Finite-State Methods and Natural Language Processing FSMNLP 2005

The Fifth Volume in the Series of International Workshops
on Finite-State Methods in Natural Language Processing

Abstracts of the Workshop

Edited by:
A. Yli-Jyrä, L. Karttunen, J. Karhumäki

Organization

FSMNLP 2005 is organized by the Department of General Linguistics, University of Helsinki in cooperation with CSC, the Finnish IT Centre for Science.

Invited Speakers

Tero Harju	University of Turku, Finland
Lauri Karttunen	Palo Alto Research Center, Stanford University, USA

Program Committee

Steven Bird	University of Melbourne, Australia
Francisco Casacuberta	Universitat Politècnica de València, Spain
Jean-Marc Champarnaud	Université de Rouen, France
Jan Daciuk	Gdansk University of Technology, Poland
Jason Eisner	Johns Hopkins University, USA
Tero Harju	University of Turku, Finland
Arvi Hurskainen	IAAS, University of Helsinki, Finland
Juhani Karhumäki, *co-chair*	University of Turku, Finland
Lauri Karttunen, *co-chair*	PARC and Stanford University, USA
André Kempe	Xerox Research Centre Europe, France
George Anton Kiraz	Beth Mardutho: The Syriac Institute, USA
András Kornai	Budapest Institute of Technology, Hungary
D. Terence Langendoen	University of Arizona, USA
Eric Laporte	Université de Marne-la-Vallée, France
Mike Maxwell	Linguistic Data Consortium, USA
Mark-Jan Nederhof	University of Groningen, the Netherlands
Gertjan van Noord	University of Groningen, the Netherlands
Kemal Oflazer	Sabanci University, Turkey
Jean-Eric Pin	CNRS/University Paris 7, France
James Rogers	Earlham College, USA
Giorgio Satta	University of Padua, Italy
Jacques Sakarovitch	CNRS/ENST, France
Richard Sproat	University of Illinois at Urbana-Champaign, USA
Nathan Vaillette	University of Tübingen, Germany
Atro Voutilainen	Connexor Oy, Finland
Bruce W.Watson	University of Pretoria, South Africa
Shuly Wintner	University of Haifa, Israel
Sheng Yu	University of Western Ontario, Canada
Lynette van Zijl	Stellenbosch University, South Africa

Organizing Committee

Anssi Yli-Jyrä, *chair*	CSC, Finalnd
Hanna-Maria Westerlund	University of Helsinki, Finland
Sari Hyvärinen	University of Helsinki, Finland
Antti Arppe	University of Helsinki, Finland

Steering Committee I (FSMNLP traditions)

Lauri Karttunen	PARC and Stanford University, USA
Kimmo Koskenniemi	University of Helsinki, Finland
Gertjan van Noord	University of Groningen, the Netherlands
Kemal Oflazer	Sabanci University, Turkey

Steering Committee II (local aspects)

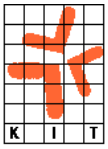
Lauri Carlson	University of Helsinki, Finland
Tero Harju	University of Turku, Finland
Lauri Hella	University of Tampere, Finland
Arvi Hurskainen	University of Helsinki, Finland
Fred Karlsson	University of Helsinki, Finland
Krista Lagus	Helsinki University of Technology, Finland
Kerkko Luosto	University of Helsinki, Finland
Matti Nykänen	University of Helsinki, Finland

Additional Referees

Rafael C. Carrasco	Universitat d'Alacant, Spain
Loek Cleophas	Technische Universiteit Eindhoven, The Netherlands
Yvon Francois	GET/ENST and LTCI, France
Ernest Ketcha Ngassam	University of South Africa and University of Pretoria, South Africa
Ines Klimann	Universite Paris 7, France
Sylvain Lombardy	Universite Paris 7, France
David Picó-Vila	Universidad Politécnica de Valencia, Spain
Enrique Vidal	Universidad Politécnica de Valencia, Spain
Juan Miguel Vilar	Universitat Jaume I, Spain
M. Inés Torres	Universidad País Vasco, Spain
Anssi Yli-Jyrä	University of Helsinki, Finland

Sponsoring Institutions

CSC, the Finnish IT center for science, Finland
University of Helsinki, Finland
The KIT Network, Finland
Academy of Finland
Connexor Ltd., Finland
Lingsoft Ltd., Finland



Official Proceedings

Revised selected papers will appear in the official FSMNLP 2005 proceedings, to be published by Springer in the series of Lecture Notes in Artificial Intelligence.

Table of Contents

Invited Papers

Characterizations of Regularity 1
Tero Harju

Finnish Optimality-Theoretic Prosody 2
Lauri Karttunen

Contributed Papers

Partitioning Multitape Transducers 3
François Barthélemy

Squeezing the Infinite into the Finite 4
Tamás Bíró

A Novel Approach to Computer-Assisted Translation based on Finite-State Transducers 5
Jorge Civera, Juan M. Vilar, Elsa Cubel, Antonio L. Lagarda, Sergio Barrachina, Francisco Casacuberta, Enrique Vidal

Finite State Registered Automata and their uses in Natural languages 6
Yael Cohen-Sygal, Shuly Wintner

TAGH: A Complete Morphology for German based on Weighted Finite State Automata 7
Alexander Geyken, Thomas Hanneforth

Klex: A Finite-State Transducer Lexicon of Korean 8
Na-Rae Han

Longest-Match Pattern Matching with Weighted Finite State Automata 9
Thomas Hanneforth

Finite-state syllabification 10
Mans Hulden

Algorithms for Minimum Risk Chunking 11
Martin Jansche

Collapsing ϵ-Loops in Weighted Finite-State Machines 12
J Howard Johnson

WFSM Auto-Intersection and Join Algorithms 13
André Kempe, Jean-Marc Champarnaud, Franck Guingne, Florent Nicart

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop

Further Results on Syntactic Ambiguity of Internal Contextual Grammars 14
Kuppusamy Lakshmanan

Error-Driven Learning with Bracketing Constraints 15
Takashi Miyata, Kôiti Hasida

Parsing with Lexicalized Probabilistic Recursive Transition Networks 16
Alexis Nasr, Owen Rambow

Integrating a POS Tagger and a Chunker Implemented as Weighted Finite State Machines 17
Alexis Nasr and Alexandra Volanschi

Modelling the Semantics of Calendar Expressions as Extended Regular Expressions 18
Jyrki Niemi, Lauri Carlson

Using Finite State Technology in a Tool for Linguistic Exploration 19
Kemal Oflazer, Mehmet Dinçer Erbaş, Müge Erdoğmuş

Applying a Finite Automata Acquisition Algorithm to Named Entity Recognition 20
Muntsa Padró and Lluís Padró

Principles, Implementation Strategies, and Evaluation of a Corpus Query System 21
Ulrik Petersen

On Compact Storage Models for Gazetteers 22
Jakub Piskorski

German Compound Analysis with *wfsc* 23
Anne Schiller

Scaling an Irish FST morphology engine for use on unrestricted text 24
Elaine Uí Dhonnchadha, Josef Van Genabith

Improving Inter-Level Communication in Cascaded Finite-State Partial Parsers . 25
Sebastian van Delden, Fernando Gomez

Pivotal Synchronization Expressions: A Formalism for String Alignments 26
Anssi Yli-Jyrä, Jyrki Niemi

Abstracts of Interactive Presentations

A complete FS model for Amharic morphographemics 27
Saba Amsalu and Dafydd Gibbon

Tagging with Delayed Disambiguation 29
José M. Castaño, James Pustejovsky

A New Algorithm for Unsupervised Induction of Concatenative Morphology . . . 31
Harald Hammarström

Morphological Parsing of Tone: An Experiment with Two-Level Morphology on the Ha language 33
Lotta Harjula

Describing Verbs in Disjoining Writing Systems 35
Arvi Hurskainen, Louis Louwrens, George Poulos

An FST grammar for verb chain transfer in a Spanish-Basque MT System 38
Iñaki Alegria, Arantza Díaz de Ilarraza, Gorka Labaka, Mikel Lersundi, Aingeru Mayor and Kepa Sarasola

