

Designing for Text Refinement Tasks

Fredrik Olsson
fredrik.olsson@sics.se

16 oktober 2000

Det här är en revidering av det tidigare licentiatavhandlingsförslaget, daterat till den 23:e augusti, 2000.

Syftet med den här texten är att formulera frågeställningar och ansatser till en licentiatavhandling. Om de tankar jag tar upp här möter de krav på vetenskaplighet och rimlighet som institutionen för lingvistik i Uppsala och SICS har, så hoppas jag ha avhandlingen klar i början av år 2001.

1 Bakgrund

Resten av den här texten är organiserad som följer. I avsnitt 1.1 tar jag upp relevanta delar av vad jag har gjort tidigare inom datorlingvistik, medan avsnitt 1.2 beskriver vad jag gör just nu. Avsnitt 1.3 tar upp vad jag vill göra i framtiden när det gäller min forskning. Det finns (åtminstone) en röd tråd (avsnitt 1.4) som knyter samman perioderna och som jag hoppas kan utvecklas till en licentiatavhandling.

Avsnitt 2 handlar om hur jag har tänkt lägga upp arbetet med avhandlingen. Dess syfte tas upp i avsnitt 2.1, den tänkta metoden beskrivs kort i avsnitt 2.2, ett utkast till disposition av uppsatsen finns i avsnitt 2.4 och det hela avslutas med en tidsplan i avsnitt 2.5.

1.1 SVENSK

Mellan februari 1998 och december 1999 jobbade jag mer eller mindre heltid med ett projekt kallat SVENSK (Eriksson och Gambäck, 1997; Olsson et al., 1998). Det övergripande målet med SVENSK var att samla in, integrera och distribuera program för olika typer av behandling av skriven svenska. Vi valde GATE, General Architecture for Text Engineering,

som plattform att integrera den insamlade programmen i. Spridningen av SVENSK är tänkt att ske till forskare och icke vinstdrivande institutioner som ett hjälpmedel i forskning och undervisning i Sverige. För att nå huvudmålet har vi varit tvugna att lösa en mängd delmål av mjukvaruteknisk, lingvistisk och 'politisk' natur (Gambäck och Olsson, 2000; Olsson och Gambäck, 2000).

Det är främst de mjukvarutekniska och lingvistiska frågeställningarna jag har sysslat med och som har varit till stor hjälp så här långt i arbetet med Kaba (se nästa avsnitt).

1.2 Kaba

Kaba är arbetsnamnet på ett system för textförädling som jag, Kristofer Franzén, Jussi Karlgren och Anna Jonsson har skissat på och börjat implementera sedan hösten 1999. textförädling anser vi vara ett samlingsbegrepp för program som på olika sätt framhäver, sammanställer eller på något sätt strukturerar relevanta bitar av information ur löpande text. Exempel på sådana tillämpningar är informationsextraktion med tillhörande deluppgifter, sammanfattning, nyckelordsutdragning och indexering.

Det finns flera anledningar till att vi inte valde en redan existerande plattform för Kaba, bl.a. motsvarar inget idag existerande system våra behov och vi vill också ha möjligheten att fritt distribuera källkoden till Kaba.

1.3 Kaba i framtiden

Hur Kaba kommer att se ut i framtiden beror till stor del på resultatet av den eventuella avhandlingen. De övergripande målen med Kaba inkluderar följande: Kaba kommer att vara en verktygslåda för utvecklare att skapa textförädlingssystem med; det ska finnas hjälpmedel (programbibliotek och färdiga

moduler) som underlättar för utvecklaren att göra systemet attraktivt för slutanvändare (t.ex. integrerade maskininlärningsmetoder för utvärdering och intrimning av program), bl.a. kommer det att finnas möjlighet att integrera externa komponenter (på liknande sätt som man kan göra i GATE) samt en Kaba-specifik formalism för att bygga vissa typer av interna komponenter (t.ex. grammatiker för namnigenkänning, diskursmodeller och nyckelordsigenkänning); Kaba ska kunna användas som en del av ett större system.

Kaba kommer således att vara en generell verktygslåda för en viss typ av uppgifter där uppgifternas natur tillåts styra utformningen och intrimningen av systemen.

1.4 Den röda tråden

Den röda tråden i mitt arbete med SVENSK och Kaba är erfarenheterna som jag fått genom att arbeta med andras plattformar och genom att vara med och bygga en. Jag tror att Kaba kan bli bättre än något av de ramverk som finns idag, men för att vi ska komma dit krävs till att börja med en genomtänkt design för en genomtänkt klass av uppgifter.

Gemensamt för alla typer av ramverk och verktygslådor för naturligt språk är att de syftar till att underlätta och snabba upp framtagningen av nya komponenter eller anpassningen av dessa till nya typer av data.

