

Reusing Swedish Language Processing Resources in SVENSK

Fredrik Olsson Björn Gambäck

Information and Language Engineering Group
Swedish Institute of Computer Science
Box 1263, S-164 28 Kista, Sweden
fredriko,gamback@sics.se

Mikael Eriksson

Department of Element Management Development
Ericsson Utvecklings AB, AXE Research & Development
Box 1505, S-125 25 Älvsjö, Sweden
Mikael.P.Eriksson@uab.ericsson.se

Abstract

The SVENSK project is developing an integrated toolbox of language processing components and resources for Swedish. SVENSK employs GATE, General Architecture for Text Engineering from the University of Sheffield as a platform in which the components are to be integrated. The goal is that the resources included in SVENSK should be freely available for non-commercial use. A wide range of different modules have been incorporated so far, both in-house modules, commercially available modules, and modules from academia.

The results of the integration of the modules in the GATE environment are very encouraging: it is possible to mix modules from different sources, written in programming languages from completely different paradigms and have them interact with each other, thus maintaining a high degree of reuse of algorithmical resources. However, the use of Tcl/Tk and the associated API for processing structurally relatively complex data, is time consuming and considerably slows the processing in GATE.

1. Introduction

Traditionally, the level of reusability of language processing resources within the research community has been very low. Most of the recycling of linguistic resources has been concerned with reuse of data, that is, of corpora, lexica, and (to some extent) grammars, while the algorithmic resources far too seldom have been shared between different projects and institutions. Even though there are some repositories for algorithms (e.g., the DFKI Natural Language Software Registry, an initiative of the Association for Computational Linguistics) in the same fashion as there are archives for data (such as the Linguistic Data Consortium or the Oxford Text Archive), a prime obstacle to reuse of algorithmic resources has been that there have been few easily-available development platforms for language engineering. A researcher willing to reuse somebody else's processing components has been forced to invest major efforts into issues of integration, inter-process communication, and interface design.

In a national Swedish effort to circumvent this problem, the SVENSK project aims to develop a multi-purpose language processing system for Swedish based, where possible, on existing components. Rather than building a single monolithic language processing system which attempts to meet the (diverse) needs of academia and industry, the project creates a general toolbox of reusable language processing components and resources, primarily aimed at teaching and research. The generality of the system arises from its adherence to general principles of language engineering, combined with tools which facilitate adaptation to specific domains and applications.

The reusability of the system arises from having each component integrated into a language engineering development platform called GATE, General Architecture for Text Engineering (Cunningham et al., 1996b), which together with the definition of standard interfaces, ensures interoperability of the major components. The system makes it possible to incorporate linguistic data resources (such as lexica, tagged corpora, etc.) provided by Swedish universities and research institutes, so that SVENSK can be seen as a computational linguist's toolkit for the Swedish language.

In the US, there has been some efforts in the direction of open architectures for language processing, both by single companies — such as the multimodal Open Agent Architecture™ from SRI International (Cohen et al., 1994; Moran et al., 1997) — and in particular within the research programmes sponsored by DARPA, the Defense Advanced Research Projects Agency where a general architecture called TIPSTER (Grishman and others, 1997) has been agreed upon.

In Europe, there has so far has been few efforts in this direction. ALEP, the Advanced Language Engineering Platform (Simpkins and Groenendijk, 1994; Bredenkamp et al., 1997) is an initiative of the European Commission which aims in this direction. It provides a range of processing resources and is particularly targeted at supporting multilinguality. ALEP does, however, impose its own formalisms (for grammars, etc.) on the users.

The German Verbmobil architecture (Bub and Schwinn, 1996) incorporates components developed at several different sites and in many different programming paradigms. It employs a communication package called ICE (Intarc Communication Environment), but is of course primarily developed for the specific needs of the Verbmobil project.