Finite state transducers based on k-TSS grammars for speech translation 40
Alicia Pérez, Francisco Casacuberta, M. Inés Torres, Víctor Guijarrubia

Abstracts of Software Demos

Unsupervised Morphology Induction Using Morfessor 43
Mathias Creutz, Krista Lagus, Sami Virpioja

SProUT – a General-Purpose NLP Framework Integrating Finite-State and Unification-based Grammar Formalisms 44
Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer

Tool Demonstration: Functional Morphology 46
Markus Forsberg, Aarne Ranta

From Xerox to Aspell: A First Prototype of a North Sámi Speller Based on TWOL Technology 48
Børre Gaup, Sjur Moshagen, Thomas Omma, Maaren Palismaa, Tomi Pieski, Trond Trosterud

A Programming Language For Finite State Transducers 50
Helmut Schmid

FIRE Station 52
Bruce Watson

Author Index 53

Characterizations of Regularity

Tero Harju

Department of Mathematics, University of Turku, Finlan

Regular languages have many different characterizations in terms of automata, congruences, semigroups **etc**. In this talk we have a look at the more recent result, obtained during the last two decades, namely characterizations using morphic compositions, equality sets and well ordered structures.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, p. 1

Finnish Optimality-Theoretic Prosody

Lauri Karttunen

Palo Alto Research Center, Stanford University, USA

A well-known phenomenon in Finnish prosody is the alternation of binary and ternary feet. In native Finnish words, the primary stress falls on the first syllable. Secondary stress generally falls on every second syllable: (vói.mis).(tè.li).(jòi.ta) 'gymnasts' creating a sequence of trochaic binary feet. However, secondary stress skips a light syllable that is followed by a heavy syllable. In (vói.mis.te).(lèm.me) 'we are doing gymnastics', the first foot is ternary, a dactyl.

Within the context of Optimality Theory (OT, Prince and Smolensky 1993), it has been argued that prosodic phenomena are best explained in terms of universal metric constraints. OT constraints can be violated; no word can satisfy all of them. A language-specific ranking of the constraints makes some violations less important than others. In her 1999 dissertation, A unified account of binary and ternary stress, Nine Elenbaas gives an analysis of Finnish in which the alternation between binary and ternary feet follows as a side effect of the ordering of two particular constraints, *Lapse and *(L'. H) The *Lapse constraint stipulates that an unstressed syllable must be adjacent to a stressed syllable or to word edge. The *(L'. H) constraint prohibits feet such as (tè.lem) where a light stressed syllable is followed by a heavy unstressed syllable. The latter constraint of course is outranked by the constraint that requires initial stress on the first syllable in Finnish regardless of the its weight. In his 2003 article on Finnish Noun Inflection, Paul Kiparsky gives essentially the same account of the binary/ternary alternation except that he replaces the *(L'.H) rule by a more general StressToWeight constraint.

Although OT constraints themselves can be expressed in finite-state terms, Optimality Theory as a whole is not a finite-state model if it involves unbounded counting of constraint violations (Frank and Satta 1998). With that limitation OT analyses can be modelled with finite-state tools. In this paper we will give a full computational implementation of the Elenbaas and Kiparsky analyses using the extended regular expression calculus from the 2003 Beesley & Karttunen book on Finite State Morphology. Surprisingly, it turns out that Elenbaas and Kiparsky both make some incorrect predictions. For example, according to their accounts a word such as kalasteleminen 'fishing' should begin with a ternary foot: (ká.las.te).(lè.mi).nen. The correct footing is (ká.las).(tè.le).(mì.nen). There may of course be some ranking of OT constraints under which the binary/ternary alternation in Finnish comes "for free". It does not emerge from the Elenbaas and Kiparsky analyses.

This case study illustrates a more general point: Optimality Theory is computationally difficult and OT theorists are much in the need of computational help.

Partitioning Multitape Transducers

François Barthélemy

CNAM-Cédric, 292 rue Saint-Martin, F-75003 Paris, France
and INRIA, domaine de Voluceau, F-78153 Rocquencourt cedex, France
barthe@cnam.fr

Abstract. In this paper, we define a class of transducers closed under intersection and complementation, which are the operations used for contextual rule compilation. This class of transducers is not theoretically more powerful than the Epsilon-Free Letter Transducers most commonly used. But they are more convenient for morphological description whenever the correspondence between lexical and surface forms is not a symbol-to-symbol matching.
A complete set of operations on transducers is defined, including some operations (projection and join) which change the number of tapes of the transducers.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 3 – 3

Squeezing the Infinite into the Finite

Handling the OT Candidate Set with Finite State Technology

Tamás Bíró

Humanities Computing, University of Groningen
t.s.biro@rug.nl

Abstract. Finite State approaches to Optimality Theory have had two goals. The earlier and less ambitious one was to compute the optimal output by compiling a finite state automaton for each underlying representation. Newer approaches aimed at realizing the OT-systems as FS transducers mapping any underlying representation to the corresponding surface form. After reviewing why the second one fails for most linguistically interesting cases, we use its ideas to accomplish the first goal. Finally, we present how this approach could be used in the future as a—hopefully cognitively adequate—model of the mental lexicon.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 4 – 4

A Novel Approach to Computer-Assisted Translation based on Finite-State Transducers*

Jorge Civera[1], Juan M. Vilar[2], Elsa Cubel[1], Antonio L. Lagarda[1], Sergio Barrachina[2], Francisco Casacuberta[1], and Enrique Vidal[1]

[1] Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Instituto Tecnológico de Informática, E-46071 València, Spain
tt2iti@iti.upv.es
[2] Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I, E-12071 Castellón de la Plana, Spain

Abstract. Computer-Assisted Translation (CAT) is an alternative approach to Machine Translation, that integrates human expertise into the automatic translation process. In this framework, a human translator interacts with a translation system that dynamically offers a list of translations that best completes the part of the sentence already translated. Stochastic finite-state transducer technology is proposed to support this CAT system. The system was assessed on two real tasks of different complexity in several languages.

* This work has been supported by the European Union under the IST Programme (IST-2001-32091) and the Spanish project TIC2003-08681-C02.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 5 – 5

Finite State Registered Automata and their uses in Natural languages

Yael Cohen-Sygal and Shuly Wintner

Department of Computer Science
University of Haifa
{yaelc,shuly}@cs.haifa.ac.il

Abstract. We extend *finite state registered automata* (FSRA) to account for medium-distance dependencies in natural languages. We provide an extended regular expression language whose expressions denote arbitrary FSRAs and use it to describe some morphological and phonological phenomena. We also define several dedicated operators which support an easy and efficient implementation of some non-trivial morphological phenomena. In addition, we extend FSRA to *finite-state registered transducers* and demonstrate their space efficiency.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 6 – 6

TAGH: A Complete Morphology for German based on Weighted Finite State Automata

Alexander Geyken[1] and Thomas Hanneforth[2]

[1] Berlin-Brandenburg Academy of Sciences
[2] University of Potsdam

Abstract. TAGH is a system for automatic recognition of German word forms. It is based on a stem lexicon with allomorphs and a concatenative mechanism for inflection and word formation. Weighted FSA and a cost function are used in order to determine the correct segmentation of complex forms: the correct segmentation for a given compound is supposed to be the one with the least cost. TAGH is based on a large stem lexicon of almost 80.000 stems that was compiled within 5 years on the basis of large newspaper corpora and literary texts. The number of analyzable word forms is increased considerably by more than 1000 different rules for derivational and compositional word formation. The recognition rate of TAGH is more than 99% for modern newspaper text and approximately 98.5% for literary texts.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 7 – 7

Klex: A Finite-State Transducer Lexicon of Korean

Na-Rae Han

Department of Linguistics, University of Pennsylvania, Philadelphia, PA 19104, USA,
nrh@ling.upenn.edu,
WWW home page: http://www.cis.upenn.edu/∼nrh/klex.html

Abstract. This paper describes the implementation and system details of Klex, a finite-state transducer lexicon for the Korean language, developed using XRCE's Xerox Finite State Tool (XFST). Klex is essentially a transducer network representing the lexicon of the Korean language with the lexical string on the upper side and the inflected surface string on the lower side. Two major applications for Klex are morphological analysis and generation: given a well-formed inflected lower string, a language-independent algorithm derives the upper lexical string from the network and vice versa. Klex was written to conform to the part-of-speech tagging standards of the Korean Treebank Project, and is currently operating as the morphological analysis engine for the project.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 8 – 8

Longest-Match Pattern Matching with Weighted Finite State Automata

Thomas Hanneforth

Universität Potsdam, Institut für Linguistik, PF 601553, 14415 Potsdam, Germany
tom@ling.uni-potsdam.de

Abstract. I present a new method of longest match pattern matching based on weighted finite state automata. Contrary to the approach of Karttunen [1] we do not need expensive complementation operations to construct our pattern matching transducer.

