å gjera, finst det som regel ingen informasjon om dette i databasen eller i den semantiske delen av systemet. Systemet kan dermed ikkje svara tilfredsstillande. Dersom brukaren ikkje veit kva

[28] Eg kjem likevel, for ikkje å komplisera det heile meir enn nødvendig, til å fortsetja å bruka omgrepet. ’Naturleg språk’ er i seg sjølv eit noko omdiskutert omgrep. Sjå t.d. Karlgren for ei meir detaljert innføring [Karlgren -92, s.2-4].

[29] Den versjonen av BussTUC-systemet som ligg tilgjengeleg på Team Trafikk sine nettsider (http://www.team-trafikk.no/)

[30] Alle døma i gjeldande avsnitt er døme på innputt som er problematiske i høve til versjon av systemet sist oppdatert i desember -02 . I den nyaste versjonen, det vil seie den som nå er i bruk, fungerer samtlege innputt greit.

for feiltype det er tale om kan det føra til at spørsmål innehaldande konsept systemet ikkje har informasjon om vert omformulerte og prøvde på ny – utan hell – medan spørsmål som systemet i utgangspunktet ville kunna svart på dersom dei hadde vorte formulerte på ein litt annan måte rett og slett vert gitt opp. Sekvensen D1 gjengjeven nedanfor er henta frå feilmeldingsdelen av BussTUC-korpuset. Den er eit typisk døme på eit innputt som ikkje går gjennom, og som vert omformulert og prøvd på nytt utan suksess. I dette tilfellet kan det sjå ut til at brukaren prøver å spesifisera kva buss det er tale om ved hjelp av klokkeslettet. Bussen som går 16.45 vert med andre ord omtala som ”*16.45 bussen*”, noko systemet ikkje er laga for å handtera. Tilbakemeldinga systemet gir er av den meir generelle sorten og gir brukaren få indikasjonar i retning kvar problemet ligg. Brukaren tolkar tydelegvis feilen, som må kunna seiast å vere av den konseptuelle typen, som ein lingvistisk feil og prøver altså på nytt etter å ha endra format på klokkeslettet.

D1 ”*- - - Uforståelig ved * - - -*
*når går neste etter 16 . 45 bussen forbi strindheim til sentrum ? **
*- - - Uforståelig ved * - - -*
*når går neste etter 1645 bussen forbi strindheim til sentrum ? **
*- - - Uforståelig ved * - - -*
*når går neste etter 16 : 45 bussen forbi strindheim til sentrum ? **”

Både dette og til dels problemet nemnt i avsnittet før kan til ei viss grad løysast ved at ein legg til rette for såkalla diagnostiske tilbakemeldingar, det vil seie at systemet gir beskjed om kva som er grunnen til at eit gitt spørsmål ikkje går gjennom. BussTUC-systemet verkar, til tross for dømet ovanfor, til å ha vorte betre òg på dette området i løpet av dei åra det har vore i bruk. Det gir blant anna tilbakemelding dersom brukarar ikkje oppgir kva buss eller haldeplassar det er tale om i innputt der dette er påkrevd for at systemet skal kunna generera eit svar. Som nemnt i avsnittet ovanfor finst det lite spesifikk informasjon om kva BussTUC taklar og ikkje med omsyn til type innputtsetningar. Den typen spesifikke tilbakemeldingar som er skildra her kan likevel seiast å dekka ein del av informasjonsbehovet til brukarane[31]. Dette er kanskje ein vel så bra måte å handtera problemet på som det å leggja ut meir enn det som allereie finst av informasjon om systemet. Det viser seg at dei fleste brukarar les svært lite av denne typen informasjon likevel, spesielt dersom det vert (for) mykje av han; Som oftast føretrekk mange å testa og prøva seg fram i staden.

Ei anna ulempe som kan knytast til systemkompetanse er at eventuelle endringar og forbetringar av eit NLI ikkje vil vere synlege for brukarane. Når ny funksjonalitet vert lagt til ein applikasjon bestående av eit menybasert grensesnitt vert det som oftast lagt til ein ny knapp eller eit nytt menyval som gjer at ein kan aksessera den nye funksjonaliteten. For ein brukar av eit grensesnitt basert på språk opptrer eventuelle endringar i bakgrunnen, utan noko visuell endring. Utan eksplisitt informasjon får brukaren gjerne ikkje med seg endringa i det heile [Coates -02, s.6].

5.2.2 Misforstått intelligens

Språk og intelligens heng nøye saman. At NLIDB-system kan sjå ut til å forstå språk kan òg, i likskap med enkelte av fenomena skildra ovanfor, føra til at brukarane får forventningar til systema som ikkje heilt kan innfriast. Brukarar kan komma i skade for å tru at eit gitt system, i og med at det ’kan’ språk, òg er intelligent på andre måtar – at det har ei form for sunn fornuft – at det kan trekka slutningar og liknande. Long omtalar fenomenet, som er eit relativt

[31] Sjå appendiks B for ei oversikt over denne og andre typar tilbakemeldingar.

utbreitt eit når det gjeld system baserte på naturleg språk, som ei form for antropomorfisme[32] [Long-94, s.2]. På nettsidene til Team Trafikk står det lite og ingenting om kva Bussorakelet faktisk er. Enkelte brukarar verkar til å tru at det er ein person som sit og svarar på spørsmåla dei tastar inn. Innputt som *"hvor mange bussruter kan du i hodet?"*, *"hvor er du i fra?"*, *"er det gøy å jobbe i team trafikk?"* og *"er det masse spørsmål til dere?"*, henta frå feilmeldingsdelen av korpuset, kan i alle fall tyda på det. Dette ser vel og merka ut til å vere ei rein misforståing og ikkje nødvendigvis feile forventningar. Vidare finst det nok av døme på at brukarar forventar at systemet veit, det vil seie har informasjon om, langt meir enn det faktisk gjer / har. *"hvilken pris er det for månedskort for studenter orkanger trondheim?"*, *"hvor snart må man bruke en overgangsbillett?"* og *"hvor lenge har kontoret åpent på lørdag?"* er typiske døme. I tillegg finn ein døme på innputt så dårleg formulerte at sjølv eit menneskeleg orakel ville hatt problem med å tyda dei: *"breidablikk byåsen hegdalen bergheim terrasse må være ved hegdalen til kl 2000?"*, *"buss fra byen til lundåsen i dag før kl 18 passerer lerkendal når?"*. Meir om dette i neste avsnitt.

5.2.3 Naturleg språk som medium

I vitskapsfilosofien kan ein lesa om Leibniz og hans visjonar om eit eige eintydig og universelt språk innan vitskapen, 'characteristics universalis'. Leibniz ønska å uttrykka alle vitskaplege sanningar og påstandar i ei form for symbollogikk og utvikla eit resonneringskalkulus, 'calculus ratiocinator', meint for manipulering av dei logiske uttrykka. Leibniz sitt arbeid medverka i stor grad til innføringa av logikk som matematisk disiplin, men draumen om det ideelle språk vart aldri eigentleg ein realitet. Omsetjinga av alle vitskapelege utsegner – nedskrivne på diverse naturlege språk – til logikk skulle visa seg umogleg å gjennomføra. Mange ser på det å nytta naturleg språk i menneske-maskin-interaksjon som like umogleg. Dei meiner naturleg språk er for komplekst og fleirtydig, og enkelte vil altså gå så langt som til å hevda at språk er direkte ueigna som medium i høve til det å kommunisera med databasesystem og datasystem generelt. Som me har sett tidleare er eit NLIDB-system avhengig av å kunna tolka og omsetja innputtsetningar til ei eller anna form for eintydige uttrykk, som logikkuttrykk, for å kunna henta ut informasjonen det er tale om frå databasen. Det seier seg sjølv at dette ikkje er ei direkte problemfri oppgåve dersom fleire av innputta inneheld fleirtydige konstituentar eller er fleirtydige som heilskap. Ambiguitet er då òg – som nemnt, blant anna i kapittel 3 – eit av hovudproblemområda og ei av dei største utfordringane NLP-feltet står ovanfor.

Samanlikna med skjema-, meny- og grafikkbaserte grensesnitt, der det held å fylla inn nokre få felt eller klikka seg fram ved hjelp av musa, kan bruk av grensesnitt baserte på naturleg språk i mange tilfelle framstå som tungvint og omstendeleg. Karlgren gjengir eit døme på ei setning, henta frå Bretan, som dei fleste – sjølv dei med engelsk som morsmål – normalt har problem med å formulera og til dels forstå: *"Show me the area and population of all the countries that are members of NATO and which do not import any Volvo cars."* [Bretan i Karlgren, s.4] Det varierer sjølvsagt frå person til person kor vidt denne typen setningar faktisk representerer eit problem, men for eit offentleg tilgjengeleg system som BussTUC gjeld det å ta omsyn til og leggja til rette for flest mogleg brukarar og typar brukar. Ein finn nok av døme på det som må kunna klassifiserast som formuleringsproblem i høve til BussTUC-systemet. Ved gjennomgangen av feilinnputta i korpus kjem det tydeleg fram at ein del brukarar i somme tilfelle slit med å få pressa inn alt av informasjon som trengs for å få generert eit relevant svar frå systemet i eitt og same innputt. Dette ser ut til å gje utslag på ein av to måtar: Enten vert innputta av den særskilt korte og ofte ugrammatiske sorten – skrivne i

[32] Det å projisera menneskelege eigenskapar over på ikkje-menneskelege objekt

ein slags telegram- eller kommandoaktig stil – eller så vert dei særskilt lange og komplekse, ofte beståande av meir enn ei setning. Enkelte innputt, spesielt av den sistenemnte typen, er skrivne i ein stil som kan minna om munnleg stil. Dei er noko usamanhengande og dårleg strukturerte, med dei enkelte informasjonsbitane stokka om einannan og syntaktiske konstituentar innskotne i hytt og vèr. Somme typiske døme:

D2 ”*når går bussen neste buss til vestlia?*”
D3 ”*når buss fra lade til sentrum neste går?*”
D4 ”*når går neste buss etter klokken 4 nr 9 fra voll?*”
D5 ”*når går neste etter klokken 1550 buss nummer 20 fra tyholt til sentrum?*”
D6 ”*jeg skal fra st olavs hospital og til persaunet hvilken rute bruker jeg da ?*”
D7 ”*når klokken er 700 og jeg må på jobb klokken 800 fra kattem når bør jeg gå ?*”
D8 ”*hei jeg lurer på når bussen går fra ila til sentrum jeg skulle vært der senest klokka 23.0.*”

I somme tilfelle, som i D9 og D10 nedanfor, vert så og seie same syntaks brukt om att, til tross for at systemet gir feilmelding på første forsøk:

D9 ”*- - - Uforståelig ved * - - -*
*hvordan kommer man fra scrøder til sjøveien 20 etter 1800 skal * bytte buss i munkegaten.*
*- - - Uforståelig ved * - - -*
*hvordan kommer man fra scrøder til sjøveien 20 etter 1800 bytte buss i munkegaten.**”

D10 ”*- - - Uforståelig ved * - - -*
*når 2 går neste bussene fra moholt til gløs ? **
*- - - Uforståelig ved * - - -*
*når de to går neste bussene fra moholt til gløs ? **
*- - - Uforståelig ved * - - -*
*når de 2 går neste bussene fra moholt til gløs ? **”

Det at brukarar uttrykker seg svært ulikt kan i seg sjølv vere problematisk med omsyn til system som BussTUC. Eitt og same spørsmål kan formulerast som ein lang, omstendeleg setning, eller som ein kort, meir eller mindre konsis kommando. Igjen er spørsmålet kor grensa skal gå. Skal ein nøya seg med å gjera systemet i stand til å handtera kun dei vanlegaste innputt-typane, eller bør målet vere å dekka mest mogleg av variasjonen? Oppdeling av ord og bruk av spesialteikn som bindestrek og skråstrek og anna er ting som kan vere med å vanskeleggjera parsingprosessen ytterlegare. Stadig fleire norske språkbukarar har ein tendens til å anglifisera norsken sin, særleg når det gjeld samansette ord. Samansette ord som normalt skal skrivast i eitt på norsk vert ofte skrivne som to separate ord eller, i somme tilfelle, med bindestrek. Innputtet ”*når går neste buss fra ringve skole vei 3 til sentrum?*”, der ordet *skolevei* er delt opp i *skole* og *vei*, eksemplifiserer fenomenet. Problemet med spesialteikn, som òg vart teke opp i kapittel 3, er at dei ofte kan tyda ulike ting i ulike kontekstar. Dessutan nyttar folk dei relativt ulikt og i ulike samanhengar, noko som sjølvsagt gjer det ekstra komplisert å skulla fastslå kva som eigentleg er meint i kvart enkelt tilfelle. Døma nedanfor, to innputt innehaldande skråstrek og to kolon, viser dette.