The University of Sheffield GATE system — the platform underlying SVENSK — is based on the US TIPSTER architecture and is in itself language and domain indepen-

dent, providing merely an environment in which language processing resources may communicate with each other using a standardised interface, a so-called “wrapper”. The strong modularity of GATE gives users the choice of working with supplied, off-the-shelf systems or to create their own, built from integrated modules in a point-and-click fashion. The system is also equipped with a set of “viewers”, that is, programs that format and display the output from the modules in a number of different ways depending on the nature of the data under consideration.

In the rest of the paper, we will first discuss the overall system architecture in the next section and then in Section 3 describe the different language processing components which have been included in the SVENSK toolbox so far. The integration task is discussed in Section 4, while Section 5 outlines the future directions which we intend the SVENSK project to move in.

2. System Architecture

The SVENSK system as such is mainly the sum of a fairly large set of different reusable language processing resources. In this section, we will briefly outline the SVENSK project itself, but first go into detail on the GATE platform, discussing both its architecture and the design scheme which it in turn builds on, the TIPSTER architecture.

2.1. The TIPSTER Architecture

The TIPSTER project is led by DARPA and is funded by a number of US Government agencies. The main aim of the project is to provide developers and users with an architecture that allows for document detection and information retrieval in very large texts.

The architecture is described in (Grishman and others, 1997). It is developed with governmental agencies with similar text handling requirements in mind. There are four components to the architecture: *detection*, *extraction*, *annotation* and *document management*. Detection is the technology which perform text retrieval, Extraction is the technology for identifying entities and relations between entities in free text. The Annotation Model allows for information sharing between the former two components, while Document Management handles files.

2.2. GATE

The “General Architecture for Text Engineering”, GATE, language engineering platform (Cunningham et al., 1996b) is developed at the University of Sheffield and funded by the U.K. Engineering and Physical Sciences Research Council (EPSRC). GATE provides a communication and control infrastructure for linking together sets of language engineering software and is based on the Tcl/Tk communication toolkit. GATE does not adhere to a particular linguistic theory, but is rather an architecture and a development environment designed to fit the needs of researchers and application developers. However, a set of example components are included in the GATE release; together these components form a system called VIE, for “Vanilla Information Extraction” (Humphreys et al., 1996a).

Each component integrated into GATE has a standard I/O interface, which conforms to the annotation model of the TIPSTER architecture.¹ This means that GATE compatibility equals TIPSTER compatibility, allowing developers (and users) to easily add all types of TIPSTER compatible components to the platform, and then link them together to form an application.

GATE supports reuse of resources, data as well as algorithms, since it provides for well-defined, user-friendly application programmers interfaces (APIs). Once a module has been integrated in the system, it is very easy to combine it with already existing modules to form new systems. GATE contributes to the portability of components in the sense that software written in programming languages stemming from completely different paradigms can be mixed.

GATE is available free for non-commercial use. It will most likely only be used as a development platform in the SVENSK system; commercial exploitation is not expected to use GATE directly, only the relevant SVENSK components.

2.3. The SVENSK Project

The SVENSK project (Eriksson and Gambäck, 1997) is divided into three phases, the first of which was covering the period from spring 1996 to the end of 1996 and the second from the beginning of 1997 to August 1997. The third phase started in January 1998 and runs until the end of 1999. SVENSK is funded by the Swedish National Board for Industrial and Technical Development (Nutek) and SICS.

During the first two phases of the project, a large set of reusable, modular components have been integrated into the GATE environment, components for text preprocessing, for morphological processing and tagging, as well as for different levels of grammatical analysis. Some of the resources come from commercial companies and some from Swedish academia, while others have been developed at other projects as well as directly for SVENSK. The components, which will be described in detail in Section 3, are

SWETWOL: a two-level morphology for Swedish.

SWECG: a constraint grammar style tagger for Swedish.

SWETWOL and SWECG were originally developed at the Department of General Linguistics, University of Helsinki and then turned into a commercial product by Lingsoft Inc., Helsinki.

Brill Tagger: a Swedish version of Eric Brill’s part-of-speech tagger.

UCP: a large-scale chart processing component which can operate at several levels of linguistic description.