References

1. Karttunen, L. Directed Replacement. In: Proceedings of the 34rd Annual Meeting of the ACL, Santa Cruz, CA, 1996.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 9 – 9

Finite-state syllabification

Mans Hulden

The University of Arizona
Department of Linguistics
PO BOX 210028
Tucson AZ, 85721-0028
USA
mhulden@email.arizona.edu

Abstract. We explore general strategies for finite-state syllabification and describe a specific implementation of a wide-coverage syllabifier for English, as well as outline methods to implement differing ideas about the English syllable encountered in the phonological literature. The syllable is a central phonological unit to which many allophonic variations are sensitive. How a word is syllabified is a non-trivial problem and reliable methods are useful in computational systems that deal with non-orthographic representations of language, for instance phonological research, text-to speech systems, and speech recognition.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 10 – 10

Algorithms for Minimum Risk Chunking

Martin Jansche

Center for Computational Learning Systems
Columbia University, New York

Abstract. Stochastic finite automata are useful for identifying substrings (chunks) within larger units of text. Relevant applications include tokenization, base-NP chunking, named entity recognition, and other information extraction tasks. For a given input string, a stochastic automaton represents a probability distribution over strings of labels encoding the location of chunks. For chunking and extraction tasks, the quality of predictions is evaluated in terms of precision and recall of the chunked/extracted phrases when compared against some gold standard. However, traditional methods for estimating the parameters of a stochastic finite automaton and for decoding the best hypothesis do not pay attention to the evaluation criterion, which we take to be the well-known F-measure. We are interested in methods that remedy this situation, both in training and decoding. Our main result is a novel algorithm for efficiently evaluating expected F-measure. We present the algorithm and discuss its applications for utility/risk-based parameter estimation and decoding.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 11 – 11

Collapsing ϵ-Loops in Weighted Finite-State Machines

J Howard Johnson

Institute for Information Technology,
National Research Council Canada,
Ottawa Canada,
Howard.Johnson@nrc-cnrc.gc.ca

Abstract. Weighted finite-state automata pose a number of challenges for software developers. One particular difficulty is that ϵ-transitions must be treated more carefully than is necessary for unweighted automata. The usual weighted ϵ-closure algorithm always produces $O(n^2)$ transitions for a ϵ-loop with n states. An approach that removes ϵ-loops without performing a full ϵ-closure is proposed and it is shown how this can be efficiently implemented using sparse matrix operations.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 12 – 12

WFSM Auto-Intersection and Join Algorithms

A. Kempe[1], J.-M. Champarnaud[2], F. Guingne[1,3], F. Nicart[1,3]

[1] Xerox Research Centre Europe – Grenoble Laboratory
6 chemin de Maupertuis – 38240 Meylan – France
Andre.Kempe@xrce.xerox.com – http://www.xrce.xerox.com

[2] PSI Laboratory (Université de Rouen, CNRS)
76821 Mont-Saint-Aignan – France
Jean-Marc.Champarnaud@univ-rouen.fr
– http://www.univ-rouen.fr/psi/

[3] LIFAR Laboratory (Université de Rouen)
76821 Mont-Saint-Aignan – France
{Franck.Guingne,Florent.Nicart}@univ-rouen.fr
http://www.univ-rouen.fr/LIFAR/

Abstract. The join of two n-ary string relations is a main operation regarding to applications. n-Ary rational string relations are realized by weighted finite-state machines with n tapes. We provide an algorithm that computes the join of two machines via a more simple operation, the auto-intersection. The two operations generally do not preserve rationality. A delay-based algorithm is described for the case of a single tape pair, as well as the class of auto-intersections that it handles. It is generalized to multiple tape pairs and some enhancements are discussed.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 13 – 13

Further Results on Syntactic Ambiguity of Internal Contextual Grammars

K. Lakshmanan

School of Technology and Computer Science
Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba
Mumbai - 400 005, India
laksh@tifr.res.in

Abstract. Ambiguity plays an important role in checking the relevances of formalism for natural language processing. As contextual grammars were shown to be an appropriate description for natural languages, analyzing syntactic ambiguity of contextual grammars deserves a special attention. In this paper, we continue the study on ambiguity of internal contextual grammars which was investigated in [1] and [2]. We achieve solutions to the following open problems addressed in the above papers. For each $(i, j) \in \{(2, 1), (1, 0), (0, 1)\}$, are there languages which are inherently i-ambiguous with respect to grammars with arbitrary selector, but j-ambiguous with respect to grammars with finite selector? Does the statement hold for deterministic contextual grammars too?

References

1. L. Ilie. On Ambiguity in Internal Contextual Languages. *II Intern. Conf. Math. Linguistics '96*, Tarragona, (C. Martin-Vide ed.), John Benjamins, 1997, 29–45.
2. C. Martin-Vide, J. Miguel-Verges, Gh. Păun and A. Salomaa. Attempting to Define the Ambiguity in Internal Contextual Languages. *II Intern. Conf. Math. Linguistics '96*, Tarragona, (C. Martin-Vide ed.), John Benjamins, Amsterdam, 1997, 59–81.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 14 – 14

Error-Driven Learning with Bracketing Constraints

Takashi Miyata[1] and Kôiti Hasida[2,1]

[1] Core Research for Evolutional Science and Technology,
Japan Science and Technology Agency
[2] Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology

Abstract. A chunking algorithm with a Markov model is extended to accept bracketing constraints. The extended algorithm is implemented by modifying a state-of-the-art Japanese dependency parser. Then the effect of bracketing constraints in preventing parsing errors is evaluated. A method for improving the parser's accuracy is proposed. That method adds brackets according to a set of optimal brackets obtained from a training corpus. Although the method's coverage is limited, the F-measure for the sentences to which the method adds brackets is improved by about 7%.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 15 – 15

Parsing with Lexicalized Probabilistic Recursive Transition Networks

Alexis Nasr and Owen Rambow

[1] Lattice-CNRS (UMR 8094),
Université Paris 7, Paris, France
alexis.nasr@linguist.jussieu.fr
[2] Center for Computational Learning Systems,
Columbia University, New York, NY, USA
rambow@cs.columbia.edu

Abstract. We present a formalization of lexicalized Recursive Transition Networks which we call Automaton-Based Generative Dependency Grammar (GDG). We show how to extract a GDG from a syntactically annotated corpus, present a chart parser for GDG, and discuss different probabilistic models which are directly implemented in the finite automata and do not affect the parser.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 16 – 16

Integrating a POS Tagger and a Chunker Implemented as Weighted Finite State Machines

Alexis Nasr and Alexandra Volanschi*

LATTICE-CNRS (UMR 8094)4, Université Paris 7
alexis.nasr, alexandra.volanschi}@linguist.jussieu.fr

Abstract. This paper presents a method of integrating a part-of-speech tagger and a chunker. This integration lead to the correction of a number of errors made by the tagger when used alone. Both tagger and chunker are implemented as weighted finite state machines. Experiments on a French corpus showed a decrease of the word error rate of about 12%.
Keywords : Part-of-speech tagging, chunking, weighted finite state machines...

* This work was partly funded by the Project WATSON - Technolangue.