D11 ”*når går buss 66 brundalen / jakobsli torsdag 600 700?*”
”*når går bussen fra strandveikaia til flatåsen / kolstad etter kl 19 . 0?*”

D12 *"hvilke avganger finnes det i kveld fra : dalen hageby til bakke bru?"*
"når går neste 5 busser til thønning owesens fra sentrum etter kokken 8 : 0?"

Problemet med innputt som dette er at mange av dei, som setningane i D11, vert ugrammatiske òg når systemet ignorerer spesialteikna.

5.3 Ulike feiltypar

Véronis nyttar distinksjonane Chomsky innførte, nemleg performans (performance) og kompetanse (competence), til å kategorisera ulike feiltypar som kan oppstå ved bruk av NLP-system [Véronis i Karlgren -92, s.4-5]. Typologien er henta frå Karlgren og vist i tabell 5.1 nedanfor. Den gir ein god oversikt over dei fleste feiltypane som kan førekomma med omsyn til NLP-systeminteraksjon. I praksis kan det by på problem å skulla plassera ulike feilmeldingsinnputt innanfor rett kategori, mykje fordi typologien er såpass spesifikk. Utan eventuelt å spørja brukaren direkte kan ein i somme tilfelle ikkje veta sikkert om eit innputt innehaldande stavefeil, til dømes, bør kategoriserast som ein performans- eller kompetansefeil. Stavefeil kan enten skuldast at brukaren tastar feil eller bommer på ein tastaturtast (performansfeil) eller at brukaren har feil oppfatning av korleis det aktuelle ordet skal stavast (kompetansefeil). Det er likevel mogleg å finna klare døme på dei ulike feiltypane, samt å fastslå – med ein viss sannsynlegheit – kva for feiltype / -typar det finst flest av i dei ulike kategoriane laga ut ifrå feilmeldingskorpuset skildra i kapittel 2.

	Lexical	**Syntactic**	**Semantic**
System performance	- Letter substitution - Letter insertion - Letter deletion - Sub-string substitution		
System competence	- Word missing in dictionary - Missing inflection rule	- Missing rule	- Incomplete knowledge representation - Missing inference rule - Unexpected situation
User performance	- Letter substitution - Letter insertion - Letter deletion - Letter transportation - Sub-string substitution - Syllabic error	- Word substitution - Word insertion - Word deletion - Word transportation	
User competence	- Grapheme substitution - Incorrect segmentation - Wrong inflexion - Non-word or completely garbled word	- Wrong agreement - Homophone - Punctuation error - Rule violation	- Conceptual error - Presupposition violation - Reasoning error - Dialogue law violation - Scenaric or script violation

Tabell 5.1: Véronis si klassifisering av feil som kan oppstå når menneske interagerer med datamaskinar og –system på naturleg språk.

Dei aller fleste av feila i feilmeldingsinnputta tilhøyrande den sjølvlaga kategorien 'Feilstavingar' verkar til å vere av typen performansfeil. *"når går neste 5 busser fra sentrumg syd til ikea ?*" er eit typisk døme på ein brukarperformansfeil på leksikalsk nivå (ein bokstav for mykje, altså 'Letter insertion'-kategorien i tabellen ovanfor). Det neste dømet, derimot, kan tyda på å vere ein kompetansefeil (òg på leksikalsk nivå):

*"- - - Uforståelig ved * - - -*
*når må jeg ta buss 7 fra studentersamfunnet for å vare * på flatåstoppen kl 1130?*
*- - - Uforståelig ved * - - -*
*når må jeg ta buss 7 fra studentersamfunnet for å vare * på flatåsen kl 1130 ?*
*- - - Uforståelig ved * - - -*
*når går buss 7 fra studentersamfunnet for å vare * på flatåsen kl 1130?*"

I og med at ordet som er markert som feil i første innputt ikkje vert retta i dei påfølgjande, liknande innputta kan det tenkjast at brukaren har feil oppfatning av korleis det aktuelle ordet skal stavast.

Som oversikta i kapittel 2 viser består den største av dei gruppene feilmeldingsinnputt det der vert operert med av innputt innehaldande ord eller teikn som ikkje finst i systemet. Ein del av desse orda og teikna ser ut til å verta brukte veldig sporadisk og dukkar altså opp i korpus kun nokre få gonger. Med unntak av namn på haldeplassar og aktuelle stadar burde ikkje behovet for å leggja til denne typen lågfrekvente ord vere særleg stort. Andre ord og teikn vert brukte noko meir og dukkar stadig opp i korpus. Desse burde, så fremt det let seg gjera på ein rimeleg enkel og grei måte, leggjast til. Spesielt når det gjeld teikn er det ikkje alltid like problemfritt å utvida systemet på denne måten. Som nemnt i avsnitt 5.2.3 ovanfor kan teikn som bindestrek, skråstrek og andre nyttast i fleire ulike samanhengar, med tildels ulik tyding. Som nemnt tidlegare er det ikkje alltid at fordelane forbunde med å leggja til spesialteikn veg opp for ulempene det kan medføra. Som oftast er feila tilhøyrande kategorien 'Manglande ord og teikn' kompetansefeil, men om dei skal reknast som systemkompetansefeil på eit leksikalsk nivå (feil på grunn av at ord manglar i systemordbok) eller som brukarkompetansefeil på eit semantisk nivå (konseptuell feil eller eventuelt feil som følgje av feile antakingar) kjem altså heilt an på kva for ord eller teikn det er tale om, kor ofte brukarar nyttar det enkelte ordet eller teiknet, kor avgjerande det er at det vert teke med, osb. Det er, som nemnt tidlegare, eit vurderingsspørsmål kvar grensa for kva som bør leggjast til og ikkje skal gå. I somme tilfelle kan det svara seg, i staden for å utvida systemet, å heller satsa på å læra brukaren meir om kva teikn og uttrykk som kan brukast og ikkje (sjå avsnitt 5.4 nedanfor). Det meste av det som er skildra i dette avsnittet gjeld òg i høve til feiltypane som finst i kategorien 'Forkortingar og dialekt' og til dels i 'Irrelevant'; Igjen er det av stor tyding at brukarane får klare grenser å halda seg til i høve til kva som er mogleg og ikkje. Når det gjeld kategorien 'Irrelevant' fell svært mange av innputta utanfor heile typologiskjemaet, ganske enkelt fordi innhaldet ikkje har noko med bruksområdet til BussTUC-systemet å gjera.

Feila gjort i høve til feilmeldingsinnputta i kategorien 'Ugrammatisk / ufullstendig' høyrer som oftast til i den syntaktiske kolonna i typologiskjemaet og verker til å vere av typen brukarperformans- eller brukarkompetansefeil. *"når går må * jeg ta bussen fra kattem for å være i sentrum kl 1900?*" og D2 ovanfor er typiske døme på brukarperformansfeil (eit ord for mykje), medan D9 og D10 ovanfor kan tyda på å vere brukarkompetansefeil, enten av den syntaktiske (brot på regel) eller eventuelt den semantiske sorten (konseptuell feil eller antakingsfeil). Døma frå og med D3 til og med D5 kan vere både performans- og

kompetansefeil, avhengig av om brukar har vore bevisst syntaksen han har brukt, eller ikkje.

5.4 Forslag til forbetringar

Det seier seg sjølv at enkelte feil er enklare å redusera enn andre. Systemperformans og -kompetanse er det som oftast mogleg å forbetra – i alle fall på det leksikalse og syntaktiske nivået – i form av å utbetra eller leggja til reglar og ord. Kategorien 'Manglande ord og teikn' utgjer, som nemnt, ei av det største gruppene feilmeldingsinnputt. Ved å leggja til rette for bruk av dei orda som opptrer hyppigast kan ein dermed oppnå ei relativt stor forbetring på ein rimeleg enkel måte. To av forskarane ved Hewlett-Packard sine laboratorier, Whittaker og Stenton, påpeiker det same i sin artikkel om brukerstudier og design av system baserte på naturleg språk: Dei fleste feila som oppstår i tilknyting til systemet dei evaluerer skuldast ord som ikkje er lagt til i ordlistene til systemet. Somme av dei manglande orda som brukarane nyttar viser seg å vere synonym til ord som allereie finst i systemet og kan leggjast til som nettopp det [Whittaker -89, s.121]. I tilknyting til BussTUC har til dømes ordet *hvis* vist seg å vere problematisk. Eit innputt som "*hvilken buss passer best å ta fra Fiolsvingen hvis jeg skal være på torvet klokken 20?*" får feilmeldinga *Uforståelig ved* * med asteriksen plassert ved sida av nettopp *hvis* i den utgåva av systemet som er utgangspunkt for denne oppgåva. Innputtet går gjennom så snart ein byter ut *hvis* med *når*. Problemet med *hvis* burde med andre ord kunna løysast ved å definera *hvis* som eit slags synonym til *når*.

Når det gjeld brukarperformans- og -kompetansefeil, som ein stor del av døma ovanfor må kunna karakteriserast som, vert det straks vanskelegare. Ein del av feila på det leksikalske nivået burde det vere mogleg å retta opp ved hjelp av ein betre stavekontroll. Somme av feila tilhøyrande det syntaktiske nivået kan ein unngå ved å opna for større variasjon med omsyn til setningskonstruksjon. Kategorien 'Ugrammatisk / ufullstendig' inneheld til dømes ein del innputt av typen "*når går de to neste forbi einar tambarskjelves gt og til prof brochs gt?*" der subjektet, det vil seie ordet *buss*, er utelete. I og med at det er eit system for bussruteopplysning det her er tale om, burde ein nok tillata denne typen innputt og sørgja for at ordet *neste* kan brukast som eit slags subjekt aleine, på lik linje med *neste buss/-er/-ene*.

Dei resterande feiltypane krev løysingar av eit noko annan slag. Karlgren påpeikar at mange kompetansefeil oppstår på grunn av at det er skilnad mellom brukar- og systemkompetanse. I staden for å retta performansfeil burde hovudfokuset til systemet i slike situasjonar vere å visa, og dermed læra, brukaren kva kompetanse det har. For å oppnå dette må språket systemet nyttar, i alle feil- og tilbakemeldingar, vere vennleg og lett å læra, det vil seie det må vere konsekvent og koherent og avgrensingane bør enkelt kunna oppdagast ved prøving og feiling. Karlgren meiner ein bør utnytta kunnskap om naturleg, menneskeleg oppførsel for å laga betre NLIDB-system. Han skildrar blant anna det som vert kalla 'the Audience Design Principle' på engelsk, prinsippet om at menneske tilpassar interaksjonen ut ifrå kva kunnskap og evne motparten har, eller ser ut til å ha. Diskurs er i stor grad eit samarbeidsprosjekt basert på felles forståing, og tilbakemeldingane kommunikasjonspartane gir einannan er med på å finstilla partane si oppfatning av einannan. Kommunikasjonssituasjonen verkar til å vere avhengig av kor like partane er, eller syns dei er, samt kor godt dei kjenner einannan. I følgje Karlgren er dette trekk som bør takast omsyn til og som bør kunna dragast nytte av i utviklinga av system baserte på interaksjon på naturleg språk.