UCP and the tagger were developed for other projects by Uppsala University.

DUP: a deep-level unification-based processor consisting of a grammar and an LR-parser.

¹Currently, the GATE platform supports document management but not the information retrieval classes, see (Cunningham et al., 1996a, 16-18).

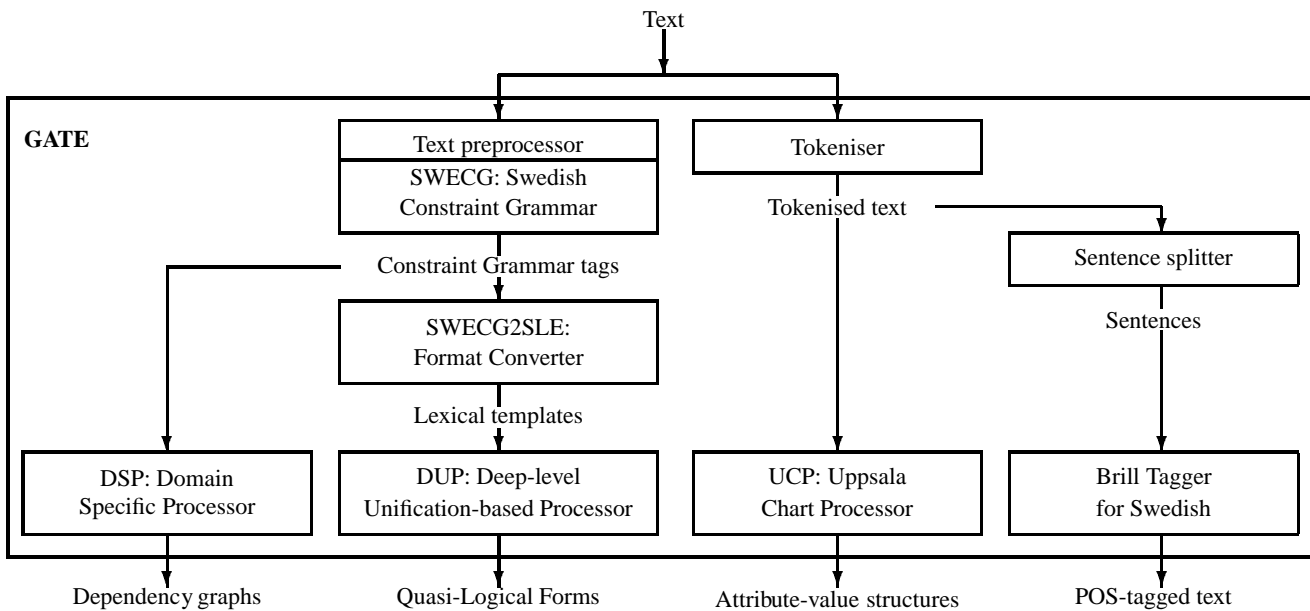


Figure 1: The SVENSK system

DSP: a domain-specific, shallow-level processor in dependency grammar style.

DUP and DSP were previously developed at SICS for other projects: DUP together with Telia and SRI International; DSP together with the Royal Institute of Technology and Stockholm University.

SWECG2SLE: a conversion module that converts SWECG tags into corresponding feature structures which can serve as input to DUP.

Tokeniser: a rule based tokeniser utilizing a stop list of frequent Swedish abbreviations for identifying tokens in unrestricted text.

Sentence Splitter: a rule based sentence splitter for Swedish.

These modules were developed at SICS explicitly for the SVENSK project; the tokeniser and sentence splitter together with Uppsala University.

All these components conform to GATE interface standards, hence allowing their integration into user applications.

3. Language Components

Figure 1 shows how the modules in GATE are interconnected to form different processing chains in the system. As indicated in the figure, several processing chains are possible, for example, SWECG → DSP and SWECG → SWECG2SLE → DUP. This section gives a description of the core components of the SVENSK system.