Det finns olika typer av sådana verktyg, dels för att utveckla enstaka komponenter, t.ex. ordklasstaggare och lexikon och dels för att ta fram hela system, t.ex. maskinöversättningssystem, stavningskontroller och informationsextraktionssystem. Den sistnämnda kategorin av system är den som intresserar mig mest. Till den hör ramverk som t.ex. GATE, ALEP och DARPA Communicator. Men, jag anser att det finns flera fundamentala problem med de ansatser till generella ramverk och verktygslådor som finns idag;

(P-1) De är inte fokuserade på en klass av uppgifter. Att ett verktyg är generellt behöver inte, och ska inte i min mening, betyda att det ska gå att konstruera system för att lösa godtyckliga uppgifter. Det är sedan länge känt att full textförståelse inte är möjlig, i alla fall inte inom de teorier och praktiker som finns idag. Men så fort man begränsar sig (domän, täckningsgrad, språkliga konstruktioner), så lyckas man betydligt bättre. T.ex. fungerar informationsextraktion just för att uppgifterna som ska lösas ofta är väldefinierade (även om det som gör att IE fungerar också är

det som hittills gjort att tekniken inte är spritt allmängods; anpassningen av IE-system till nya domäner är fortfarande tidsödande och svårt).

(P-2) De är inte anpassade till mer än en typ av användare. Det finns minst tre typer av användare av ett system, utvecklare, "underhållare" (systemerare) och slutanvändare. Utvecklaren är den som utifrån en verktygslåda bygger ett system för att lösa en viss uppgift, t.ex. ett hjälpmedel för att känna igen och kategorisera namn i tidningstext som ett hjälpmedel i omvärldsbevakningen hos företag. Utvecklaren måste veta allt om hur systemet fungerar, han måste också veta och förstå hur det är tänkt att användas. "Underhållaren" är den som underhåller programmet m.h.a. de verktyg som utvecklaren tillhandahåller (anpassar det till nya typer av data samt utvärderar det m.a.p. relevanta mått för täckning m.m.). "Underhållaren" ska inte behöva vara expert på hur systemet fungerar, däremot måste han vara expert på hur det ska användas och ha djupa kunskaper inom domänen där det verkar. Slut användaren är, i fallet med omvärldsbevakning, den människa som läser igenom mängder med texter varje dag för att sammanställa information som svarar mot ett visst behov. Det förefaller självklart att dessa tre olika typer av användare har olika behov. Jag har dock inte sett något ramverk som hjälper utvecklaren att göra verktyg som är anpassade till olika typer av användare.

(P-3) De stöder antingen integrering av externa komponenter eller bygger på formalismer för att ta fram ramverksspecifika komponenter. GATE tillhör den förra kategorin. Det har visat sig att det går att få program att kommunicera med varandra även om de från början inte var avsedda att göra det. GATE kräver ingen information om *hur* olika program fungerar, så länge man vet *vad* de kräver för att kunna exekvera samt vilken typ av information de producerar. ALEP, å andra sidan, är ett exempel på ramverk som kräver att utvecklaren använder en speciell formalism. Detta medför en hög inlärningströskel och att det tar tid innan man kommer igång med det man förutsatt sig göra. Det finns förstås för- och nackdelar med båda ansatserna och det jag vill påpeka är att det, mig veterligen, inte finns något ramverk som stöder dem båda.

(P-4) De är inte designade för integrerade maskininlärningsmetoder. Maskininläring är ett bra verktyg i konstruerandet och underhållandet

av textförädlingssystem och är ett relativt nytt grepp i sammanhanget. Få av de existerande plattformarna innehåller färdigimplementerade maskininlärningsmetoder som utvecklare kan använda som delar av, eller som hjälpmedel för att konstruera, nya system. Jag tror att integrerade maskininlärningsmetoder skulle underlätta hanteringen av textförädlingssystem på ett flertal sätt och för samtliga typer av användare; utvecklare kan t.ex. behöva dem för att anpassa informationsextraktionssystem till nya texttyper och domäner; "underhållaren" kan använda dem för att inordna nya texter i en existerande ontologi; slutanvändaren kan komma att använda dem indirekt om de används för t.ex. underhåll av en användarprofil som styr vilken information en specifik slutanvändare får.

Hur ovanstående fyra punkter är relevanta för mitt avhandlingsarbete förklaras i avsnitt 2.1 (nedan).