5.4.1 Menneske-maskin-interaksjon

Det er eit kjent faktum at menneske oppfører seg noko ulikt etter kven eller kva dei kommuniserer med, samt kva slags kanal og modalitet som vert brukt. Det finst mange ulike studium som alle konkluderer med det same: Det finst utan tvil vesentlege forskjellar mellom

dei ulike interaksjonsformene. Det er forskjell på menneske-menneske-interaksjon og menneske-maskin-interaksjon, sjølv når den menneskelege parten har høve til å nytta sitt eige språk i begge interaksjonsformer. Skriftleg interaksjon er ulik munnleg interaksjon og det finst skilnadar mellom meir eller mindre interaktiv interaksjon. Det er innlysande, så ulik mennesket sjølv som datamaskinen er, at menneske-maskin-interaksjon skiljer seg frå interaksjon menneske imellom på mange måtar. Naturleg nok har menneske ganske andre forventningar og haldningar til ein maskin enn til andre menneske, noko som set sitt preg på interaksjonen. Menneske-maskin-interaksjon der mennesket kommuniserer med maskinen ved hjelp av eige språk representerer dessutan ein relativt ny type kommunikasjonsmodalitet, ein faktor som i seg sjølv påverkar interaksjonen. Det normalt så markante skiljet mellom munnleg og skriftleg interaksjon verkar på mange måtar mindre skarpt sett i høve til denne forma for interaksjon. Enkelte NLIDB-innputt er skrivne i ein stil som kan minna om munnleg stil. Stilen brukarar av NLIDB-system vel kan òg sjå ut til – både når det gjeld ordval og syntaks – å vere prega av liknande kommunikasjonskanalar, som chat, e-post, SMS, med fleire. Somme av døma framstilte ovanfor viser litt av dette. Fleire forskarar, både før og etter Karlgren, har hevda at brukarar av NLP-system, og då særskilt nye brukarar, nyttar eit enklare språk ovanfor maskinen enn dei vanlegvis ville ovanfor ein menneskeleg kommunikasjonspartner. Karlgren siterer ein av forskarane, Bozena Thompson[33]: ”Monotony of structure is the rule rather than the exception in human-computer interaction.” [Thompson i Karlgren -92, s. 9].

Brukarar verkar til å vere meir forsiktige når dei interagerer med datasystem og -maskinar. Dei har ein tendens til å nytta kun deler av den lingvistiske uttrykksfridomen dei har tilgjengeleg og prøver gjerne å unngå bruk av pronomen og liknande, i den tru at systemet ikkje kan handtera dette. Det kan vere eit problem i høve til enkelte system at brukarane nyttar ein enklare og meir einsarta syntaks enn det dei faktisk har høve til å nytta. Systemet Karlgren bruker i sine eksperiment er i stand til å handtera samanhengande diskurs og innputt med pronomen, noko som er fordelaktig og praktisk – så lenge brukarane faktisk oppdagar at det lar seg gjera. Karlgren rapporterer at kun nokre få av eksperimentdeltakarane hans utnytta denne systemeigenskapen, og dei få som gjorde det gjorde det etter først å ha spurt ein av rådgjevarane tilstades under eksperimentet. Høflegheitsformer, som indirekte talehandlingar, vert typisk uteletne når menneske interagerer med datamaskinar. Vidare finn ein langt fleire brå skift i dialogretninga i menneske-maskin-interaksjon enn ein vanlegvis gjer i andre interaksjonsformer.

Sjølvsagt finst det stor variasjon innanfor menneske-maskin-kommunikasjon òg. Det finst mange faktorar som speler inn. Forskjellen på munnleg og skriftleg interaksjon gjer seg gjeldande, her som for andre kommunikasjonstypar, om enn ikkje i like stor grad. Ein annan viktig faktor er at kvar enkelt systembrukar er forskjellig. Deltakarane i Karlgren sitt eksperiment er av ulike kjønn og aldrar og har ulik bakgrunn. Somme av deltakarane er erfarne med omsyn til bruk av datamaskinar og ulik programvare, og enkelte av desse igjen har brukt databasesystem før og kjenner til formelle databasespørjingsspråk. Det kjem tydeleg fram at desse deltakarane prøver å nytta den kunnskapen og erfaringa dei sit inne med; Ein del av innputta desse brukarane gav inn til systemet har mykje til felles med vanlege SQL-uttrykk. Som oftast skiljer meir datakyndige brukarar seg òg ut i form av å ha for låge forventningar til NLIDB-system, medan brukarar som er meir uerfarne med omsyn til bruk av datamaskinar og -system ofte viser seg å ha for høge forventningar i høve til kva systemet er i

[33] Thompson har blant anna arbeidd med NLIDB-systemet ASK (A Simple Knowledgeable system)

stand til. Felles for dei aller fleste av deltakarane er likevel at dei har lita erfaring og er lite vane med denne typen interaksjon og med bruk av grensesnitt baserte på naturleg språk.

5.4.2 Likskapar mellom interaksjonsformene

Forskjellar og variasjon finst det altså, både mellom menneske-menneske-interaksjon og menneske-maskin-interaksjon og innanfor kvart av felta. Det finst like fullt enkelte fellestrekk mellom interaksjonsformene, fellestrekk som altså kan visa seg å vere nyttige å ta til etterretning med omsyn til det å utvikla gode NLIDB- og NLP-system generelt sett. Ut ifrå sine studium meiner Karlgren å finna sterke indikasjonar på at brukarar av NLIDB-system prøver å etterlikna språket systema nyttar og "ser ut til å føretrekka", som ein av deltakarane på Karlgren sitt prosjekt uttrykte det. Ein menneskeleg kommunikator vil i følgje Karlgren, enten det er ein menneske-menneske- eller menneske-maskin-kommunikasjonsituasjon det er tale om, heile tida sjå etter teikn på kva type interaksjon motparten ønskjer eller støttar og prøva å tilpassa seg kommunikasjonspartnaren. Eit anna fellestrekk er at kommunikasjonen endrar seg ettersom den menneskelege kommunikasjonsparten prøver å tilpassa seg. Dette er nok eit viktig poeng i tilknyting til utvikling av NLP-system: Ein utviklar bør spørja seg korleis systemet kan medvirka til at denne endringa resulterer i ein mest mogleg funksjonell og effektiv kommunikasjon. Målet bør vere å byggja system som naturleg hjelper brukarar til å handla og oppføra seg på ein formålsteneleg og funksjonell måte. Igjen bør ein altså, i følgje Karlgren, venda seg til psykologien og til studium av menneskeleg oppførsel for å komma nærare eit slikt mål [Karlgren -92].

Andre forskarar har ei noko anna oppfatning når det gjeld akkurat dette. Shneiderman meiner at grensesnitt baserte på naturleg språk som einaste form for interaksjon ikkje utnyttar eigenskapane til datamaskinen og at menneske-menneske-interaksjon difor ikkje nødvendigvis er ein passande modell for menneske-maskin-interaksjon. Han påpeiker at ein datamaskin kan prosessera og framstilla informasjon svært mykje raskare enn ein brukar kan skriva inn kommandoar eller innputt. Med tanke på dette verkar det mest fordelaktig å nytta maskinen til nettopp å framstilla store mengder informasjon og tillata at vanlege brukarar, det vil seie ikkje-ekspertbrukarar, kan utføra sine oppgåver ved ganske enkelt å velja mellom objekta som vert framstilte [Shneiderman -98, kap. 8.7.1]. På den måten kjem det i tillegg klart fram kva for funksjonar som finst tilgjengelege, ein eigenskap òg Coates nemner at ofte manglar i tilknyting til NLI (sjå avsnitt 5.2.1 ovanfor). Shneiderman avskriv likevel ikkje grensesnitt baserte på naturleg språk heilt og fullt. Han meiner interaksjon ved hjelp av naturleg språk kan ha sine fordelar og kan fungera dersom det vert brukt til rett bruksområde og passande oppgåver, samt av rette typen brukarar.

Systemet Karlgren arbeidde med har ein særskilt eigenskap som ser ut til å tyda mykje med omsyn til læreprosessen systembrukarane gjennomgår under interaksjonen: Det nyttar parafraser. Etter analyse og parsing repeterer systemet innputt og gir brukaren valet mellom å be systemet halda fram eller eventuelt å forsøka på ny før systemet så går vidare med henting av informasjon frå databasen og generering av endeleg utputt. Parafrasane er eintydige og skrivne på ein enkel og konsekvent måte. Dei representerer på mange måtar språket til systemet, det vil seia språket det er mest fordelaktig å nytta når ein interagerer med systemet. Det kjem tydeleg fram av innputt deltakarane på Karlgren sitt prosjekt gav inn til systemet at dei vart prega og tok lærdom av parafrasane. Denne systemeigenskapen var i utgangspunktet ikkje tiltenkt ein slik funksjon, men viste seg altså å vere eit nyttig ledd i ein slags interaktiv brukaropplæringsprosess. Eit liknande opplegg burde òg kunna nyttast med omsyn til fleirtydige innputt: Systemet burde kunna presentera alle dei aktuelle tydingane det måtte finna for brukaren og la brukaren velja kva for ei av dei det skal gå vidare med.

5.4.3 Hjelpeprogram og valmoglegheiter

Deltakarane i prosjekteet til Karlgren hadde tilgang til ein prototyp på eit hjelpeprogram tilhøyrande systemet der dei blant anna fekk oversikt over vokabularet og kunne finna og testa ut ulike konsept og relasjonar. Somme brukarar nytta dette relativt mykje, gjerne før kvar interaksjonssekvens. Andre brukte ikkje hjelpeprogrammet i det heile, men føretrekte å bruka meir tid på prøving og feiling. Dette er bare eitt av mange døme på at ulike brukarar føretrekk ulike ting og gjer ting på ulike måtar. For system som BussTUC er det, som nemnt tidlegare, viktig å leggja til rette for flest mogleg brukarar. Av den grunn burde kanskje ei hjelpefil eller eit hjelpeprogram som inneheld litt meir informasjon om systemet vere tilgjengeleg (frå nettsida), for dei som måtte ønskja det. Karlgren understrekar at dei fleste brukarar, til tross for intensjonen om at NLIDB-system skal vere sjølvforklarande og intuitive med omsyn til bruk, faktisk treng ei eller anna form for opplæring. Ein kan ikkje gå ut ifrå at grensesnittet, om det framstår aldri så enkelt, faktisk verkar intuitivt og enkelt for kvar og ein brukar.

Det er eit stort pluss om brukarane har høve til å velja mellom ulike grensesnitt og interaksjonsformer, etter kva dei føretrekk og kva type innputt det er tale om. På denne måten bøter ein på ein stor del av dei negative aspekta skildra i føregåande del. Dette tek ein nå høgde for i tilknyting til Bussoraklet. Den versjonen av BussTUC som ligg tilgjengeleg via Team Trafikk sine sider består nå av to deler – eit avansert og eit enkelt orakel. Den avanserte delen – det vil seie delen basert på innputt i form av norske og engelske setningar og som denne oppgåva er laga ut ifrå – er den same som tidlegare, medan den enkle delen består av felt der ein enten fyller ut ved hjelp av ein meny eller ved å tasta inn manuelt. Ved å nytta den enkle delen unngår ein dei fleste samanbrot som skuldast stavefeil eller syntaktiske feil som gal konstituentrekkefølgje. Den avanserte delen kan nyttast til innputt som ikkje let seg uttrykka i det heller avgrensa, enkle oraklet.