3.1. Tokenisation and Segmentation Resources

The tokeniser and sentence splitter for Swedish (Olsson, 1998) are both based on similar modules for English

included in the GATE release (Humphreys et al., 1996b).

3.1.1. A Tokeniser for Swedish

The tokeniser functions as a pre-processing module for the sentence splitter and UCP modules described below and uses mainly structural — and only a limited amount of linguistic — information in its aim to recognise tokens. The problem of tokenisation is often set aside even though tokens are the basic items in many text processing systems. Tokenisation is far from trivial. Some, but not all, of the tokens may be described and identified without the use of linguistic knowledge.

The linguistic knowledge employed by the tokeniser is in the form of a stop list containing frequent abbreviations. The list is used for distinguishing the cases of a token containing a period (.) in which the period should be considered as a sentence delimiter from those cases where it should not. The tokeniser also recognises and signals consecutive new-line characters in the input, information that may be used at later processing steps to identify headings as sentence fragments.

The stop list of abbreviations is obtained from a 300,000 word portion of the Stockholm-Umeå Corpus (Ejerhed et al., 1992). After extracting them, the abbreviations were sorted according to frequency, and all those which occurred more than once and had a period in them were picked out.

3.1.2. A Sentence Splitter for Swedish

The sentence splitter for Swedish is a pre-processing module intended for use in the GATE platform, which may as well be run as a stand alone program. Its primary task is to provide the Brill tagger described in Section 3.2.1 with sentences and annotations associated with them.

The problem of splitting a text into sentences is related to that of tokenisation. Possible approaches to solving the problem of finding and, if necessary, disambiguating sentence boundaries may involve a system that uses rules (e.g.,

a hand-made or induced grammar), probabilistics (e.g., HMMs), or some sort of neural network. Common to all approaches is that they require one or more pre-processing modules such as a tokeniser, morphological analyser, or a program that calculates word frequencies.

The sentence splitter for Swedish implements a rule-based approach using a minimum of information since it is not geared towards any specific corpora (thus assuming a minimum of information to be present in the input). The information present is mostly structural. The amount of linguistic knowledge is small and implicitly provided by the tokeniser.

3.2. Resources for Tagging and Morphology

Currently, there are three components that facilitate part-of-speech tagging and morphological processing in SVENSK; a version of the Brill tagger for Swedish; the Swedish two-level morphology and constraint grammar; and the morphological processing level of the Uppsala Chart Processor.

3.2.1. A Swedish Version of the Brill Tagger

The Eric Brill's tagger (Brill, 1992) uses a method called transformation-based error-driven learning for inferring rules from a tagged training corpus. The rules are then applicable for tagging previously unseen text. The Brill tagger is public domain software.

The Swedish version of the tagger has been trained on the Swedish side of a multilingual parallel corpus developed at Uppsala University (Prütz, 1997).

3.2.2. Two-Level Morphology and Constraint Grammar

SWETWOL (Karlsson, 1992) is a comprehensive morphological description of Swedish utilizing the two-level morphology technique (Koskenniemi, 1983). It consists of a set of finite-state rule automata and a dictionary of approximately 80,000 entries where a typical entry covers all inflectional forms of one word, and possibly also forms of some words derived from the entry. The SWETWOL output is the words in the input text together with their associated tags. There could be several tag sets associated to one word, corresponding to some ambiguity in the morphological analysis.

SWECG (Swedish Constraint Grammar) is a surface-oriented syntactic description of Swedish grammar using the same dependency-grammar-related description technique as in the English Constraint Grammar (Karlsson et al., 1995). The SWECG component disambiguates multiple morphological readings, determines the location of intra-sentential clause boundaries, and assigns syntactic roles to individual words. It picks the correct alternative with an error rate not exceeding 0.3%, leaving some 5% of the morphological ambiguities undecided.

Since SWETWOL and SWECG are commercial products from Lingsoft Inc., SVENSK users are required to obtain a separate license for these.

3.2.3. The Uppsala Chart Processor, UCP

The Uppsala Chart Processor, UCP, is written in LISP and was originally developed at the Center for Computational Linguistics, Uppsala University in the early 80's

(Sågval-Hein, 1981).