Modelling the Semantics of Calendar Expressions as Extended Regular Expressions

Jyrki Niemi and Lauri Carlson

University of Helsinki, Department of General Linguistics,
PO Box 9, FI–00014 University of Helsinki, Finland
{jyrki.niemi,lauri.carlson}@helsinki.fi

Abstract. This paper proposes modelling the semantics of natural-language calendar expressions as extended regular expressions (XREs). The approach covers expressions ranging from plain dates and times of the day to more complex ones, such as *the second Tuesday following Easter*, including expressions denoting disconnected periods of time. The paper presents one possible underlying string-based temporal model, sample calendar expressions with their representations as XREs, and possible applications in reasoning and natural-language generation.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 18 – 18

Using Finite State Technology in a Tool for Linguistic Exploration

Kemal Oflazer, Mehmet Dinçer Erbaş, Müge Erdoğmuş

Faculty of Engineering and Natural Sciences
Sabancı University
Tuzla, Istanbul, Turkey 34956

oflazer@sabanciuniv.edu
{derbas,mugeerdogmus}@su.sabanciuniv.edu

Abstract. Intelligent, interactive and pervasively accessible tools for providing information about elements of a language are crucial in learning a language, especially in an advanced secondary language learning setting and learning for linguistic exploration. The paper describes a prototype implementation of a tool that provides intelligent, active and interactive tools for helping linguistics students inquire and learn about lexical and syntactic properties of words and phrases in Turkish text. The tool called LINGBROWSER uses extensive finite state language processing technology to provide instantaneous information about morphological, segmental, pronunciation properties about the words in any real text. Additional resources also provide access to semantic properties of (root) words.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 19 – 19

Applying a Finite Automata Acquisition Algorithm to Named Entity Recognition

Muntsa Padró and Lluís Padró

TALP Research Center
Universitat Politècnica de Catalunya
{mpadro,padro}@lsi.upc.edu

Abstract. In this work, Causal-State Splitting Reconstruction algorithm, originally conceived to model stationary processes by learning finite state automata from data sequences, is for the first time applied to NLP tasks, namely Named Entity Recognition. The obtained results are slightly below the best systems presented in CoNLL 2002 shared task, though given the simplicity of the used features, they are really promising.
Once the viability of using this algorithm for NLP tasks is stated, we plan to improve the results obtained at NER task, as well as to apply it to other NLP sequence recognition tasks such as PoS tagging, chunking, subcategorization patterns acquisition, etc.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 20 – 20

Principles, Implementation Strategies, and Evaluation of a Corpus Query System

Ulrik Petersen

University of Aalborg
Department of Communication, Kroghstræde 3
DK — 9220 Aalborg East, Denmark
ulrikp@hum.aau.dk
http://emdros.org/

Abstract. The last decade has seen an increase in the number of available corpus query systems. These systems generally implement a query language as well as a database model. We report on one such corpus query system, and evaluate its query language against a range of queries and criteria quoted from the literature. We show some important principles of the design of the query language, and argue for the strategy of separating what is retrieved by a linguistic query from the data retrieved in order to display or otherwise process the results, stating the needs for generality, simplicity, and modularity as reasons to prefer this strategy.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 21 – 21

On Compact Storage Models for Gazetteers

Jakub Piskorski

DFKI GmbH
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
Jakub.Piskorski@dfki.de

Abstract. This paper describes compact storage models for gazetteers using state-of-the-art finite-state technology. In particular, we compare the standard method based on numbered indexing automata associated with an auxiliary storage device with a pure finite-state representation, the latter being superior in terms of space and time complexity when applied to real-world test data. Further, we pinpoint some pros and cons for both approaches and provide results of empirical experiments which form handy guidelines for selecting a suitable data structure for implementing a gazetteer.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 22 – 22

German Compound Analysis with *wfsc*

Anne Schiller

Xerox Research Centre Europe
6 chemin de Maupertuis, 38250 Meylan, France
Anne.Schiller@xrce.xerox.com

Abstract. Compounding is a very productive process in German to form complex nouns and adjectives which represent about 7% of the words of a newspaper text. Unlike English, German compounds do not contain spaces or other word boundaries, and the automatic analysis is often ambiguous. A (non-weighted) finite-state morphological analyzer provides all potential segmentations for a compound without any filtering or prioritization of the results.
The paper presents an experiment in analyzing German compounds with the Xerox Weighted Finite-State Compiler (wfsc). The model is based on weights for compound segments and gives priority (a) to compounds with the minimal number of segments and (b) to compound segments with the highest frequency in a training list. The results with this rather simple model will show the advantage of using weighted finite-state transducers over simple FSTs.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 23 – 23

Scaling an Irish FST morphology engine for use on unrestricted text

Elaine Uí Dhonnchadha[1,2] and Josef Van Genabith[1]

[1]National Centre for Language Technology, School of Computing
Dublin City University, Dublin 9, Ireland
{euidhonn, josef}@computing.dcu.ie
[2]Institiúid Teangeolaíochta Éireann, 31 Plás Mhic Liam
Baile Átha Cliath 2, Ireland

Abstract. This paper details the steps involved in scaling-up a lexicalised finite-state morphology transducer for use on unrestricted text. Our starting point was a base-line inflectional morphology engine [1], with 81% token coverage measured against a 15 million word corpus of Irish texts [2]. Manually scaling the FST lexicon component of a morphology transducer is time-consuming, expensive and rarely, if ever, complete. In order to scale up the engine we used a combination of strategies including semi-automatic population of the finite-state lexicon from machine-readable dictionary resources and from printed re- sources using optical character recognition, the addition of derivational morphology and the development of morphological guessers. This paper details the coverage increase contributed by each step. The full system achieves token coverage of 93% which is extended to 100% through the use of morphological guessers.

References

1. Uí Dhonnchadha, E.: An analyser and generator for irish inflectional morphology using finite state transducers. Master's thesis, School of Computing, Dublin City University, Dublin, Ireland (2002)
2. ITÉ: Corpas Náisiúnta na Gaeilge. ITÉ (2003)

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 24 – 24

Improving Inter-Level Communication in Cascaded Finite-State Partial Parsers

Sebastian van Delden[1] and Fernando Gomez[2]

[1] Division of Mathematics and Computer Science
University of South Carolina Upstate
800 University Way, Spartanburg SC 29303, USA
svandelden@uscupstate.edu
[2] Department of Computer Science
University of Central Florida
Orlando FL 32816, USA
gomez@cs.ucf.edu

Abstract. An improved inter-level communication strategy that enhances the capabilities of cascaded finite-state partial parsing systems is presented. Cascaded automata are allowed to make forward calls to other automata in the cascade as well as backward references to previously identified groupings. The approach is more powerful than a design in which the output of the current level is simply passed to the next level in the cascade. The approach is evaluated on randomly extracted sentences from the Encarta encyclopedia. A discussion of related research is also presented.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 25 – 25

Pivotal Synchronization Expressions: A Formalism for String Alignments

Anssi Yli-Jyrä[1] and Jyrki Niemi[2]

[1] Language Bank Service, CSC Scientific Computing Ltd., Finland
[2] Department of General Linguistics, University of Helsinki, Finland
{aylijyra,janiemi}@ling.helsinki.fi

Abstract. Three very different formalisms, *generalized synchronization expressions* (GSEs) [1], *interleave-disjunction-lock (IDL) expressions* [2] and *generalized restrictions* (GRs) [3] have been used for specifying constraints on parallel or interleaved processes, phrases and bracketings. The current paper synthesizes some key ideas of these formalisms in a single framework, *pivotal synchronization expressions* (PSEs). The current paper sketches the semantics of these expressions in terms of regular st-languages [1] whose words are partioned into sequences of *pivots*. We anticipate that PSEs can be used to give descriptions for many kinds of regular string relations in natural language processing.