5.4.4 Tilkjennegjering og oppretting av feile forventningar

Som nemnt i innleiingskapitlet til denne oppgåva kommenterer McKeown behovet mange brukarar har for å gjera seg meir kjent med databasen og systemet før dei går i gang med å spørja meir spesifikke spørsmål av den sorten systemet opphaveleg er laga for å svara på [McKeown -83, s. 10]. Dette, viser det seg, er nok eit typisk trekk ved menneskeleg oppførsel og interaksjon og kan vere viktig å leggja til rette for i NLIDB-system. Det finst nok av døme på denne typen spørsmål og testing i tilknyting til BussTUC-systemet. Så lenge nettversjonen av systemet kan brukast gratis vert han sjølvsagt – som kapittel 2 seier litt om – utsett for alle moglege former for irrelevante innputt. Ut ifrå det som finst innsamla av korpusmateriale er det tydeleg at ein del brukarar ser på bruken og det ein til nøds kan kalla testing av systemet som ei form for underhaldning. Brukarar spør om alt frå ”*hva er meningen med livet?*” til ”*hvor syk er jeg?*”. Samstundes finst det enkelte, tilsynelatande irrelevante innputt – somme i form av det som gjerne vert kalla metakunnskapsspørsmål (spørsmål om metakunnskap, det vil seie kunnskap om kunnskap) – som det kan vere verdt å merka seg og eventuelt leggja til rette for å kunna svara på, på eit eller anna vis. Dette gjeld spørsmål såpass banale som ”*hva heter du?*”, ”*hvor gammel er du?*”, ”*hva har du informasjon om?*” og ”*hva er 1+1?*”. Eit system og såkalla orakel som ikkje veit sitt eige namn, sin eigen alder eller kva 1+1 er, oppnår ikkje nødvendigvis noko stor tillit hjå brukarane. Systemet bør i det minste kunna gje ei tilbakemelding i form av ei passande feilmelding. Samstundes er det ikkje meininga at Turing-testen må vere bestått for at systemet skal vere bra til det det er meint å skulla brukast til. Det finst sjølvsagt grenser for kva ein kan leggja til av informasjon, spesielt av typen som har lite med sjølve bruksområdet å gjera. Igjen må ein finna ein balansegang når det gjeld kva systemet bør vere i stand til å svara på og ikkje. BussTUC-systemet ser ut til å ha vorte betre òg på dette området. Det er tilrettelagt for, og kan svara på, langt fleire av denne typen

spørsmål nå enn tidlegare, og nyttar i større grad passande tilbakemeldingar i tilfelle der det ikkje kan generera noko meir spesifikt svar ut ifrå databasen. Dette er utan tvil gjort vel så mykje for sporten si skuld som for noko anna, men kan altså ha noko å seie med omsyn til brukarane sine haldningar til systemet. Det kan vere av tyding at systemet framstår – òg på andre, grunnleggjande område enn det det hovudsakleg er laga for – som eit 'intelligent' system.

Når det gjeld feile forventningar poengterer McKeown at brukarar ofte avslører, indirekte så vel som direkte, eventuelle feile antakingar om eit gitt system, eller databasen til eit gitt system, gjennom sine innputt til systemet. Ho ser på innlemming av antakingar i ytringar som eit formelt trekk ved naturleg språk, eit trekk som kan utnyttast for å komma fram til og eventuelt korrigera misoppfatningane brukarar måtte ha [Ibid.].[34]

5.5 Oppsummering og avsluttande kommentar

Eit NLIDB-system bør framstå som intelligent i den forstand at brukarane har tiltru til det og ser på det som nyttig og brukbart til det det er meint å nyttast til. Samstundes bør det komma tydeleg fram at det faktisk er eit datasystem og at det difor finst avgrensingar. Brukarane må – helst gjennom sjølve interaksjonen med systemet – få høve til å læra seg kva kompetanse systemet har, både i form av reint lingvistiske eigenskapar og i form av konseptuell dekningsgrad. Eventuelle parafraser og systemutputt generelt sett bør vere nøye gjennomtenkte og koherente med omsyn til språk og nyttast bevisst til å styra læreprosessen til brukarane. Feil- og tilbakemeldingar bør elles vere diagnostiske, det vil seie mest mogleg forklarande og spesifikke i høve til det gjeldande innputtet. Long siterer Véronis som oppsummerer det heile slik: "If the system cannot adapt to the user, the user should be able to adapt to the system."[Véronis i Long -94, s. 5]. Brukarane bør, så sant det let seg gjera, få valmoglegheiter og alternativ med omsyn til interaksjonsform og hjelpefunksjonar.

Mykje av det som er nemnt ovanfor med omsyn til forbetringar er vel og merka mest aktuelt i førekant av ei systemutvikling. Korleis systemet kan konstruerast for å unngå samanbrot og oppnå ein mest mogleg effektiv og fruktbar interaksjon bør vere noko som er tenkt grundig gjennom før ein startar på sjølve utviklingsarbeidet. Blant anna bør det vere avgjort kor grensene skal gå med omsyn til språket som skal nyttast, gjerne på grunnlag av tidlegare erfaringar og eventuelt oppsamla døme på faktiske inn- og utputt, som korpuset tilhøyrande BussTUC-systemet. Eit system som BussTUC, i bruk og langt på veg ferdig, er ikkje like enkelt å forbetra på eit så grunnleggjande plan. BussTUC er vel og merka eit slags prøve- og forskingsprosjekt der utviklinga har gått føre seg gradvis og over lang tid. Ein har i stor grad gått ut ifrå dei faktiske inn- og utputta etterkvart som desse har vorte samla inn, samt tilbakemeldingar elles, og utbetra og tilpassa systemet litt etter litt i høve til desse. Systemet har vorte merkbart betre i løpet av den tida det har vore i bruk og har, utan tvil, sørgja for å gje utviklarane ny, verdifull kunnskap og erfaring undervegs.

[34] McKeown viser blant andre til Mays (1980): *Correcting misconceptions about database structure*

6. Konklusjon

Denne avsluttande delen startar med ei kort oppsummering av hovudpunkta i oppgåva. Deretter følgjer ein diskusjons- og konklusjonsdel som tek føre seg resultata som er komne fram til, samt ein del av dei generelle aspekta ved NLIDB-system som er vorte nemnde i dei tidlegare kapitla. Siste, og avsluttande, avsnitt tek føre seg gjenståande oppgåver og utsiktene for NLI og NLIDB fram i tid.

Det føregåande har vore eit forsøk på å avdekka og søka løysingar på, i første rekke, ambiguitetsrelaterte problem knytta til BussTUC-systemet. Ulike løysingsmetodar har vorte presenterte og somme prøvde ut. Det har vorte vist at tillegg og / eller endringar i grammatikken og tilhøyrande deler av systemet kan løysa enkelte av dei gjenståande problema. I somme tilfelle må andre, meir generelle løysingstilnærmingar til, noko kapitlet 'Brukarstudium og bruksanalyse' er med på å demonstrera. Eit av spørsmåla oppgåva har søkt svar på er om bruk av ein kartparsingsmetode kan medverka til eit betre system med omsyn til disambiguering.

Eitt av spørsmåla som stadig dukkar opp i tilknyting til BussTUC-systemet, og som vil vere aktuelt for alle grensesnitt baserte på naturleg språk, er kor grensa skal gå med omsyn til kva som faktisk bør kunna handterast av systemet og ikkje. På den eine sida er målet at system av denne typen bør vere så intuitive og enkle å bruka som mogleg. For å oppnå dette må den lingvistiske kompetansen til systemet nødvendigvis vere svært god og den konseptuelle dekningsgrada større enn ho i utgangspunktet ville tronge å vere sett i høve til sjølve bruksområdet. På den andre sida vil det alltid finnast avgrensingar for kva som faktisk let seg gjennomføra i praksis og, ikkje minst, for kva som faktisk er verdt å prøva å gjennomføra i praksis. Ei detaljert og omfattande brukarvegleiing er i strid med tanken om eit mest mogleg sjølvforklarande og intuitivt grensesnitt. Enkelte forskarar, blant dei Karlgren, meiner like fullt at ei viss innføring i bruk er på sin plass, ikkje minst fordi denne typen grensesnitt framleis er relativt sjeldne. Det seier seg ikkje sjølv korleis ein best skal bruka eit gitt NLI-system. Avgrensingar kjem ein ikkje utanom og det er viktig at brukarane får eit inntrykk av kva for avgrensingar det i eit kvart tilfelle er tale om. Eit anna viktig poeng er at ulike NLIDB-system kan ha heilt ulike avgrensingar, ikkje bare når det gjeld konseptuell dekningsgrad, men òg når det gjeld lingvistisk kompetanse.

Som det har vorte vist tidlegare finst det nok av døme på at brukarane testar BussTUC-systemet. Testinga kan til tider minna om ei slags turingtesting der det er om å gjera å "lura" systemet, det vil seie avdekka at det faktisk er eit datasystem og ikkje ein person som svarar på spørsmåla. Som nemnt tidlegare kan òg underhaldingsverdi og tidtrøyte sjå ut til å vere av ei viss tyding når det gjeld testing. Det er uvisst i kva grad system som BussTUC bør framstå som "intelligente" for at brukarane skal få tiltru til dei og faktisk villa bruka dei. Det viser seg at spesielt uerfarne databrukarar gjerne forventar seg at system som ser ut til å forstå språk skal vere meir "intelligente" enn dei er og at det oppstår problem på grunn av dette. Som alltid er det viktig å finna ein middelveg. Det som først og fremst er viktig er sjølvsagt at systemet handterar det det er meint å skulla handtera. Om systemet handterer meir enn det er meint å skulla er det ein bonus, men burde strengt tatt ikkje vere nødvendig. Coates seier det slik: "The aims when testing applications that use natural language (NL) as their interface style should not be of the form "can I fool it?", but rather of the "can I use it?" category." [Coates - 02]

BussTUC-systemet er utvikla som eit forskingsprosjekt så vel som eit system berekna på faktisk bruk, noko som gjer det litt spesielt på fleire måtar. Mykje av det som har vorte lagt til av informasjon har vorte lagt til på grunnlag av dei innsamla innputtdøma, ein del mest for nettopp underhaldingsverdien si skuld. Generelt bør hovudregelen vere at avgrensingane i tilknyting til systemkompetansen er noko som vert gjort bevisst og mest mogleg eintydig, logisk og fornuftig, samt at ein informerer brukarane om kva systemet kan og ikkje.

Problema knytte til det å nytta naturleg språk som interaksjonsmetode kan på mange måtar sjå ut til å vega tyngre enn nytteverdien. Shneiderman sidestiller NLI med automatisk omsetjing og meiner at ein, til tross for forbetringar, aldri kan oppnå ein så høg kvalitet på sjølve språkprosesseringsdelen som må til for å laga verkeleg gode, robuste system som dekker meir enn bare eit avgrensa område. Kor nyttig og brukbart eit system er har ulike brukarar gjerne ulike oppfatningar om. Kva folk liker og ikkje er sjølvsagt svært individuelt, avhengig av kva oppgåver det er som skal utførast, med meir. Shneiderman påpeiker at det er viktig å prøva å finna ut – ved hjelp av grundig gjennomtenkte, eksperimentelle testar – kva for brukarar, oppgåver og grensesnittdesign naturleg språk egnar seg best for. Han meiner at spesielt brukarar med god kjennskap til spesifikke domene kan ha nytte av NLI. Ein må bort ifrå tanken på at NLI skal brukast for einkvar pris [Shneiderman -98, kap. 8.7.1].

Det finst ei mengd eksperiment der bruk av grensesnitt baserte på naturleg språk vert samanlikna med bruk av andre typar grensesnitt. Resultata er mildt sagt sprikande. Bruksområde, kvaliteten på dei ulike grensesnitta, opplæring, typen brukarar og nivået desse er på, testsituasjonen – om studiet har gått føre seg ute i ”felten” eller i meir kunstige omgjevnadar – er alle variablar som speler inn på resultata [Ogden og Bernick -96]. Det som kan slåast fast er at det alltid vil finnast både fordeler og ulemper forbunde med grensesnitt baserte på naturleg språk, som for alle andre typar grensesnitt. Bruksområdet og typen brukarar er, som Shneiderman påpeikar, to av dei viktigaste faktorane som speler inn med omsyn til kor vellukka ein applikasjon vert. Det finst utan tvil område der andre typar grensesnitt passer betre enn naturlegspråklege grensesnitt. Det finst òg applikasjonar der eit NLI gjerne ikkje fungerer som einaste interaksjonsalternativ.