UCP is able to process the input in a variety of ways and with respect to different levels of linguistic description, e.g., only morphological processing of the input, or syntactic processing using a top-down filtering, chart-guided bottom-up parsing approach. The system has been used within several research projects, most recently as a part of a grammar checker for controlled Swedish in the SCANIA project (Sågval-Hein et al., 1997).

3.3. Grammatical Processing Resources

Two different grammatical descriptions for Swedish and associated parsers have been integrated into the system, so far. One large-scale unification-based grammar with an LR parser and one shallow-level dependency-based processor.

3.3.1. Deep-Level Unification-Based Processor

The deep-level processing component DUP comprises a Swedish unification grammar (Gambäck, 1997) and an LR parser (Samuelsson, 1994) developed at SICS. The grammar is supplemented with a fully-defined lexicon of about 1500 words. The component provides a relatively 'deep' level of analysis but at the cost of robustness. DUP is written purely in Prolog.

The unification-based grammar was originally developed for the Swedish Language Engine system (Gambäck and Rayner, 1992) and designed from the Core Language Engine, a similar system for English built by SRI International Cambridge, England (Alshawi et al., 1992). The grammar formalism is a feature-category type with declarative bidirectional rules, which means that the same grammar can be used for both analysis and generation of sentences into (resp. from) their equivalences in Quasi Logical Form (Alshawi and van Eijck, 1989), an underspecified, compositional semantic representation.

3.3.2. Domain-Specific Processor

The domain-specific processing component DSP builds directly on the SWECG output to provide semantic representations and is aimed at projects which require a shallower, but robust, natural language interface for a specific application (unlike DUP, there is no compositional semantic analysis). As DUP, DSP is Prolog-based. It was originally developed as a speech recognition back-end in the multimodal agent system OLGA (Beskow et al., 1997).

DSP builds an explicit dependency graph representation from the SWECG surface analysis: the representation may include multiple graphs spanning the same input in cases of ambiguity; or it may be a set of dependency analyses in cases where the input is incomplete or fragmentary. The dependency graph analyses are transformed into semantic representations by matching the graphs against domain-specific templates (Sunnehall, 1996).

4. Integrating Modules in GATE

The infrastructure of GATE provides several levels of integration, reflecting how closely a new module should be connected to the core system; however, at all integration

levels a wrapper code must surround the core module code. This code describes the communication between the module and the GATE system, i.e., it describes the mapping between the I/O of the module and the corresponding structure in the TIPSTER Annotation Model.

Also, for each module, there must be a specification of the type of input and output data of the module in terms of the Annotation Model. This is the information that GATE needs in order to connect the module to the other integrated modules in the system.

There are two APIs facilitating the integration of modules in GATE; one in C++ and one in Tcl. The choice of integration level for a new module affects mainly the wrapper coding and the way the module is loaded into the system. Three levels of integration, or couplings, are provided:

tight coupling

demands that the wrapper must be written in C, or in languages which obey C linkage conventions. The wrapper is compiled together with the whole GATE system. This is the most efficient level concerning execution speed, but each time a new version of the module is to be loaded, the whole GATE system must be recompiled. Thus, this level is for final versions of the modules, and for modules that always should reside in GATE.

dynamic coupling

also requires that the wrappers should be written in C, but shared libraries make the module loadable in run-time.

loose coupling

allows the user to write the wrapper in Tcl, making it easy to develop a wrapper and load the module at run-time. Also, wrappers can be altered and reloaded without having to leave GATE. Thus, this coupling is most suitable for the development phase.

In the SVENSK system each module is loosely coupled, allowing for ease of integration. However, the use of Tcl/Tk and the associated API for processing structurally relatively complex data, is time consuming and considerably slows the processing in GATE. Below are descriptions of the integration considerations for each of the modules in SVENSK.

4.1. Tokenisation and Segmentation

Both the tokeniser and the sentence splitter for Swedish were designed with a future integration in GATE in mind, but may also be used as stand alone programs.