References

1. Salomaa, K., Yu, S.: Synchronization expressions with extended join operation. Theoretical Computer Science **207** (1998) 73–88
2. Nederhof, M.J.: IDL-expressions: A formalism for representing and parsing finite languages in natural language processing. Journal of Artificial Intelligence Research **21** (2004) 287–317
3. Yli-Jyrä, A.M., Koskenniemi, K.: Compiling contextual restrictions on strings into finite-state automata. In Cleophas, L., Watson, B.W., eds.: The Eindhoven FASTAR Days, Proceedings. Computer Science Reports 04/40, Eindhoven, The Netherlands, Technische Universiteit Eindhoven (2004)

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 26 – 26

A complete FS model for Amharic morphographemics

Saba Amsalu and Dafydd Gibbon

Universität Bielefeld, Germany

Our aim was to develop a complete morphographemic model for Amharic, the official language of Ethiopia, which urgently needs computational linguistic tools for information retrieval and natural language processing. Amharic is a Semitic language, with SOV word order and a complex morphology with consonantal roots and vowel intercalation, extensive agglutination, and both consonantal and vocalic stem modification. Previous computational models of Amharic lexemes are fragmentary, being restricted to affix stripping and radical extraction [2], [4], [3], [1], The verb analysis by [8] is the only previous FS based approach. FS and related approaches to other Semitic languages have also tended to concentrate on selected features of theoretical interest, such as the well–known analyses of Arabic intercalation [9], [5], [10].

In contrast, we have developed the first complete FS generator/analyser of Amharic morphology for all parts of speech (POS), including loan and native noun morphology, biradical, triradical and quadradical verb root generation, with vowel intercalation, conditioned internal vowel changes, agglutinative affixation of 13 affix classes, and full and partial reduplication. Phonological gemination is not represented in the Ethiopian Fidel orthography, and thus is not implemented.

Our development approach is linguistic rather than statistical, and includes novel features for modelling intercalation and reduplication. the analysis results are evaluated for precision and recall. The software used is XFST, with SERA (System for Ethiopian Representation in ASCII) romanisation. A port to Fidel Unicode is in progress.

Part of the system architecture is outlined in the activity diagram in Figure 1, which shows the FST verb cascade in generation direction, but is interpretable in both directions. Biradicals are generated from triradicals and quadradicals are independently generated; cf. [11], [6], [7], then vowels are intercalated, affixes are concatenated and phonological alternations processed.

Amharic has noun stem reduplication (with epenthetic vowel) (cf. Figure 2). A shell wrapper outside the FS system feeds XFST with a stream of words; the actual reduplication is then performed in the FS context using a novel bracketing ‘diacritic’ convention (not ‘flag diacritic’ [5]). Formally, this is a heuristic which treats the surface lexicon as the union of singleton sets of surface forms and applies the reduplication FST to the singleton sets individually.

For evaluation purposes we generate/analyse all POS separately. The FSTs for each POS are not unioned, because the individual FSTs are to be integrated into an FST chunk parser/tagger. Each POS is evaluated individually on a test corpus for standard recall and precision scores (ambiguity scores are currently implicit in the precision values). Recall/precision values for small finite POS sets are, trivially, 1/1; verbs attain 0.94/0.54, nouns attain 0.85/0.94, and adjectives 0.88/0.81. The lower precision value for verbs is due to affix ambiguities (morphological syncretism).

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 27 – 28

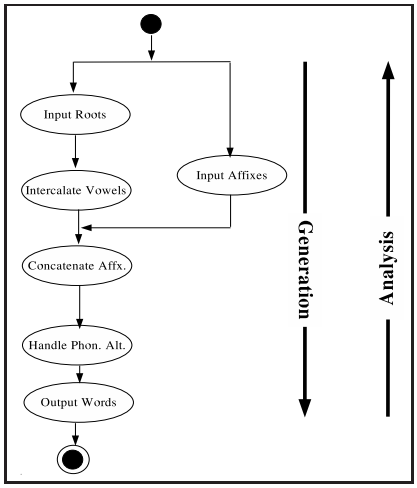


Fig. 1. FST cascade architecture.

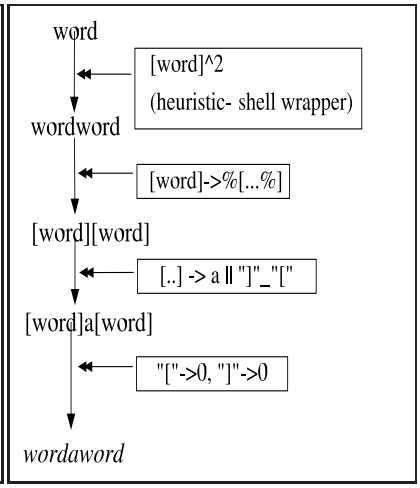


Fig. 2. Reduplication cascade.

References

1. Nega Alemayehu and Peter Willett. Stemming of amharic words for information retrieva. *Literary and Linguistic computing*, 17(1):1–17, 2002.
2. Lars Asker Atalech Alemu and Gunnar Eriksson. Building an amharic lexicon from parallel texts. In *Proceedings of: First Steps for Language Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, a workshop at LREC*, Lisbon, 2004.
3. Abiyot Bayou. Developing automatic word parser for amharic verbs and their derivation. Master's thesis, Addis Ababa University, Addis Ababa, 2000.
4. Tesfaye Bayu. Automatic morphological analyzer for amharic: An experiment involving unsupervised learning and autosegmental analysis approaches. Master's thesis, Addis Ababa University, Addis Ababa, 2002.
5. Ken Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI, Stanford, 2003.
6. M. Lionel Bender, Hailu Fulas, and C. H. Dawkins. *Amharic Verb Morphology*. Michigan State University, African Studies Center, East Lansing, 1978.
7. C. H. Dawkins. *The Fundamentals of Amharic*. Sudan Interior Mission, Addis Ababa, Ethiopia, 1960.
8. Sisay Fissaha and Johann Haller. Amharic verb lexicon in the context of machine translation. *TALN*, 2003.
9. Martin Kay. Nonconcatenative finite–state morphology. In *EACL Proceedings*, pages 2–10, 1987.
10. Sabine Reinhard and Dafydd Gibbon. Prosodic inheritance and morphological generalisations. In *Proceedings of EACL*, 1991.
11. Baye Yimam. Root reductions and extensions in amharic. *Ethiopian Journal of Languages and Literature*, (9), 1999.

Tagging with Delayed Disambiguation

José M. Castaño and James Pustejovsky

Department of Computer Science
Brandeis University, Waltham, MA, 02453
{jcastano,jamesp}@cs.brandeis.edu

Abstract. We discuss problems inherent in domain specific tagging (biomedical domain) and their relevance to tagging issues in general. We present a novel approach to this problem which we call *tagging with delayed disambiguation* (TDD). This approach uses a modified, statistically-driven lexicon together with a small set of morphological, heuristic, and chunking rules which are implemented using finite state machinery. They make use of both delayed disambiguation and the concept of tag underspecification as an ordered sequence of tags.

Current tagging techniques are very well established and there seems to be little room to improve the standard accuracy of 96%-97%. The availability of off-the-shelf taggers trained on significantly large corpora and their ability to be re-trained on any tagset and corpus are the determining factors for their extensive use and popularity. However when tagging a new domain, the performance drops significantly, e.g., TnT trained on the WSJ, has an accuracy of 85.19% on the GENIA Corpus.

Our approach to tagging and chunking within the bio-medical domain was motivated by practical concerns, namely targeted information extraction, in our MEDSTRACT project. This domain has been recognized as being particularly impervious to robust entity extraction [1], and our initial impression was that none of the off-the-shelf taggers we tried was performing well on this corpus. We present a novel approach called *Tagging with Delayed Disambiguation* (TDD), which uses a quantitatively derived lexicon and hand-crafted chunking rules, using finite state machinery. It has been a popular approach to address chunking as a tagging problem (e.g., CoNLL-2000 Shared Task on Chunking). The approach we have taken for tagging disambiguation is just the opposite, where disambiguation is delayed and done while chunking (if it is done at all). The key concepts in this approach are: (a) treating regular ambiguity in the language as underspecification; (b) delaying the disambiguation process, and resolving the ambiguities as chunking is performed.