Dei fleste forskarane er stort sett samde om ein ting, nemleg at det å kombinera ulike metodar i såkalla multimodale grensesnitt kan vere positivt. For BussTUC-systemet sin del verkar kombinasjonen av eit avansert og eit enkelt orakel til å vere svært vellukka. Det såkalla enkle oraklet er å føretrekke framfor det avanserte for det ein kanskje kan kalla standard spørsmål. I det enkle oraklet fyller brukaren inn standard element som avreisetid og -stad, framkomststad, osb. ved å velja ut ifrå lister med alternativ. Den store fordelen med dette er at ein slik får nytta ein fast, standard syntaks, noko som kan verka til å trengast. Vidare unngår ein stavefeil og det er enkelt å finna rett namn på haldeplass / stasjon. Det avanserte oraklet kan nyttast til alle innputt som avvik frå denne standarden, eller dersom ein som brukar føretrekk denne interaksjonsforma framfor den menybaserte. Med det såkalla enkle oraklet framstår nå BussTUC som eit spanande og feksibelt system. Dei BussTUC-brukarane eg kjenner til verkar stort sett godt nøgde med systemet og ser på oraklet som eit artig og interessant innslag på Team Trafikk-sidene.

Frå eit HCI-perspektiv vert NLI stort sett sett på som positivt. Grensesnitt baserte på naturleg språk er på mange måtar i tråd med målet om det som på engelsk vert omtala som ’ubiquitous computing’. Poenget er at datamskinen vert skyven meir i bakgrunnen og ut av fokus slik at brukar kan konsentrera seg om oppgåva ho skal utføra, i staden for på sjølve reiskapen

oppgåva skal utførast ved hjelp av. Spesielt vert NLI baserte på tale sett på som fordelaktige, mykje fordi desse kan nyttast av brukarar med handikap og brukarar som nyttar hendene til arbeid eller andre ting – som til dømes bilkøyring.

6.3 Fram i tid

Det er vanskeleg å seie om grensesnitt av denne typen vil verta meir vanlege fram i tid. Som introduksjonskapitlet fortel har ikkje prediksjonane slått til så langt. Det er likevel sannsynleg at det etterkvart vil verta fleire system som tilrettelegg for denne typen interaksjonsform, gjerne komplettert av ein eller fleire andre typar grensesnitt. Utviklinga går i retning stadig betre og meir effektive naturlegspråklege grensesnitt, mykje grunna framskitt på andre, relaterte område innan datalingvistikk og språkprosessering.

Databasesystem er kanskje den typen system NLI først og fremst vert forbunde med, men som nemnt i innleiinga til oppgåva kan grensesnitt baserte på naturleg språk, skriftleg og munnleg, òg nyttast til mange andre typar system. Det finst allereie døme på NLI brukte i tilknyting til operativsystem, teksteditorar, rekneark (spreadsheets), internettnavigasjon og ressurslokasjon, for å nemna noko. Fram i tid vil det truleg dukka opp fleire bruksområde. Dersom dette vert tilfellet vil langt fleire brukarar kjenna til og etter kvart verta meir komfortable og vane med interaksjonsforma. Dette vil i sin tur føra til at interaksjonen går lettare, ikkje minst fordi brukarane vil ha eit klarare bilete av kva eit system av denne typen kan og ikkje.

Truleg vil det òg dukka opp generelle retningslinjer og kanskje standardar for utvikling av denne typen grensesnitt. Dette er viktig for å kunna skapa mest mogleg homogene system, særskilt med omsyn til dei lingvistiske systemeigenskapane og avgrensingane knytte til desse. Jo meir like dei ulike grensesnitta er med omsyn til korleis dei bør brukast, kva for innputt som kan nyttast osb., jo enklare vert det for brukarane å tilpassa seg og ta i bruk nye system av denne typen.

Referanser

Allen, James F. (1995): *Natural Language Understanding*, 2. utgåve. Benjamin/Cummings, Redwood City, California.

Amble, Tore (2002): *The Understanding Computer – Natural language understanding in practice*. Institutt for datateknikk og informasjonsvitenskap (IDI), NTNU, Trondheim

Amble, Tore, Ranang, Martin T. og Sætre, Rune (2002): *The Understanding Computer: A Tutorial*. Institutt for datateknikk og informasjonsvitenskap (IDI), NTNU, Trondheim

Androutsopoulos, I., Ritchie, G. D. og Thanisch, P (1995): *Natural Language Interfaces to Databases - an Introduction*. Research Paper no 709, Dept. of AI, University of Edinburgh og cmp-lg/9503016, Mars -95. Finst òg i Journal of Natural Language Engineering, 1 (1):??, 1996 og online tilgjengeleg på http://citeseer.nj.nec.com/cachedpage/150857/1

Arnold, Doug: *Chart Parsing* [online] Tilgjengeleg på http://www.cs.ualberta.ca/~lindek/650/papers/chartParsing.pdf [Sitert 14.9.03]

Baustad, Jostein (1992): *Ambiguity in Natural Language*. Hovudfagsoppgåve skriven ved Institutt for datasystem og telematikk, NTNU, Trondheim.

Bratseth, Jon S. (1997): *Bustuc - A Natural Language Bus Traffic Information System* [online] Tilgjengeleg på http://www.idi.ntnu.no/~tagore/rapporter/bustuc.pdf [Sitert 6.10.03]

Carbonell, J. G. (1986): *Requirements for Robust Natural Language Interfaces: The LanguageCraft and XCALIBUR Experiences*. Frå den ellevte COLING-konferansen halden i Bonn, Tyskland, s. 162-163

Coates, Andrew P. (2002): Kapitlet *Spoken Natural Language Interface* fra *Speech Controlled Animation* [online]. The University of York. Tilgjengeleg på http://ska.ai-depot.com/paper/node68.html [Sitert 5.11.03]

Cohen, Philip, R (1992). *The role of natural language in a multimodal interface*. Frå det 5. årlege ACM-symposium: "User interface software and technology", s 143-149.

Connecticut Libraries: *Looking at Books Online*. [online]. Utgåve 38, nr. 11 (1996). Tilgjengeleg på http://cla.uconn.edu/reviews/mediaeqn.html [Sitert 9.7. 2003]

Covington, Michael A. (1994): *Natural Language Processing for Prolog Programmers*. Prentice Hall, Englewood Cliffs, NJ.

Faarlund, Jan Terje, Lie, Svein og Vannebo, Kjell Ivar (1997): *Norsk referansegrammatikk***.** Oslo : Universitetsforlaget

Gamut, L.T.F. (1991): *Logic, Language & Meaning. Volume 2: Intensional Logic & Logical Grammar*. The University of Chicago Press

Gazdar, Gerald og Mellish, Chris (1990): *Natural Language Processing in PROLOG. An Introduction to Computational Linguistics*. Mackays of Chatham plc.

Hancox, Peter.J: *A brief history of NLP* [online] Tilgjengeleg på http://www.cs.bham.ac.uk/~pjh/sem1a5/pt1/pt1_history.html [Sitert 5.2.03]

Inman, Dave (1997): *Ambiguity*. [online] Tilgjengeleg på http://www.scism.sbu.ac.uk/inmandw/tutorials/nlp/ambiguity/ambiguity.html [Sitert 2.3.03]

ITS - Information Technology Services (Windows Services): *Introduction to Data Modeling* [online] Tilgjengeleg på http://www.utexas.edu/its/windows/database/datamodeling/rm/overview.html [Sitert 11.3.03]

Jung, Hanmin og Lee, Gary Geunbae (2002): *Multilingual Question Answering with High Portability on Relational Databases* [online] Tilgjengeleg på http://www.isi.edu/~cyl/wsqa-coling2002/papers/P0007.pdf [Sitert 6.3.03]

Järvinen, Pertti (1992). *On Research into the Individual and Computing Systems* [online]. Tilgjengeleg på http://www.cs.uta.fi/reports/pdf/A-1992-4.pdf [Sitert 5.7.03]

Johannessen, Kjell S. (1987): *Tradisjoner og skoler i moderne vitenskapsfilosofi*, Sigma forlag, kap 2,7.

Jurafsky, Daniel Jurafsky og Martin, James H. (2000): *Speech and Language Processing*. Prentice Hall

Karlgren, Jussi (1992). *Discourse Modality and User Expectations, The Interaction of Discourse Modality and User Expectations in Human-Computer Dialog* [online]. Tilgjengeleg på http://www.sics.se/jussi/Artiklar/1992_Lic/ [Sitert 6.6.03]

Karlgren, Jussi (1992): *How Users Adapt to Natural Language Interfaces* [online] University of Stockholm, Department of Computer and Systems Sciences. Tilgjengeleg på http://www.sics.se/~jussi/Papers/1992_ANLP_Trento/trento.html [Sitert 10.5.03]

Long, Byron (1994): *Natural Language as an Interface Style* [online]. Tilgjengeleg på http://www.dgp.toronto.edu/people/byron/papers/nli.html [Sitert 2.10.03]

Lyse, Gunn Inger (2003): *Fra speilmetoden til automatisk ekstrahering av et betydningstagget korpus for WSD-formål* [online] Tilgjengeleg på http://www.ub.uib.no/elpub/2003/h/516001/Hovedoppgave.pdf [Sitert 4.7.03]

McKeown, Kathleen R. (1983): *Natural Language Systems: How are they meeting human needs?* [online] Tilgjengeleg på http://portal.acm.org/citation.cfm?id=809684 [Sitert 8.4.03]

Ogden, Willian C. og Bernick, Philip (1996): *Using Natural Language Interfaces* [online]. Tilgjengeleg på http://crl.nmsu.edu/Research/Pubs/MCCS/pdf/mccs-96-299.pdf [Sitert 22.10.03]

Orr, Robert H.: *GOTTFRIED WILHELM LEIBNIZ*. [online]. Tilgjengeleg på http://www.engr.iupui.edu/~orr/webpages/cpt120/mathbios/gleib.htm [Sitert 1.9. 2003]

Pereira, Fernando C. N. (1983): *Logic for Natural Language Analysis*. Technical Note 275, SRI International, Menlo Park, CA, USA

Pereira, Fernando C. N. og Shiber, Stuart M. (1987): *Prolog and natural-language analysis*. Stanford, CA

Quiroga-Clare, Cecilia (2003): *Language Ambiguity – A Curse and a Blessing* [online] Translation Journal, Volum 7, nr. 1 Januar 2003. Tilgjengeleg på http://accurapid.com/journal/23ambiguity.htm [Sitert 2.3.03]

Saeed, John I. (2000): *Semantics*. Blackwell Publishers Ltd. NB! Trykt fleire gonger: 1997, -98, -99.