The tokeniser is implemented in C, employing `flex` (Levine et al., 1995), which is a tool for generating finite state lexical scanners. `flex` allows the developer to describe tokens in terms of regular expressions that it then translates to C code, utilized in the tokeniser. The tokeniser expects the input to be plain text, and the output is an annotation for each token in the input. The integration of the tokeniser in GATE was quite straightforward: since it returns each token together with the associated span of byte offsets, the wrapper itself does not have to calculate the offsets.

The sentence splitter is implemented in the Perl programming language. The process of integrating it in GATE differed somewhat from that of the tokeniser since the input to the splitter needs to be more structured. It requires the input to be tokens with the associated byte offsets, as output by the tokeniser. The output produced by the splitter is sentences together with the corresponding spans of byte offsets.

4.2. Tagging and Morphological Processing

The Brill tagger and SWECG predates GATE. However, since the I/O formats of these modules are well defined, it was possible to wrap the components up in Tcl code to have them “talk” to GATE and to the other components in the system.

SWECG does not have information of the exact position (in byte offset) of each word in the input text. Since the Annotation Model requires such information, a preprocessing procedure has been implemented. For each word a position, together with the other information of the word produced by SWECG, is stored in an annotation structure. The position finding procedure is not integrated into SWECG, but resides in the wrapper code.

The Brill tagger relies on information provided by the tokeniser and the sentence splitter; thus, the wrapper does not contain a preprocessing procedure for calculating byte offsets in the input. Unlike the Brill tagger included in the VIE system, the source code of the tagger for Swedish has not been altered, so this wrapper should enable integration of any tagger conforming to the I/O format of the original Brill tagger.

UCP assigns zero or more morphological analyses to each token in the input, each analysis appearing as an attribute-value structures spanning several lines. Thus, the wrapper is required to process a number of lines in the output before information about one token can be recorded. The UCP wrapper is therefore quite complex and the time spent in it is relatively longer than the time spent in the wrappers integrating the other components described.

To enable users to view the results from UCP, a new GATE-viewer for displaying the output from UCP was implemented and integrated in GATE. It takes a hypertext approach towards solving the problem of showing an arbitrary number of analyses to the user; when the user clicks on a portion of the input, displayed in one window, the corresponding analyses are shown in another window.

4.3. Grammatical Resources

DUP does not need any specific position finding procedure, since it takes the input positions from the output of SWECG. But changes were made to the source code so that the positions are propagated to the output of DUP.

The DUP module works with an internal lexicon. In addition, DUP gets lexical information from SWECG. Thus, DUP can get information from either one of the lexica, or both. Connecting independent linguistic algorithmic modules often requires a mapping between data structures treated by the different modules. In the case of connecting SWECG with DUP, the tagset of SWECG must be mapped

onto the feature structures of DUP.

There are several ways to transform data structures in GATE. One way is to transform the data in one of the modules, for example in the parser in DUP, or in the wrapper (as is done in the preprocessing module for SWECG). But then a communicating module is committed to always take input data of the same kind as the other module. If another module is to be connected, treating different types of data, the transformation must be made in the module itself.

Another way is to have a separate module that transforms between data sets. This is a dynamic approach since no module needs to be changed. Connecting another module means adding a new transformation module between the old and the new ones. Connecting SWECG to DUP, we have adopted the second approach, i.e., we have developed a separate module (called SWECG2SLE in Figure 1) which undertakes the necessary format conversions.

DSP treats the same data as SWECG outputs, so no transformation module between them is needed: The input to DSP is the word annotations from SWECG, and the output is the annotations of the semantic structures produced by DSP. As in the DUP module, some changes in the code of DSP had to be made, since the word positions must be propagated to the output.

5. Conclusions and Future Work

The paper has described the SVENSK project, which develops a toolbox of reusable language processing resources. SVENSK uses GATE, General Architecture for Text Engineering, as a platform in which the resources are integrated. A range of different modules have been incorporated into the system so far, both in-house modules, commercially available modules, and modules from Swedish academia.