Chunking rules (finite state patterns) refer to prefixes in an ordered sequence of tags (e.g., *"VBN VBD"*; past tense vs. past participle) which are interpreted as underspecified tags. They are implemented using 195 left handle rules. We also use a small set of 28 heuristic disambiguation rules *à la Brill* and a simple morphological module (33 suffixes and 44 prefixes). The system reflects common linguistic knowledge of English.

The availability of GENIA [2] and the corpus of the UPenn Biomedical Information Extraction Project (*BioPenn*) [3] made it possible to train taggers in the Biological domain and allowed us to evaluate the performance of our approach against reference

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 29 – 30

corpora.The first set of tests evaluated the TDD approach using the modified Brill lexicon (DD1) and the TnT tagger trained in the WSJ (as it comes with the distribution, TnT1). A second set of tests evaluated our approach compared with TnT trained on the WSJ and a set of approximately 1,000 abstracts from the GENIA corpus (TnT-2). The evaluation was performed on the whole set of the GENIA corpus, so there were 1,000 abstracts that were not used for training or deriving the lexicon of TnT. In TDD-2, we replaced Brill's Lexicon by the lexicon obtained from the training of TnT-2.

Corpus	TDD-1	TnT-1	TDD-2	TnT-2
GENIA	95.18	85.19	97.92	97.62
WSJ	95.35	97.11	95.78	97.33
BioP	94.60	87.52	95.76	94.55

Table 1. Accuracy Results

TDD-2 obtains state of the art accuracy (97.92%) on the GENIA corpus. The performance is lower on the WSJ, but it is remarkably good (95.78%) considering that, during the development we did not take into account any properties of this corpus. Our approach is similar to the one presented by [4], in the sense that it uses hand-crafted rules and keeps multiple tags. It is different however, because we use a statistically derived lexicon, and employ several disambiguation rules, and the evaluation is made against publicly available corpora (GENIA and WSJ), with the tagsets used in those corpora.

The rule-based, manually encoded tagger adapts itself and is easier to modify for specific purposes and domains. The annotated corpus can be seen as the desired output. It appears that a TDD approach provides a more stable result, with no need of retraining, while being less sensitive to the sparse data effect, guideline criteria and the lexicon employed .

Acknowledgements

This research has been funded by NLM grant 5R01 LM0006649-5.

References

1. Hirschman, L., Park, J.C., J. Tsujii, L.W., Wu, C.H.: Accomplishments and challenges in literature data mining for biology. Bioinformatics Review **18** (2002)
2. Collier, N., Mima, H., Lee, S., Ohta, T., Tateisi, Y., Yakushiji, A., Tsujii, J.: The GENIA project: Knowledge acquisition from biology texts. Genome Informatics (2000) 448–449
3. S.Kulick, A.Bies, M.Liberman, M.Mandel, R.McDonald, M.Palmer, A.Schein, L.Ungar: Integrated annotation for biomedical information extraction. In: NAACL/HLT Workshop on Linking Biological Literature, Ontologies and Databases: Tools for Users. (2004) 61–68
4. Samuelsson, C., Voutilainen, A.: Comparing a linguistic and a stochastic tagger. In Cohen, P.R., Wahlster, W., eds.: Proceedings of the ACL'97, Somerset, New Jersey, Association for Computational Linguistics (1997) 246–253

A New Algorithm for Unsupervised Induction of Concatenative Morphology

Harald Hammarström

Chalmers University of Technology
412 96 Gothenburg
Sweden
{harald2@cs.chalmers.se}

1 Introduction

This paper sketches a new algorithm for unsupervised induction of concatenative morphology. The algorithm differs markedly from previous approaches in both segmentation and paradigm induction. It is illustrated here with the respect to suffixes, using the following notation:

- W: the set (not bag) of words in the corpus
- $s \triangleleft w$: s is a suffix of the word w i.e there exists a (possibly empty) string x such that $w = xs$
- $Stems(s) = \{x | xs \in W\}$: the set of all strings ("stems") that make a word in the corpus if appended with s
- $f(s) = |\{w \in W | s \triangleleft w\}|$: the number of words with suffix s (equals $|Stems(s)|$)
- $s_i(w)$: the suffix of w that begins at position $0 \leq i \leq |w|$
- $Q(w) = \{s_i(w) | i < |w|\}$: the set of (non-empty) suffixes of s
- $S = \bigcup_{w \in W} Q(w)$: all suffixes in the corpus

2 Segmentation

The segmentation takes a corpus as input and output a ranked list of (all) suffixes. The ranking is meant to say how salient a suffix is for the language of the corpus, and is computed in three steps:

1. **Relative Frequency Increases:** Define $Z : S \times W \rightarrow \mathbf{Q}^+ \cup \{\mathbf{0}\}$:

$$Z(s,w) = \begin{cases} 0 & \text{if not } s \triangleleft w \\ 1 & \text{if } s = s_0(w) \\ \frac{f(s_i)}{f(s_{i-1})} & \text{if } s = s_i(w) \text{ for some } 0 < i < |w| \end{cases} \quad (1)$$

   Note that f, and hence Z, depends on W.
2. **Accumulation:** Calculate $Z^W : S \rightarrow \mathbf{Q}^+$:

$$Z^W(s) = \sum_{w \in W} Z(s,w) \quad (2)$$

3. **Re-scale:** Scale on suffix-length by a parametre $p = 2$: $\overline{Z^W}(s) = |s|^p \cdot Z^W(s)$

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 31 – 32

3 Paradigm Induction

The paradigm induction phase outputs a ranked list of paradigms given a ranked list of segmented suffixes. By paradigm we simply mean a non-empty set of suffixes, and the ranking is meant to convey how salient a declension pattern is for the language in question. At first glance, the task of finding paradigms looks exceedingly difficult since the number of theoretically possible paradigms is exponential in the number of suffixes, and paradigms in real languages are often not mutually disjoint. Moreover, in a corpus of a real language we cannot expect to rely on there existing words that occur in all its forms.

1. **Testing Paradigm Heuristic** Suppose we have a hypothesis of a paradigm P. We give a test metric using the idea that suffixes of P ought to show up on the "same stems". First, for each suffix $x \in S$, define its quotient function $H_x(y) : S \rightarrow [0, 1]$ as:
$$H_x(y) = \frac{|\{z|\exists z(z \in Stems(x) \wedge zy \in W)\}|}{|Stems(x)|} \tag{3}$$
Construct a rank by summing the quotient functions of the members of P:
$$V_P(y) = \sum_{x \neq y \in P} H_x(y) \tag{4}$$
The $Rank_P(x) : S \rightarrow \mathbf{N}$ is then simply $|\{y|V_P(y) > V_P(x)\}|$.
Now, the test $VI(P)$ is a measure of how "high up" the sum of ranks of the members of P are, compared to the optimal sum (which depends on $|P|$ and is $0 + \ldots + |P| - 1$):
$$VI(P) = \frac{|P|(|P| - 1)}{2\sum_{x \in P} rank_P(x)} \tag{5}$$
2. **Gradient Search** It is intractable to list all hypotheses of paradigms P, thus we suggest a way to "grow" paradigms. Start with a one member paradigm and greedily improve the VI-score, by successively adding or taking away one suffix at a time (until the score doesn't improve by a one-member change):
$$G(P) = argmax_{p \in \{P\} \cup \{P \text{ xor } s | s \in S\}} VI(p) \tag{6}$$
$$G^*(P) = \begin{cases} P & \text{if } G(P) = P \\ G^*(G(P)) & \text{if } G(P) \neq P \end{cases} \tag{7}$$
Where P xor s means $P \setminus \{s\}$ if $s \in P$ and $P \cup \{s\}$ if $s \notin P$.

Naturally, the induced paradigms $A = \{G(\{s\})|s \in S\}$ are all those that can be grown from (at least one) suffix in the corpus, and finally we rank them by their VI-score and average "suffixness" of its members:
$$R(P) : A \rightarrow \mathbf{Q}^+ \cup \mathbf{0}$$
$$R(P) = \frac{VI(P)}{|P|} \sum_{s \in P} \overline{Z^W}(s) \tag{8}$$

Morphological Parsing of Tone: An Experiment with Two-Level Morphology on the Ha language

Lotta Harjula

Institute for Asian and African Studies, University of Helsinki, Finland

Morphological parsers are typically developed for languages without contrastive tonal systems. Ha, a Bantu language of Western Tanzania, proposes a challenge to these parses with both lexical and grammatical pitch-accent (Harjula 2004) that would, in order to describe the tonal phenomena, seem to require an approach with a separate level for the tones. However, since the Two-Level Morphology (Koskenniemi 1983) has proven successful with another Bantu language, Swahili (Hurskainen 2004), it is worth testing its possibilities with the tonally more challenging Bantu languages.