Sag, Ivan A. og Wasow, Thomas (1999): *Syntactic Theory*. CSLI Publications

Sterling, Leon og Shapiro, Ehud (1994): *The Art of Prolog*. The MIT Press

The Net Net: *The Media Equation: how people treat computers, television, and new media like real people and places* – eit samandrag av Caitlin Burke. [online] Tilgjengeleg på http://www.thenetnet.com/schmeb/schmeb15.html [Sitert 6.9.03]

Thompson, Bozena H. og Thompson, Frederick B. (1982): *Knowledgeable Contexts for User Interaction* [online]. Calefornia Institute of Technology. Tilgjengeleg på http://caltechcstr.library.caltech.edu/archive/00000418/00/5051_TR_82.pdf [Sitert 11.9.03]

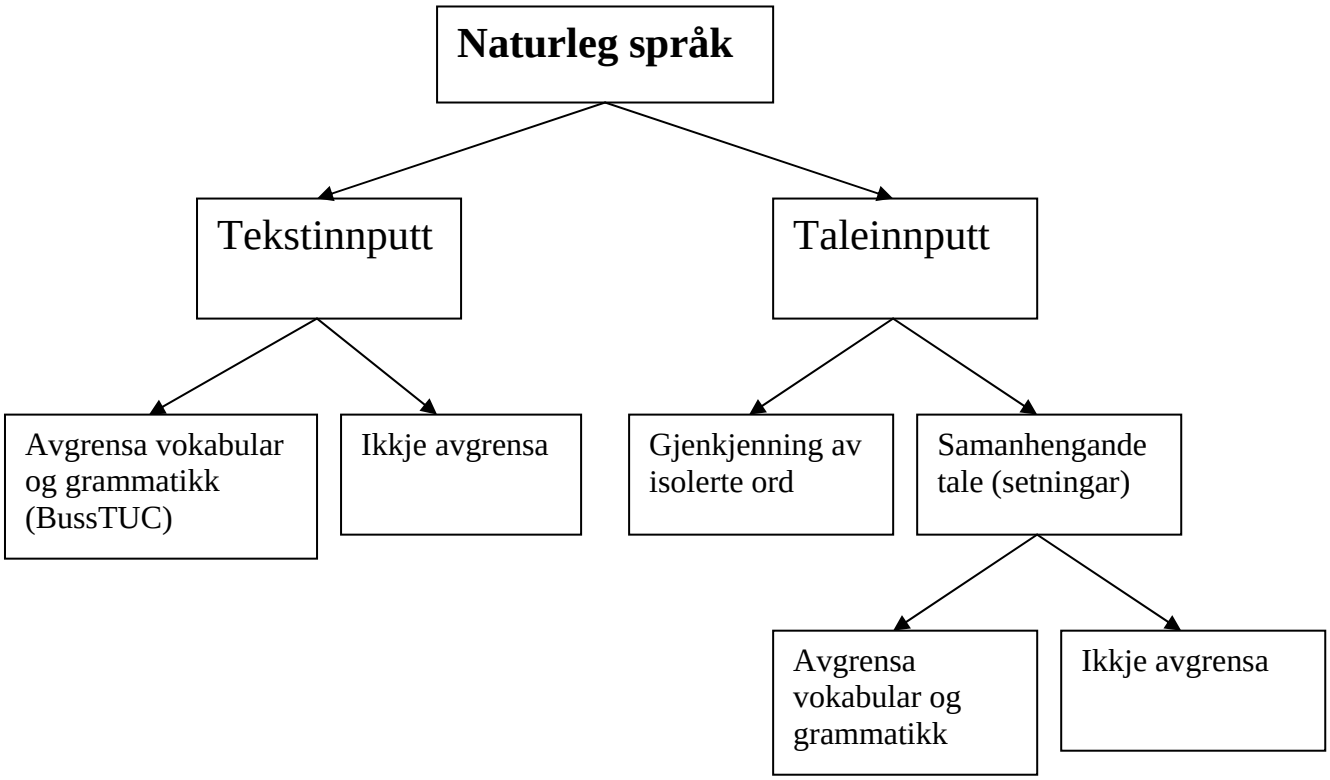
Véronis, Jean. *Error in natural language dialogue between man and machine*. I: International Journal of Man-Machine Studies. Årg. 35 (1992), s187-217 (frå Karlgren)

Whittaker, Steve, Stenton, Phil (1989): *User Studies and the Design of Natural Language Systems*. Frå den fjerde europeiske ACL-konferansen (the Association of Computational Linguistics), s. 116-123.

Zelkowitz, Marvin V, ed. (1998): *NLP – An HCI Perspective* frå *Advances in Computers* s. 1-66. Academic Press, Inc. [online] Tilgjengeleg på http://www.cacs.louisiana.edu/~manaris/publications/advances-in-computers-vol-47.html [Sitert 14.3.03]

Appendiks A

Oversikt over hovudtypane av språkprosessering:



Appendiks B

Feilmeldingar frå Web-innputt tekne ut til vidare testing

Innputta som er tekne med her er innputt som opprinneleg fekk feilmeldinga - - - *Uforståelig ved* * - - - når gitt inn til systemet. For innputt med andre feilmeldingar (i korpus eller eventuelt etter vidare testing) er feilmeldinga oppgitt i parantes.

Gir OK svar nå

- når går bussen fra berg studentby til ila pir i dag etter kl 10 .
- på hvilke tidspunkter går buss 36 forbi stokkanhaugen mot midtbyen etter kl 1900?
- Hvilken buss går om cirka en halv time fra Moholt til sentrum?
- Når går buss fra Fagerliveien i dag som er i byen ca 1835
- neste buss fra flatåsen til sentrum klokka etter 1200
- Hvor ofte går buss nr. 9 fra Heimdal til Hammersborg, etter kl. 1600?
- hvilken buss må jeg ta fra prinsen kino slik at at jeg er på lerkendal 16.50?
- Fra Tors vei til sentrum, ankomst før 1920.
- når går første buss fra samfundet til lade i dag som er fremme senest 0830
- Når går det buss fra Moholt Studentby til Jernbanestasjonen tirsdag, som er framme innen klokken 0735
- Når går buss 8 til Nardosentert fra Lerkendal, hvis jeg må være der senest kl 17:45?
- når må jeg ta buss fra sentrum til brundalen videregående skole for å være der senest 12.40?
- nest siste buss fra gløshaugen til munkegata som kommer frem til kl 13
- hva heter busstoppen som er nærmest olav magnussons vei 9?
- hva heter busstoppen som er nærmest valentinlystsenteret?

Gir delvis OK svar nå

- hvilken linje på nattbussen går * fra granåsen .
Svar: ”Det er ikke mulig natt til torsdag. Nattbusser går bare natt til lørdag og natt til søndag. Holdeplassen nærmest Granåsen er Granåsen VM-anlegget."
- Kva heiter busstoppet som er før Sverresborg folkemuseum når ein kjem frå byen?
Svar: ”Holdeplassen nærmest byen er Sentrumsterminalen. Du må oppgi en buss i slike spørsmål Første buss, nummer 93 passerer Dronningensgate D1 klokken 0526. Neste buss heretter, nummer 52 passerer Munkegata M2 klokken 1626. Siste buss, nummer 60 passerer Munkegata M3 klokken 0032.”
- Når går trikk fra sentrum lørdag som er på Ugla klokken 1820?
Svar: Gir svar om at det ikkje går trikk sentrum – Ugla lørdag.
- neste fem busser fra nardosenteret til sentrum.
Svar: Svarer med to bussar.
- neste fem busser fra vestlia.
Svar: Svarer med to bussar (til sentrum).
- når kan jeg tidligst * komme meg til sentrum fra lundåsen på en søndag ?
Svar: ”Buss nr. 91 passerer Lundåsen kl. 0609 og Dronn.gt D3 kl. 0645.
... Buss nr. 4 passerer Lundåsen kl. 2351 og Munkegt M5 kl. 0021.”
- når må man ta buss nummer 3 fra sunnalnd skole veg for å komme * til sykehuset kl 13 . 50?
Svar: ”Buss 3 passerer aldri St.Olavs Hospital. Holdeplassen nærmest Sykehuset er St.Olavs Hospital.”
- når må man ta buss nummer 3 fra sunnalnd skole veg for å være * på sykehuset kl 13 . 50 ?
Ved å kutta ut “nummer 3” får ein svar at det ikkje går (fleire) bussar på onsdagar..

Ufullstendig setning som burde gå(?)

- neste to rute 66 fra moholt til sentrum. (Gir - - - Vennligst bruk en fullstendig setning - - -)
- de neste 2 busser fra kattemskogen til sentrum.
(Gir - - - Vennligst bruk en fullstendig setning - - -)
- hvilken buss fra voll studentby for å være * i sentrum kl 1245 . (Fungerer så snart ein nyttar ei fullstendig setning, med "må jeg ta"etter *buss*, t.d.).

Alle tider vert tolka som ønska ankomsttid (?)

- når kommer buss nr 44 til adressa når den * går fra sentrum kl 1515 .
Svar: ”Buss 44 går fra Munkegt M2 kl. 1430 til Godst. NSB kl. 1451 og buss 44 går fra Godst. NSB kl. 1512 til Adresseavisen kl. 1515.”
- når passerer nr 9 gryta når den * går 1345 fra byen ?
Svar: ”Buss 9 passerer Munkegata M5 kl. 1302 og Gryta kl. 1320.”

Komplekse setningar / fraser, samt *hvis*-setningar

- når går neste buss etter buss nr 24 fra KBS til Sentrum
- Når går første buss fra sentrum til City syd, hva er bussnummeret?
- hva er de neste fem busser som går fra voll gård til samfundet unntatt de to første.
- hvor skal jeg ta bussen fra hvis * jeg skal fra city syd til byen ?
- hvis * jeg skal til byåsen hvor må jeg gå av hvis jeg kommer fra munkegaten ?

Problem knytt til tal (rute- / bussnummer, dato, gatenr,m.m.)

- når går bussen fra moholt til elgeseter 47 etter * 23 .
- når går bussen fra moholt til elgeseter 47 etter * klokka 23 .
- når går neste buss etter 1001 bussen fra lundåsen til sentrum ?
- når går de 5 neste bussene rute nr 66 fra byen ?
- når går de 3 første 5 * bussene fra østre berg etter kl 18 til sentrum .
- når går de 3 48 fra kattem . *
- når går neste 10 66 busser fra studentersamfundet ? *
- når går neste 10 nummer 66 busser fra studentersamfundet ? *
- når passerer neste 5 buenget moholt . *
- går 4 klokka 20 . *

Dato-problem

- er det mulig å ta en buss direkte fra flatåsen til kroppanmarka når som helst i uke 47 ?
- er det mulig å ta en buss direkte fra flatåsen til kroppanmarka i uke 47 ? *
- når går neste buss til sentrum fra valentinlyst etter kokken 8 : 20 den 191202 .

S*om*- / relativsetningar

- hvilke 5 busser kommer til prinsen kinosenter fra moholt studentby som * er framme før 20.30.
- hvilke busser kan jeg ta fra moholt studentby som * er framme ved prinsen kinosenter før 20.30 .
- på hvilke tidspunkter går buss 36 forbi stokkanhaugen mot midtbyen den neste timen ? *
(Gir - - - Setningen er for lang og vanskelig - - -)

Manglar i databasen (?)

- når går buss fra * storås til trondheim ?
- når går siste ikea buss tilbake * fra ikea ?
- når går siste ikea buss fra ikea ? *
- hvilke busser går fra sentrum til tt på * sorgenfri .
- hvilken buss går til * hofstads veg ?
- Hva heter bussholdeplassen før Schrøder der buss 3 kjører.
- hvilke buss kan jeg ta forbi einar tambarskjelves gt som * korresponderer med buss 24 19 . 15 ?
- når må jeg ta buss fra scandic moholt for * å være på ntnu lade klokken 830 på lørdag ?
- hvor i byn går bussen fra?
- når går bussen fra lerkendal gård til 36 thoning owesens gate klokken 14 . 30 . 1430.
- når går bussen fra sentrum nr 15 til st olavs hospital etter kl 1600 .
- når går 66 fra charlottenlund kirke den nermeste timen . *

Kategori ikkje avgjort (evt. Fleire)

- Kva heiter busstoppet som er før Sverresborg folkemuseum når ein kjem frå byen?
- hvilket tidspunkt har buss nummer 20 sin neste avgang?
(Gir - - - Setningen er for vanskelig - - -)
- når går neste buss nr 54 etter avgang kl.2015 fra munkegata m2?
- når går neste buss nr 54 etter avgang kl.2015
- hvilke neste fire busser går * fra ila til sentrum ?
- når på fredag nermest klokken 1300 går * det buss fra sentrum til heimdal
- hvilken buss tar kortest tid fra samfundet til rønningsbakken når du skal være på rønningsbakken kl 2000 ? *
- når kjører tidligste buss 11i * dag ?
- when does the next bus leave tvetestien driving south *
- Når går neste buss etter den som går fra moholt studby 1907?
(Gir - - - Setningen er for vanskelig - - -)
- Når går neste buss til sentrum etter den som går fra moholt studby 1907?
(Gir - - Setningen er for vanskelig - - -)
- Når går tidligste buss forbi moholt studentby søndags morgen til byen?
(Gir - - - Setningen er for vanskelig - - -)
- når nå jeg ta bussen fra buenet for å være * i sentrum klokken to ?
- når nå jeg ta bussen fra buenget for å komme * til sentrum klokken to ?
- når nå jeg ta bussen fra buenget for å komme * til sentrum klokken 1400 ?
- når nå jeg ta bussen fra buenget for å være * i sentrum klokken 1400 ?
- breidablikk byåsen hegdalen bergheim terrasse jeg skal * være på hegdalen ca kl 20 . 0 .
(Gir ”Jeg kan ikke svare for så mange steder av gangen !”)
- når passerer 36 sentrum lerkendal * mellom klokken 20 . 30 og 21 . 0 .
- når må jeg dra fra skansen for å være på * håkon magnussons g 3 1030.
- når går bussen fra samfunnet til ikea så den fremme * før 2100 ?