In addition to the core components described in this paper, SVENSK can include language resources and auxiliary tools for corpus analysis, text pre-processing, testing, evaluation and so on, which have been developed at SICS or contributed by external sources. Some resources already available at SICS which would tentatively be included:

- S-VEX, the Swedish Vocabulary Expander (Gambäck, 1992), a tool for extending the fully-defined lexicon of the deep-level processing component.
- A generator component based on so-called Semantic-Head Driven Generation (Shieber et al., 1990).
- An interface component to a toy database (Gambäck and Ljung, 1993).
- The “parser-box”, a toolkit for educational purposes which includes several types of parsers.

A wide range of components from external sources could possibly be integrated, including another Swedish version of the Brill tagger developed at the Department of Swedish, Göteborg University (Nivre et al., 1996) and trained on the SUC corpus tag set.

The key aim of the project is that the resources included in SVENSK should be freely available for non-commercial

use, at least for Swedish institutions. Currently, all components except for SWECG and SWETWOL meet this requirement. Of course, processing resources included in the system in the future should preferably also match this free-for-all strategy.

6. Acknowledgements

SVENSK is funded by the Swedish National Board for Industrial and Technical Development (Nutek) and SICS. Charlotta Berglund, Scott McGlashan and Joel Sunnehall all contributed to parts of the project and Hamish Cunningham (U Sheffield) has provided excellent GATE support.

The SVENSK project is guided by a reference group composed of two academic and two industrial members, as well as a Nutek representative. The project has benefited in many ways from discussions with the members of this group, Lars Ahrenberg, Barbro Atlestam, Anna Sågvall-Hein, Carl-Wilhelm Welin, and Mats Wirén — and also by ideas from Ivan Bretan, Hercules Dalianis, Måns Engstedt, Jussi Karlgren, and Christer Samuelsson.

7. References

- Hiyan Alshawi and Jan van Eijck. 1989. Logical forms in the Core Language Engine. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Vancouver, British Columbia, June. ACL.
- Hiyan Alshawi, editor, David Carter, Jan van Eijck, Björn Gambäck, Robert C. Moore, Douglas B. Moran, Fernando C. N. Pereira, Stephen G. Pulman, Manny Rayner, and Arnold G. Smith. 1992. *The Core Language Engine*. The MIT Press, Cambridge, Massachusetts, March.
- Jonas Beskow, Kjell Elenius, and Scott MacGlashan. 1997. Olga — a dialogue system with an animated talking agent. In *Proceedings of the 5th European Conference on Speech Communication and Technology*, volume 3, pages 1651–1654, Rhodes, Greece, September.
- A. Bredenkamp, T. Declerck, F. Fouvry, B. Music, and A. Theofilidis. 1997. Linguistic engineering using ALEP. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing*, pages 92–97, Tzigov Chark, Bulgaria, September.
- Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, April. ACL.
- Thomas Bub and Johannes Schwinn. 1996. Verbmobil: The evolution of a complex large speech-to-speech translation system. In *Proceedings of the 4th International Conference on Spoken Language Processing*, Philadelphia, Pennsylvania, October.