2 Licentiatavhandlingen

I det här avsnittet ska jag förklara vad jag vill uppnå med en licavhandling om systemarkitekturer och verktygslådor för behandling av naturligt språk. Att designa, implementera och testa ett system som Kaba ligger utanför vad man kan hinna med i en lic. Jag kommer att koncentrera mig på designen och delar av implementationen och därtill relaterade frågor. Avhandlingen kommer att utgå från generella system, t.ex. GATE och ALEP, för att sedan rikta in sig på generella system för den specifika klassen av uppgifter som kan kallas textförädling.

Uppsatsen kommer att skrivas på engelska och ett förslag till titel är *Software Architectures for Language Engineering: Designing for Text Refinement*, ett annat förslag är *Software Design for Language Engineering: The Case of Text Refinement*.

2.1 Syfte

Avhandlingen kommer att vara fokuserad på system som ska användas dagligdags, dvs. praktiska system, inte på teoretiska ramverk för att undersöka lingvistiska fenomen. Inriktningen kommer alltså att vara mycket språkteknologisk.

Jag ser två huvudsyften med avhandlingen;

1. Att ta fram ett genomtänkt och klart designförslag för Kaba. Delar av förslaget kommer att vara tillämpligt i arbetet att designa och implementera en generell plattform för godtycklig uppgift inom behandling av språk.

De främsta deluppgifterna som måste lösas i arbetet med att ta fram ett designförslag är dessa:

- Vilka typer av uppgifter ska system byggda m.h.a. Kaba kunna lösa? Identifiera och exemplifiera dessa? (Jmfr. P-1 ovan). Finns det klart urskiljbara klasser av datorlingvistiska uppgifter (vad skiljer t.ex. maskinöversättningssystem från informationsextraktionssystem åt ur ett datorlingvistiskt perspektiv)? Vilka är dessa i så fall?
- Hur ska designen av Kaba göras för att tillgodose behov hos olika typer av användare? (Jmfr. P-2 ovan). Vilka typer av användare finns det och vilka är deras behov?
- Hur ska ett system som både har en intern formalism och samtidigt tillåter integrering av externa komponenter se ut? (Jmfr. P-3 ovan).
- Hur ska Kaba designas för att underlätta användandet av maskininlärningsmetoder i anslutning till att nya system byggs och gamla system underhålls? (Jmfr. P-4 ovan). Jag skulle vilja bygga in bibliotek med tillhörande gränssnitt i Kaba (både API:n och byggstenar som underlättar byggandet av grafiska användargränssnitt). Vilka maskininlärningsmetoder är lämpliga för den typ av uppgifter som textförädling innebär?

2. Att staka ut kursen för en framtida doktorsavhandling genom att väl specificera ett antal forskningsfrågor som tar avstamp i det nuvarande arbetet med att bygga Kaba.

2.2 Metod

Det är viktigt att relatera vårt arbete med Kaba till vad som har gjorts tidigare inom området. Jag kommer därför att göra en stor litteraturstudie (möjligtvis med testning av tillgängliga program där det passar). Med studien som grund kan jag plocka ut de bästa och sämsta egenskaperna hos respektive system, generalisera dessa och kombinera dem med mina erfarenheter från arbetet med SVENSK. Kombinationen ska sedan extrapoleras till Kaba.

Parallellt med litteraturstudien kommer jag att jobba med implementationen av delar av Kaba. Syftet med implementationen är att få ett

fungerande, om än inte fullt funktionellt, system som kan användas som för att illustrera och testa idéer i designen.

Eftersom jag vill begränsa arbetets omfattning och därmed inte kommer att hinna implementera Kaba fullt ut, så kommer det inte att finnas utrymme för användarstudier eller större experiment.

Hur ska jag kunna verifiera att designen av Kaba fungerar?

2.3 Mitt bidrag till forskningen

Mitt bidrag till forskningsvärlden kommer att vara en (1) en analys av existerande system, tillsammans med mina erfarenheter från SVENSK, (2) en kravspecifikation av vilka egenskaper ett system som Kaba ska uppfylla, samt (3) ett designförslag för Kaba.

2.4 Disposition

I Figur 1 finns ett förslag till disposition av avhandlingen. I det här avsnittet ska jag helt kort gå igenom och förklara delarna.

Avsnittet **Introduction** ska förklara vad avhandlingen handlar om, mitt bidrag till det hela och hur texten är organiserad. Här ska även den röda tråden presenteras på ett sådant sätt att de två följande, något splittrade kapitlen förstås i sitt sammanhang.

The notion of text refinement ska handla om vad textförädling är, vilka system, tekniker och metoder som jag anser karakteriserar området. En viktig del av kapitlet är den som tar upp exempel på hur och var textförädling kan förekomma.