Testsetningar

1a) Når går de neste X bussene?
1b) Når går de neste X-bussene?
2a) Når går de X neste X-bussene?
2b) Når går de X neste X bussene?
3a) Når går de neste X X-bussene?
3b) Når går de neste X X bussene?
4a) Når går de neste X-bussene fra ... til ...?
4b) Når går de neste X bussene fra ... til ...?
5a) Når går de neste X-bussene fra ...?
5b) Når går de neste X bussene fra ...?
6a) Når går det X-busser fra lade?
6b) Når går det X busser fra lade?
7a) Neste X-busser fra ...?
7b) Neste X busser fra ...?
8a) Går X-bussen fra ...?
8b) Går X bussen fra ...?
9a) Når går den neste X-bussen?
9b) Når går den neste X bussen?
10a) Når går neste X-buss?
10b) Når går neste X buss?
11a) Når går neste X-buss fra ... til ...?
11b) Når går neste X buss fra ... til ...?
12a) Når går neste X-buss fra ...?
12b) Når går neste X buss fra ...?

Tilbakemeldingar frå systemet

Loggfil / **Systemutputt**	**WEB-log-02**	**WEB-log-03**	**SMS-log**
Du må oppgi et sted i slike spørsmål	913	594	1
Jeg har bare ruter for busser i Trondheim	361	385	10
Det er ikke mulig (på lørdag, etc.)	227	327	26
Jeg vet ikke	271	149	1
Jeg har ingen informasjon om leder / flybuss / retning / overgang / rabatt / hyppighet / ekstraavgang / spesielle billetter / forsinkelser / kino / ruteendringer / busskort / sone / hastighet / internett / sykkel / sitteplass / vær	107	100	5
Jeg kan ikke svare på det	106	87	-
Stedet X er flertydig. Vennligst bruk et mer presist navn	89	57	8
Jeg kan fortelle deg om bussavganger i Trondheim	38	30	-
Du må oppgi en buss i slike spørsmål	5	5	1
Det er ingen rute som kalles X	-	18	-
Jeg kan ikke svare for mange steder av gangen	-	46	5
Jeg kan bare svare på en ting ad gangen	-	-	8

Appendiks C

Parsingdøme

N: når går neste buss 4?

```
w(når,[verb(reach,pres,fin),[når],name(når,n,0)])
w(går,[verb(go,pres,fin),[går],name(går,n,0)])
w(neste,[adj(next),name(neste,n,0)])
w(buss,[noun(bus,sin,u,n),name(buss,n,0)])
w(4,[nb(4,num),name(4,n,route),name(4,n,0)])
w(?,[[?],name(?,n,0)])

txt(0, w(når,verb(reach,pres,fin)), 1).
txt(0, w(når,[når]), 1).
txt(1, w(går,verb(go,pres,fin)), 2).
txt(1, w(går,[går]), 2).
txt(2, w(neste,adj(next)), 3).
txt(3, w(buss,noun(bus,sin,u,n)), 4).
txt(4, w(4,nb(4,num)), 5).
txt(4, w(4,name(4,n,route)), 5).
txt(5, w(?,[?]), 6).


*** Parse Tree ***

sentence
  onesentence
    sentence1
      greetings0
        []
      question
        whenq
          when_adverbial
            whenx
              when
                naar
                  [når]
                andwhere0
                  []
                !
                accept
                  []
          !
          question0
            ynq
              verb(go,pres,fin)
              saa0
                []
              {}
              negation
                []
              !
              st2
                {}
                st
                  noun_phrase
                    noun_phrases
                      noun_phrase1
                        np1
                          np_kernel
```

```
                the0
                  []
                aname_phrase
                  preadjs0
                    adj(next)
                    preadjs0
                      []
                  name_phrase
                    namep
                      athe0
                        []
                      noun2
                        noun
                          noun(bus,sin,u,n)
                      nameq
                        nameq1
                          name(4,n,route)
                          {}
                          {}
                          {}
                      {}
                      {}
                      !
                      accept
                        []
                {}
              noun_modifiers
                noun_complements
                  []
          noun_phrases0
            []
      verb_phrase
        verb_phrase1
          do0
            do0
              []
          reductant0
            []
          do_phrase
            vp_head
              intrans_verb
                lexv
                  aux0
                    do0
                      []
                  verb(go,pres,fin)
                  {}
              {}
            reciprov0
              []
            negation2
              negation
                []
            event00
              {}
          it0
            []
          verb_complements0
            verb_complements
              verb_complement
                adverbial1
                  nil
                {}
```

```
                            verb_complements10
                              verb_complements0
                                {}
                      !
                      accept
                        []
                    verb_phrases0
                      []
                descend00
                  []
                {}
          prep0
            []
      qtrailer0
        []
    terminatore
      terminator1
        terminator
          termchar
            [?]
          !
          accept
            []
        !
        accept
          []
    !
    accept
      []
  {}
  nil
  moresentences
    onesentence
      endofline
        nil
      !
      accept
        []
    nil
    evenmore
      []
```

```
******************

[which(A:time)::true and((A isa time and(exists(B:event)::go/4/B and
srel/in/time/A/B and event/real/B))and 4 isa bus)and
adj/next/4/real]

[which(A:time)::true and((A isa time and(exists(B:event)::go/4/B and
srel/in/time/A/B and event/real/B))and 4 isa bus)and
adj/next/4/real]

[which(A)::(4 isa bus,A isa
time,adj/next/4/real,go/4/B,srel/in/time/A/B,event/real/B)]
```

330 ms

~~~~~~

\+

Buss 4 passerer Munkegata M5 kl. 1836.
~~~~~~

Appendiks D

Reglane som vert brukte ved parsing av 1. testsetning, "når går neste buss fra Lade?"

```
sentence(ListOfS)  --->
    onesentence(S1),
    { S1 \== error}.
    check_stop.
    moresentences(S1,ListOfS).


onesentence(S) --->  sentence1(S),terminatore,!,accept.


sentence1(P) --->
    greetings0,
    question(P),
    qtrailer0.


question(P) ---> whenq(P).


whenq(which(Y)::P) --->
    when_adverbial(In,Y,Subj,P3),
    !,
    question0(P) - adverbial1(In,Y,Subj,P3),
    prep0(_HOC).


when_adverbial(in,T:time,P,T isa time and P) --->
    whenx.


whenx ---> when.


when ---> naar,andwhere0,!,accept.


naar ---> [når].


question0(P) ---> ynq(P).


ynq(P) --->
    w(verb(Speak,Pres,fin)),
    saa0,
    {Speak \== have},
    negation(_N),
    !,
    st2(P) - w(verb(Speak,Pres,fin)).


st2(Q) --->
    {user:value(world,Real)},
    st(S,N,  Com,P),
```

```
    descend00(Real,  S,Com:P,Q1),
    {negate(N,Q1,Q)}.


st(S,N,Com,P) --->
    noun_phrase(X,P1,P),
    verb_phrase(X,S,N,Com:P1).


noun_phrase(X,P1,P) --->
    noun_phrases(X,P1,P).


noun_phrases(Z,VP,P2) --->
    noun_phrase1(X,VP,P1),
    noun_phrases0(X,Z,VP,P1,P2).


noun_phrase1(X,VP,P) --->
    np1(X,VP,P).


np1(X,VP,P) --->
    np_kernel(Ind, X,P0,P1,VP, P),
    noun_modifiers(Ind,X,P0,P1).


np_kernel(Ind, X,P0,P1,VP ,P) --->
    np_head(Ind, X,P0,P1,VP, P).


np_head(Cind, XT,P0,P1,VP, P) --->
    determiner0(Num,Cind, XT,P1,VP, P),
    preadjs0(Alist),
    {Alist=true -> Num1=Num;Num1= _},
    noun_complex(XT,Q0,Num1),
    {screenmeasure(Num,XT)},
    {preadjs_template(Alist,XT,Q0,P0)}.


determiner0(_,0, X,P1,P2, P) --->
    determiner00(X,P1,P2, P).


determiner00(X,P1,P2, exists(X):: P1 and P2 ) ---> the0.


preadjs0((A,Blist)) --->
   w(adj(A)),
   preadjs0(Blist).


noun_complex(X,P,Num) --->
    noun_compound(X,P,Num).


noun_compound(X,QP,Num1) --->
    noun2(Num2,U,GN1, Y, P),
     {Num1=Num2},
    sine(Num2,U,GN1,GN2),
    ncomps0(GN2,X,Y,P,QP,_).
```

```
noun2(Num,U,Gen, A,B) --->
    noun(_NN,Num,U,Gen, A,B).


noun(Country,Num,U,Gen, X:Country, X isa Country) --->
    w(noun(Country,Num,U,Gen)).


ncomps0(n,X1,X2,P,P,name) --->
    {compatvar(X1,X2)}.


noun_modifiers(OP,X,P,Q) --->
     noun_complements(OP,X,P,Q).


noun_complements(0,X,P,R) --->
   noun_complement(X,P,Q),
   !,accept,
   noun_complements(0,X,Q,R).


noun_complement(X,P1,P1 and P3) --->
    negation(N),
    complement1_accept(Prep,Y,NP2,P3),
    {noun_compl(Prep,X,Y,P2)},
    {negate(N,P2,NP2)}.


complement1_accept(Prep,Y,SC,P3) --->  % Eigentleg "prep" --> fiksa!
    prep1(Prep),
    {\+ Prep1 = of},
    noun_phrase_accept(Y,SC,P3).


prep1(P)---> w(prep(P)).


noun_phrase_accept(X,VP,P) --->
   noun_phrase(X,VP,P),
   !,accept.


np_kernel(Ind,X,true,P1,VP,P1 and Q) --->
    the0,
    determiner0(Num,Cind,XT,P1,VP,P),
    aname_phrase(Ind,X,VP,Q).


aname_phrase(Ind,XT,VP,P1) --->
    preadjs0(Alist),
    name_phrase(Ind,XT,VP,P0),
    {preadjs_template(Alist,XT,P0,P1)}.


name_phrase(Ind,X,P,P and Q ) --->
    namep(Ind,X,Q).


namep(name,X,Y) --->
    nameq(X,Y).
```

```
nameq(C,Q) ---> nameq1(C,Q).


nameq1(N:Type,N isa Class) --->
    w(name(N,_n,Class)),
    { Class == unkn -> value(unknownflag,true);true},
    {_n \== gen},
    {type(Class,Type)}.


verb_phrase(X,S,N,Com:P12) --->
    verb_phrase1(X,S,N,Com:P1),
    verb_phrases0(X,S,P1,Com,P12).


verb_phrase1(X,S,M,ComP2) --->
    do0(M,N),
    reductant0,
    do_phrase(X,S,N,ComP2),
    !,accept.


do0(N,N) ---> do0.


do_phrase(X,S,N,Com3P3) --->
    vp_head(V,X,S,N, ComP1 ),
    it0,
    verb_complements0(V,X,S,ComP1, Com3P3 ).


vp_head(V,X,S,N,  Com3:P1 )  --->
    intrans_verb(V,X,S,N1,P,_tense,_fin),
    reciprov0(V),
    negation2(N1,N),
    event00(S,P,Com3,P1).


intrans_verb(Live,X,S,id,EXS,P,Q) --->
    lexv(iv,Live,P,Q),
    {iv_template(Live,X,S,EXS)}.


lexv(T,Live,P,Q)--->
    aux0,
    w(verb(Live,P,Q)),
    {verbtype(Live,T)}.


aux0 ---> do0.


negation2(not,not) ---> negation(id).


event00(S,P,Q,exists(S:Event)::P and Q) --->
    {type(event,Event)}.


verb_complements0(V,X,S,Com1P1,Com12P3) --->
    verb_complements(V,X,S,Com1P1,Com12P3).
```

```
verb_complements(V,X,S,Com1P1,Com12P3) --->
    verb_complement(V,X,S,Com1P1,Com12P2),
    verb_complements10(V,X,S,Com12P2,Com12P3).


verb_complement(V,X,S,(Compl and Com1):Subj,Com1:P3 ) --->
   adverbial1(Prep,Y,Subj,P3),
   {verb_compl(V,Prep,X,Y,S,Compl)}.


verb_complements10(V,X,S,Com1P1,Com12P3) --->
    verb_complements0(V,X,S,Com1P1,Com12P3).


verb_complements0(V,_,_,CP,CP) --->
    {\+ V=be1}.


terminatore ---> terminator1.


terminator1 ---> terminator,!,accept.


terminator ---> termchar,!,accept.


termchar ---> ['?'].


moresentences(S1,Square) --->
    onesentence(S2),
    check_stop,
    evenmore(S1,S2,Square).


onesentence([]) --->  endofline,!,accept.
```