- Philip R. Cohen, Adam J. Cheyer, Michelle Wang, and Soon Choel Baeg. 1994. An open agent architecture. In *AAAI Spring Symposium*, pages 1–8, Stanford, California, March.
- Hamish Cunningham, Kevin Humphreys, Robert J. Gaizauskas, and Martin Stower, 1996a. *CREOLE Developer's Manual*. University of Sheffield, England.
- Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. 1996b. GATE – a general architecture for text engineering. In *Proceedings of the 16th International Conference on Computational Linguistics*, volume 2, pages 1057–1060, København, Denmark, August. ACL.
- Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm–Umeå Corpus project. Report 33, Dept. of General Linguistics, University of Umeå, Sweden.
- Mikael Eriksson and Björn Gambäck. 1997. SVENSK: A toolbox of Swedish language processing resources. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing*, pages 336–341, Tzigo Chark, Bulgaria, September.
- Björn Gambäck and Stefan Ljung. 1993. Question answering in the Swedish Core Language Engine. In E. Sandewall and C. G. Jansson, editors, *Proceedings of the 4th Scandinavian Conference on Artificial Intelligence*, pages 212–225, Stockholm, Sweden, May. NORFA, IOS Press, Amsterdam, Holland.
- Björn Gambäck and Manny Rayner. 1992. The Swedish Core Language Engine. In L. Ahrenberg, editor, *Papers from the 3rd Nordic Conference on Text Comprehension in Man and Machine*, pages 71–85, Linköping University, Linköping, Sweden, April.
- Björn Gambäck. 1992. Lexical acquisition: the Swedish VEX system. In L. Ahrenberg, editor, *Papers from the 3rd Nordic Conference on Text Comprehension in Man and Machine*, pages 59–70, Linköping University, Linköping, Sweden, April.
- Björn Gambäck. 1997. *Processing Swedish Sentences: A Unification-Based Grammar and some Applications*. Doctor of Engineering Thesis, The Royal Institute of Technology and Stockholm University, Dept. of Computer and Systems Sciences, Sweden, June.
- Ralph Grishman et al., 1997. *TIPSTER Text Phase II Architecture Design. Version 2.3*. New York, New York, January.
- Kevin Humphreys, Robert J. Gaizauskas, Hamish Cunningham, and Saliha Azzam, 1996a. *VIE Technical Specifications*. University of Sheffield, England.
- Kevin Humphreys, Robert J. Gaizauskas, Hamish Cunningham, and Saliha Azzam, 1996b. *CREOLE Module Specifications*. University of Sheffield, England.
- Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin, Germany, January.
- Fred Karlsson. 1992. SWETWOL: A comprehensive morphological analyser for Swedish. *Nordic Journal of Linguistics*, 15(1):1–45.
- Kimmo Koskenniemi. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Doctor of Philosophy Thesis, University of Helsinki, Dept. of General Linguistics, Finland.
- John R. Levine, Tony Mason, and Doug Brown. 1995. *lex & yacc*. O'Reilly & Associates, Sebastopol, California.
- Douglas B. Moran, Adam J. Cheyer, Luc E. Julia, David L. Martin, and Sangkyu Park. 1997. Multimodal user interfaces in the Open Agent Architecture. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 61–68, Orlando, Florida, January. ACM.
- Joakim Nivre, Leif Grönqvist, Malin Gustafsson, Torbjörn Lager, and Sylvana Sofkova. 1996. Tagging spoken language using written language statistics. In *Proceedings of the 16th International Conference on Computational Linguistics*, volume 2, pages 1078–1081, København, Denmark, August. ACL.
- Fredrik Olsson. 1998. Tagging and morphological processing in the svensk system. Master of Art Thesis, Uppsala University, Sweden, March.
- Klas Prütz. 1997. Sammanställning av en träningskorpus på svenska för träning av ett automatiskt ordklassstagningssystem. (in Swedish).
- Christer Samuelsson. 1994. Notes on LR parser design. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 1, pages 386–390, Kyoto, Japan, August. ACL.
- Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16:30–43.
- N. Simpkins and M. Groenendijk. 1994. The ALEP project. Technical report, Cray Systems /CEC, Luxembourg.
- Joel Sunnehall. 1996. Robust parsing using dependency with constraints and preference. Master of Art Thesis, Uppsala University, Sweden, September.
- Anna Sågvalld-Hein, Ingrid Almqvist, and Per Starbäck. 1997. Scania Swedish – a basis for multilingual machine translation. In *Proceedings of the 19th Conference on Translating and the Computer*, London, England, November. ASLIB.
- Anna Sågvalld-Hein. 1981. An overview of the Uppsala Chart Parser version 1 (UCP-1). Technical report, Center for Computational Linguistics, Uppsala University, Sweden.