Lexical accents are naturally marked in the lexicon. The lexical accents of nouns and other word classes (except verbs) are lexically associated with certain vowels. Nominal lexical accents are only moved in restricted contexts which are easily defined in rules.

On the other hand, lexical accents of verbs are not underlyingly associated with any certain vowel but the association is determined by the complete verbal form. In addition to the lexical accents, verbal forms may have grammatical accents, realised either on the prefixes, or on the first or the second mora of the stem. The lexical accent or the absence of the accent, together with the grammatical accents, defines the grammatical forms of the verbs. Both lexical and grammatical accents are realised as high tones on certain tone-bearing units.

There are also some other grammatical tonal elements, called index forms that have floating accents, i.e. accents that are not underlyingly associated with any certain vowel. The accents of these forms are realised as a high tone either on the initial vowel or on the first vowel of the stem of the following word. When the possible lexical accent of the noun stem falls on the syllable following the accent of the index, the vowel of the augment is lengthened.

Thus, the morphological parser for Ha should be able to handle several different tonal phenomena: 1) the lexical accents, which are deleted or moved in some grammatical forms; 2) the grammatical accents and all their possible places of realisation; and 3) the floating accents of the index forms.

This experiment shows that it is indeed possible to morphologically parse a language with both lexical and grammatical accents with Two-Level Morphology rules. The basic idea is to mark the possible vowels on which the grammatical or moved lexical accents may fall in the lexicon, and write rules that allow the positions to be realised as surface accents when the appropriate tense marker or other segmental element is present. When there is no segmental but only tonal marking of a tense, the macrostem is lexically prefixed with a symbol that can be used as a context in the rules. Also, some of the grammatical accents may also be marked in the lexicon. For example, accents falling on verbal prefixes are lexically marked in a sublexicon which is only used with certain verbal forms.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 33 – 34

However, the parsing formalism presented here does not always describe the tonal phenomena that are found in the Ha language. With some type of accents the lexical accents can be mapped directly to the surface realisations, but with others the interaction of the accents causes changes on the segmental level, or the morphophonemic changes of the segmental level affect the realisations of the accents. Thus, for proper description of the language, a formalism which would allow the tones or accents to be mapped with the segmental level only after certain rules have applied in the two levels separately, is required.

References

1. Harjula, Lotta, 2004. The Ha Language of Tanzania: Grammar, Texts, and Vocabulary. Cologne: Ruediger Koeppe Verlag.
2. Hurskainen, Arvi, 2004. Swahili Language Manager: A Storehouse for Developing Multiple Computational Applications. Nordic Journal of African Studies, 13(3): 363-397. Also in: www.njas.helsinki.fi
3. Koskenniemi, Kimmo, 1983. Two-level morphology: A general computational model for word-form recognition and production. Publications No.11. Department of General Linguistics, University of Helsinki.

Describing Verbs in Disjoining Writing Systems

Arvi Hurskainen, Louis Louwrens, and George Poulos

Institute for Asian and African Studies, University of Helsinki, Finland
University of South Africa, Pretoria, South Africa

Many Bantu languages, especially in Southern Africa, have a writing system, where most verb morphemes preceding the verb stem and some suffixes are written as separate words. These languages have also other writing conventions, which differ from the way they are written in other related languages. These two systems are conventionally called disjoining and conjoining writing systems. Disjoining writing can be considered simply as an under-specified way of writing, but for computational description it is a challenge, especially if the system allows only continuous sequences of characters to be recognised as units of analysis. In order to reduce unnecessary ambiguity, verb morphemes should be isolated from such strings of characters that are real words.

There are at least three approaches for handling disjoining writing. (a) Each continuous string of characters is considered a 'word' and ambiguity is resolved after morphological description. (b) Disjoining writing is first converted to conjoining writing, in other words, it is 'normalised', and morphological description is carried out on the basis of this new writing form (Hurskainen and Halme 2001). (c) The verbs, together with disjoint prefixes and suffixes, are described directly as verb structures.

Here we are concerned with the third method.

The most efficient method of handling verbs in a disjoining writing system is to describe them directly without pre-processing. Below we shall discuss this method by using Northern Sotho language (Poulos and Louwrens 1994) as a test case. The aim is to construct a full scale implementation that includes all verb structures and all verb stems of the language.

In addition to disjoining writing, the description of the verb involves also such non-concatenative features as reduplication of the verb stem and the constraining of co-occurrence of such verb morphemes that are on different sides of the verb stem. All these phenomena are handled in the following implementation, which makes use of the finite state methods developed by Xerox (Beesley and Karttunen 2003). A very brief skeleton lexicon of the verb `bona` (to see) is described below.

```
Multichar_Symbols
  ^[ ^]
  @P.PAST.ilE@  @R.PAST.ilE@
  @P.SBJN.a@    @R.SBJN.a@
  @P.SBJN.E@    @R.SBJN.E@
  @P.HABIT.e@   @R.HABIT.e@
  @P.NORM.a@    @R.NORM.a@
LEXICON Root
  SubjPref;
LEXICON SubjPref
  ke=Sbjn+@P.NORM.a@:@P.NORM.a@ke%  FutPref;
```

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 35 – 37

```
  ke=Sbjn+@P.SBJN.E@:@P.SBJN.E@ke%  VStart;
  ke=Habit+@P.HABIT.e@:@P.HABIT.e@ke%  VStart;
  ke=Perf+@P.PAST.ilE@:@P.PAST.ilE@ke%  VStart;
LEXICON FutPref
  tla=Fut+:tla%  VStart;
LEXICON VStart
  0:^[{ VStem;
LEXICON VStem
  bon VFinV;
LEXICON VFinV
  +a@R.NORM.a@:@R.NORM.a@A VEnd;
  +a=Redup@R.NORM.a@:@R.NORM.a@A VEndRedup;
  +E@R.SBJN.E@:@R.SBJN.E@E VEnd;
  +E=Redup@R.SBJN.E@:@R.SBJN.E@E VEndRedup;
  +e@R.HABIT.e@:@R.HABIT.e@e VEnd;
  +e=Redup@R.HABIT.e@:@R.HABIT.e@e VEndRedup;
  +ilE@R.PAST.ilE@:@R.PAST.ilE@ilE VEnd;
  +ilE=Redup@R.PAST.ilE@:@R.PAST.ilE@ilE VEndRedup;
LEXICON VEnd
  0:}^1^] #;
LEXICON VEndRedup
  0:}^2^] #;
```

The full description of the verb in Northern Sotho contains a number of structures, where the verb-final vowel, or a suffix, constrains the co-occurrence of certain verb prefixes. In the above example we have four cases, where the marker of the correct word form is the verb final vowel or suffix. Because the marker is after the verb stem, it is not practical to construct the finite state lexicon separately for each case.

The Xerox tools offer a method for handling such cases. A set of flag diacritics can be used in the lexicon for controlling the co-occurrence of certain morphemes. In this implementation, we have used a pair of the P-type and R-type flag diacritics for controlling the morpheme sequences (Beesley and Karttunen 2003: 353-355).

Particularly important in the lower-side meta-language is the section of the string that is subject to reduplication. This section is delimited with special multi-character symbols `^[` and `^]`. We also see that the actual string to be defined as a regular expression is enclosed with curly brackets `{` and `}`. The multi-character symbol `^2` in the lower string triggers the reduplication of the preceding regular expression. The Xerox tool package contains a compile-replace algorithm, which makes it possible to include finite state operations other than concatenation into the morphotactic description (Beesley and Karttunen 2003: 379-380).