An Analysis of some of Today's Platforms ska handla om vad som har gjorts tidigare vad det gäller generella ansatser till mjukvaruplattformar för datorlingvistik. Jag kommer främst att titta på system som "lever" idag. Underrubrikerna kan komma att ändras beroende på resultatet av litteraturstudien. Vidare ska kapitlet beskriva de fördelar och nackdelar som jag anser finnas i dagens plattformar. Ur det här avsnittet kommer ett antal designprinciper att ta form, dvs. generaliseringar av fördelarna hos existerande plattformar kombinerade med saker vi (jag) tycker ska finnas i ett system av Kabas kaliber.

Det kapitlet som tidigare hette **Experiences from composing a general-purpose toolset for LE: SVENSK** kommer att införlivas som en del av kapitlet och kallas **A case study — SVENSK**.

Specification of requirements handlar om vilka krav man kan ställa på ett system av samma generalitetsklass som Kaba. Kravspecen är kombinationen av de två föregående kapitlen.

Avsnittet **The Design of an Text Refinement Framework: Kaba** ska handla om hur Kaba ska se ut med utgångspunkt i de föregående avsnitten. Texten kommer att börja ganska abstrakt med de egenskaper systemet ska ha, för att sedan gå ner i detalj i den implementation av Kaba-kärnan och mönstermatchningsspråket (den interna formalismen) som vi har gjort hittills.

Future Work – Evaluation criteria and methods är ett av de viktigaste avsnitten i avhandlingen. Här har jag tänkt att ta upp ett par spår som jag skulle vilja följa upp i min kommande forskning med utgångspunkt i Kaba. Därför borde den inte heta enbart **Future work**, titeln måste mer specifikt spegla inriktningen för mina framtida studier. Kapitlet ska kunna få licentiatavhandlingen att fungera som ett doktorsavhandlingsförslag.

Summary and Conclusions ska ta upp mina generella erfarenheter av systemarkitekturer för datorlingvistik som en mängd designriktlinjer. Arbetet med att implementera och konkretisera erfarenheterna i Kaba och sättet på vilket Kaba är tänkt att användas kommer också att tas upp.

I avsnittet **Appendices** har jag tänkt visa lite av den implementation vi har gjort; programmeringsgränssnittet (API) till Kaba-kärnan tillsammans med definitionen av mönstermatchningsspråket.

2.5 Tidsplan

I Tabell 2.5 finns ett utkast till en tidsplan för avhandlingsarbetet.

Datum	Aktivitet
29 sep	Lämna in kap. 2, 3 & 4
13 okt	Diskutera kap. 2, 3 & 4
3 nov	Lämna in kap. 1 & 5
15 nov	Diskutera kap. 1 & 5
8 dec	Lämna in kap. 6, 7 & bilagor
19 dec	Diskutera kap. 6, 7 & bilagor

Tabell 1: Tidsplan för avhandlingsarbetet.

Referenser

Mikael Eriksson och Björn Gambäck. 1997. SVENSK: A toolbox of Swedish language processing resources. I *Proceedings of the 2nd In-*

Ändrat!

Ändrat!

ternational Conference on Recent Advances in Natural Language Processing, sidorna 336–341, Tzigov Chark, Bulgaria, September.

Björn Gambäck och Fredrik Olsson. 2000. Experiences of language engineering algorithm reuse. I *Proceedings of the 2nd International Conferencen on Language Resources and Evaluation*, Athens, Greece, May. European Language Resources Association.

Fredrik Olsson, Björn Gambäck, och Mikael Eriksson. 1998. Reusing swedish language processing resources in svensk. I *Workshop on Minimizing the Effort for Language Resource Acquisition*, Granada, Spain, May. European Language Resources Association.

Fredrik Olsson och Björn Gambäck. 2000. Composing a general-purpose toolbox for swedish. I *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Luxembourg, Luxemburg, August. Association of Computational Linguistics. To appear in the Workshop on Using Toolsets and Architectures to Build NLP Systems.

-
1. Introduction
 - (a) Contributions by the Author
 - (b) Outline of the Thesis
 2. The notion of text refinement
 - (a) Example scenarios
 - (b) Example techniques
 3. An analysis of some of today's platforms
 - (a) TIPSTER
 - (b) Some systems
 - (c) A case study — SVENSK
 - (d) Discussion — pros and cons in today's platforms
 4. Specification of requirements for a text refinement system — Kaba
 - (a) Project dirvers
 - (b) Project constraints
 - (c) Functional requirements
 - (d) Non-functional requirements
 - (e) Project issues
 5. The Design of Kaba
 - (a) Design Principles
 - (b) A description of the initial implementation
 6. Future work – Evaluation criteria and methods
 7. Summary and Conclusions
 8. References
 9. Appendices
 - (a) Definition of the core Kaba API
 - (b) Definition of the Kaba Pattern Matching Language
-

Figur 1: Disposition av avhandlingen.