Appendiks E

Kartparsarkoden

%Operatorar (frå declare.pl):

```
:-op(1150,xfx, ---> ).
:-op( 800,fx,  def ).
:-op( 730,xfy, :: ).
:-op( 729,xfx, : ).
:-op( 727,xfy, => ).
:-op( 727,yfx, if ).
:-op( 726,xfx, then ).
:-op( 725,xfy, or ).
:-op( 720,xfy, and ).
:-op( 720,xfy, & ).
:-op( 719,yfx, butnot ).
:-op( 715, fy, not ).
:-op( 714,xfx, := ).
:-op( 713,xfx, =: ).
:-op( 710,xfx, ako ).
:-op( 710,xfx, apo ).
:-op( 710,xfx, isa ).
:-op( 710,xfx, has_a ).
:-op( 710,xfx, is_the ).
:-op( 720,xfy, -- ).
:-op( 500,xfy, \  ).
:-op( 500,xfy,-...).
:-op( 500,xfy,-.).

:-dynamic edge_inaktiv/7.
:-dynamic edge_aktiv/7.
:-dynamic chart_length/1.

:- use_module(library(system)).
:- ensure_loaded(library(terms)).
```

%%

% Laga med utgangspunkt i ein standardkode henta frå Gazdar & Mellish (-90), %

% kap 6.9 - *Implementing a simple top-down chart parser* %

%%

```
edge_inaktiv(START,START,FINISH,LABEL,TOFIND,FOUND,FILLER).
edge_aktiv(FINISH,START,FINISH,LABEL,TOFIND,FOUND,FILLER).
```

%Inaktive kantar som leitar etter passande aktive kantar søker etter kantar som %ender der dei sjølve byrjar, difor vert aktive kantar lagra for seg sjølve og indekserte %med utgangspunkt i 'FINISH'. Aktive kantar søker etter inaktive kantar som startar %der dei sjølve ender, difor vert inaktive kantar indekserte etter startnode.

% Initialisering

```
start(Symbol,String):-
        retractall(edge_inaktiv(_,_,_,_,_,_,_)),
        retractall(edge_aktiv(_,_,_,_,_,_,_)),
        retractall(chart_length(_)),
        statistics(runtime,_),
```

```
        parse(Symbol,String),
        statistics(runtime,[_,Tid]),
        write(Tid).
```
%'statistics' er henta frå fila 'sicstuc.pl' og vert nytta for å måla tida parsinga tek.

```
parse(Symbol,String):-
  start_chart(0, String),
  start_active(0,Symbol,X--X),
  foreach(edge_inaktiv(V0,V0,Vn,sentence1(_),[],Parse,Filler),
                write(Parse)).


initial(sentence(_)).

start_chart(Vn,[]):-
         assert(chart_length(Vn)).
```
%`chart_length` er lagt til for at det skal vere mogleg å stoppa for kvar gong ein %fullstendig parse er funnen. Sjå den fundamentale regelen (regel 13) nedanfor.

```
start_chart(V0,[Word|Words]):-
    V1 is V0+1,
    foreach(morfological_word(Word,Cat),
         add_edge(V0,V0,V1,Cat,[],[Word,Cat],X--X)),
    start_chart(V1,Words).
```

%`Filler` = tom differanseliste: `X--X`

```
morfological_word(Word,Cat):-
     lcodes(Word,w(Word,Cats)),
     member(Cat,Cats).
```

%`member` vert nytta for å gå gjennom alle elemeta i kategorilistene (i `lcodes`). "når" %har t.d. tre kategoriar og vert difor lagt til i kartet tre gonger, ein gong for kvar av %kategoriane. (`member` finst nedanfor).

%Pga. kategorialuttrykk:
```
start_active(V0,Cat1 – Cats4):-
    foreach(rule1(Cat1,Cats),
     add_edge(V0,V0,V0,Cat1,Cats,[Cat1],Cats4)).
```
%Kategorialgrammatikkoperatoren "-" tyder at ein skal gå ut ifrå at det finst ein %`Cats4` og prøva `Cat1` med og utan denne.
%`Cats4` vert lagt til som element i `Filler`-lista.

%Vanleg `start_active`:
```
start_active(V0,Category,Filler):-
  foreach(rule1(Category,Categories),
    add_edge(V0,V0,V0,Category,Categories,[Category],Filler)).


rule1(Category, Body) :-
     (Category ---> Body_m),
     \+ (Body_m = []),
     p2list(Body_m,Body).


rule1(Cat,[]):- (Cat ---> []).
```

% Parsaren krev at det som står i 'body' er på listeform:

```
p2list((X),[X]) :- \+ X=(_,_).
p2list((X,Y),[X|Y1]):- p2list(Y,Y1).
```

%For handtering av kategorialgrammatiske fraser.
%Vanleg parantesliste vert gjort om til differanseliste:

```
p2dlist((X),[X|Z]--Z) :- \+ X=(_,_).
p2dlist((X,Y),[X|D]--Z):- p2dlist(Y,D--Z).
```

%Behandlar dei kategorialgrammatiske operatorane '/', '\' og '-' likt:

(1)
```
add_edge(V2,V1,V2,Cat,[Cat1/Cats2|Cats],Parses,Fillers):-
    add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers).
```

(2)
```
add_edge(V2,V1,V2,Cat,[Cat1\Cats2|Cats],Parses,Fillers):-
    add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers).
```

% For behandling av kategorialuttrykk.
% For reglar innehaldande eitt 'w-uttrykk', som 'ynq'.

(3)
```
add_edge(V2,V1,V2,Cat,[Cat1 - w(Cat2)|Cats],Parses,Fillers):-
     add_edge(V2,V1,V2,Cat,[Cat1 - Cat2|Cats],Parses,Fillers).
```

%Må sjekka òg om kantar innehaldande kategorialgrammatiske uttrykk har vorte lagt %til i kartet frå før:

(4)
```
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers):-
        edge_aktiv(V2,V2,V2,Cat1,SomeCats,_,[Cats2|X]--X),!.
```

% --> første kanten `start_active` resulterer i.

%Sjekk for inaktive kantar:

(5)
```
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers):-
        edge_inaktiv(V2,V2,V2,Cat1,[],_,[Cats2|X]--X),!.
```

% Alle kategorialgrammatiske fraser kan vere komplekse, dvs. delen til høgre for den %kategorialgrammatiske operatoren kan bestå av ei heil liste:

(6)
```
add_edge(V2,V1,V2,Cat,[Cat1 - Cats2|Cats],Parses,Fillers):-
 asserta(edge_aktiv(V2,V1,V2,Cat,[Cat1|Cats],Parses,Fillers)),
        p2dlist(Cats2,Cats3),
        (member_dl(w(Cat3),Cats3) -> p2dlist(Cat3,Cat4),
            append_dl(Cat4,Cats3,Cats4);
          start_active(V2,Cat1 - Cats3)),
        start_active(V2,Cat1 - Cats4).
```

%`p2dlist` gjer paranteslista om til differanseliste.
%viss-så-løkka er laga for å fjerna eventuelle 'w'-uttrykk frå den kategorialgram. lista.

% For å unngå å leggja til reine prologuttrykk (i '{}'-parantes, som skal tas direkte) %som kantar:

(7)
```
add_edge(V2,V1,V2,Cat,[{PrologClause}|Cats],Parse,_):-
```

```
      PrologClause,
      (true -> add_edge(_,V1,V2,Cat,Cats,Parse,_))
          ;!.
```

% laga pga. 'accept' og kutt:

```
(8) add_edge(V2,V1,V2,Cat,[Funny|Cats],Parse,_):-
      rejectclause(Funny),
      add_edge(_,V1,V2,Cat,Cats,Parse,_).


rejectclause(!).
rejectclause(accept).
```

%For reglar innehaldande `w`-uttrykk, som `ynq`:

```
(9) add_edge(V2,V1,V2,Cat, [w(Cat1)|Cats], Parses,Fillers):-
      add_edge(V2,V1,V2,Cat,[Cat1|Cats],Parses,Fillers).
```

%Sjekker om kanten finst frå før - køyrer i så fall ein cut:

```
(10) add_edge(V0,V0,V1,Category,[],_,_):-
        edge_inaktiv(V0,V0,V1,Category,[],_,_),
      !.

(11) add_edge(V1,V0,V1,Category,Categories,_,_):-
        edge_aktiv(V1,V0,V1,Category,Categories,_,_),
      !.
```

%Nyttar opne variablar for både `Parse` og `Filler` for å unngå rekursjon m.o.t. %reglar som `verb_complement`-reglane o.a. `Parse`-listene er nemleg alltid ulike, %medan `Category`, dvs. `LABEL`, er lik (med unntak av argumenta som vert utvida %ved rekursjon).

% Både fundamental og topp-ned-regel:

```
(12) add_edge(V2,V1,V2,Cat1,[Cat2|Cats],Parses,Fillers):-
 asserta(edge_aktiv(V2,V1,V2,Cat1,[Cat2|Cats],Parses,Filler)),
      foreach(edge_inaktiv(V2,V2,V3,Cat2,[],Parse,Filler),
      append_dl(Filler,Fillers,Ny_filler),
      add_edge(_,V1,V3,Cat1,Cats,[Parse|Parses],Ny_filler),
      (member_dl(Cat2,Fillers) ->
      add_edge(V2,V2,V2,Cat2,[],[Cat2],X--X))
      ;
      start_active(V2,Cat2,Fillers).
```

% Den fundamentale regel:

```
(13) add_edge(V1,V1,V2,Cat1,[],Parse,Fillers):-
    asserta(edge_inaktiv(V1,V1,V2,Cat1,[],Parse,Fillers)),
    chart_length(Vn),
    ((V1 = 0, Vn = V2) -> write(Cat1),nl,get0(_);true),
    foreach(edge_aktiv(V1,V0,V1,Cat2,[Cat1|Cats],Parses,Filler,
       append_dl(Filler,Fillers,Ny_filler),
       add_edge(_,V0,V2,Cat2,Cats,[Parse|Parses],Ny_filler)).
```

%Dersom aktuelle, inaktive kant spenner over heile kartet, dvs. byrjar i første node og %ender i sluttnoda (spesifisert av `chart_length`) skal denne skrivast ut.

%`member` og `append` for differanselister (for å kunna handtera `FILLER`-listene):

```
member(X,[X|_]).
member(X,[_|Y]):- member(X,Y).
```

```
member_dl(X,V--Y):- member_var(X,V).

member_var(X,Var):- var(Var),!,fail.
member_var(X,[V|_]):- \+ var(V), X=V.
member_var(X,[_|Y]):- member_var(X,Y).

append_dl(P--Q1,Q1--R,P--R).


foreach(X,Y):- X, do1(Y), fail.
foreach(X,Y):- true.

foreach(X,Y,Z):- X, do1(Y), do1(Z), fail.
foreach(X,Y,Z):- true.
```
% `foreach` med tre argument vart lagt til pga. at `FILLER` og dermed `append_dl` %vart teke i bruk.

```
do1(Y):- Y,!.
```