The full description of the Northern Sotho verb is much more complicated than what is described above. The morpheme slots, which mark agreement for each noun class, have a total of twenty alternative prefixes, including first and second person singular and plural. Morpheme slots of this type include the subject prefix, which can be repeated after tense-aspect marking in some forms, and the object prefix. Verb extensions also increase the number of possible forms.

Some amount of complexity is added also by the object prefix of the first person singular, which is a nasal. It causes several types of sound changes in the first phoneme of the verb stem, and such forms are written conjointly.

The normal compilation of the lexicon of this size and complexity into a transducer is no problem, although the verb structure produces more than 4 billion paths. The memory problem will be encountered when 'compile-replace lower' is applied to this initial network.[1]

It is possible to reduce the number of paths by merging identical morphemes in a morpheme slot into a single entry and return them to separate readings in the post-processing phase. By this method the maximum number of morphemes in a morpheme slot is reduced from twenty to eleven and the number of paths is reduced accordingly. Another, and more efficient, method for handling the memory problem is to cut the lexicon into parts, so that only the section requiring a regular expression notation, i.e. the verb stems, will be compiled with compile-replace lower, and then these partial lexicons are composed together as a single net. Because the verb reduplication concerns the verb stem only, the section of prefixes can be treated as a partial lexicon of its own. Using this method, it was possible to compile the full Northern Sotho verb lexicon with more than 4 billion paths. The total compilation time with Compac NX 7000 in Linux environment was less than two minutes.

References

1. Beesley, Kenneth and Karttunen, Lauri, 2003. Finite State Morphology. Series: CSLI Studies in Computational Linguistics. Stanford: Center for the Study of Language and Information.
2. Hurskainen, Arvi and Halme, Riikka, 2001. Mapping between Disjoining and Conjoining Writing Systems in Bantu Languages: Implementation on Kwanyama. Nordic Journal of African Studies, 10(3): 399-414.
3. Poulos, George and Louwrens, Louis J. 1994. A Linguistic Analysis of Northern Sotho. Pretoria: Via Afrika Ltd.

[1] For discussion on memory problems see Beesley and Karttunen 2003: 418-420

An FST grammar for verb chain transfer in a Spanish-Basque MT System

Iñaki Alegria, Arantza Díaz de Ilarraza, Gorka Labaka, Mikel Lersundi, Aingeru Mayor and Kepa Sarasola

IXA Group, University of the Basque Country

We are developing an Spanish-Basque MT system using the traditional transfer model and based on shallow and dependency parsing. The project is based on the previous work of our group but integrated in OpenTrad initiative [2]. This abstract sumarizes the current status of development of an FST grammar for the structural transfer of verb chains. This task is quite complex due to the high distance between both languages. In the actual implementation we are using XRCE Finite States Tools [1].

We will focus on the translation of Spanish non-finite forms (21%), indicative forms (65%) and periphrases (6%) covering 92% of all possible cases.

Spanish finite verbs are composed by several morphemes giving information about voice, mood, aspect, tense, person and number. Verb conjugations have simple or compound tenses. The periphrases are composed by a finite auxiliary verb, an optional particle, and the main verb (non-finite form, infinitive or gerund, giving the meaning).

Basque finite verbs can be synthetic, consisting of a single word, or analytical, consisting of a participial form and an auxiliary. Participles carry information about meaning, aspect and tense, whereas auxiliaries convey information about argument structure, tense and mood.

Depending on the Spanish form of the verb, its translation into Basque should be obtained in a different way. For the non-finite forms we translate with a verbal noun or a participle. Simple and complex tense verbs can be translated as synthetic or as analytical depending on the verb and its tense and, in some cases, we need a dummy auxiliary and its aspect. For the periphrases the schemma for Basque is very different: the main verb, the translation of the periphrastic form (or a modal particle or an adverb) and in some cases a dummy verb (each one with a different aspect). In the last position another auxiliary verb which depends on the transitive feature of the main verb or of the auxiliary verb.

The FST grammar for verb chains we present takes as input the morphological information of the nodes of the Spanish verb chain, the Basque form corresponding to the Spanish main verb of the chain, agreement information about the objects (absolutive and dative) and the type of subordination of the sentence. Its output is the list of the nodes of the corresponding Basque verb chain, each one with the information necessary to decide the order of the words, and to realize the morphological generation. The grammar contains three kinds of rules:

Identification and markup rules identify the type of the Spanish verb chain, and add a different schema for the Basque verb chain depending on the type: non-finite forms, non-periphrastic verbs and four periphrasis type verbs.

```
[ esVerbChainType @-> ... "=>" euVerbChainSchema ]
```

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 38 – 39

Attributes replacement rules replace attributes in the Basque schema with their corresponding values, depending on the values of some attributes in the Spanish verb chain and/or in the Basque schema.

```
[ "euAttr" @-> "euVal" || ?* esVals ?* "=>" ?* euVals ?* _ ]
```

Cleaning rules remove the unnecessary information.

This example illustrates the process: *"porque no habré tenido que comer patatas" (because I won't have to eat potatoes)*. The input for the sequence of transducers that will transfer the verb chain is the following:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
```

The first rule identifies the input of a Spanish verb chain that has a periphrastic of type 1, and adds the schema for the Basque verb for this type:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
=> P1> (main)Aspm/Per Aspp/Dum Aspd/Aux TenseM SubObjDat +RelM
```

The next rules replace one by one the attributes of the Basque verb schema. These are some of the replacements and contexts that constraint them:

```
Attribute Value       Context
Per       behar(per)   ?* 'tener' ?* 'que' ?* "=>" 'P1' ?*
Aspp      [partPerf]   ?* VAIF ?* "=>" 'P1' ?*
Aux       edun(aux)    ?* 'tener' ?* 'que' ?*
SubObjDat [s1s][o3p]   ?*'tr'?*'pl'?* '=>' ?*'edun(aux)'?*'1s'
```

The output after all these replacements is:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
=> P1> (main)[partPerf]/behar(per)[partPerf]/izan(dum)[partFut]
/edun(aux)[indPres][subj1s][obj3p]+lako[causal morpheme]
```

The last transducer eliminates the information of the input, and returns the desired output to the MT system. The information between parenthesis will be used to decide the order of the words in the syntactic generation phase and the information between brackets will be used in order to do the morphological generation. The translation obtained in the output of the system after the generation phase is the next sentence: *"ez ditudalako patatak jan behar izango"*

References

1. Beesley K. & L. Karttunen: Finite-State Morphology. CSLI Publications, Stanford, California (2003)
2. Corbí-Bellot M., M. L. Forcada, S. Ortiz-Rojas, J. A. Perez-Ortiz, G. Ramirez-Sanchez, F. Sanchez-Martinez, I. Alegria, A. Mayor, K. Sarasola: An Open-Source Shallow-Transfer MT Engine for the Romance Languages of Spain. EAMT (2005)

Finite state transducers based on k-TSS grammars for speech translation ⋆

A. Pérez[1], F. Casacuberta[2], I. Torres[1], and V. Guijarrubia[1]

[1] Universidad del País Vasco
webperaa@lg.ehu.es {manes, vgga}@we.lc.ehu.es
[2] Universidad Politécnica de Valencia
fcn@iti.upv.es

1 Introduction

Finite State Transducers (FST) can be automatically learnt from bilingual corpus using the GIATI [1] methodology. This technique combines both statistical alignments models and classical n-gram models. Alternatively, in this work we propose the use of a syntactic approach to n-gram models: the *k-testable in the strict sense* language models (k-TSS) [2]. Another motivation behind this work, is to study the speech translation from Spanish into Basque. Basque is an agglutinative pre-Indoeuropean language of unknown origin. Regarding to the word ordering, contrary to Spanish, Basque has left recursion.

2 Two architectures for speech translation

The goal of the statistical speech translation (summarized in eq. (1)) is to find the target language string ($\mathbf{t}$) with the highest probability, given the acoustic representation ($\mathbf{x}$) of a source language string ($\mathbf{s}$).

$$\widehat{\mathbf{t}} = \arg\max_{\mathbf{t}} P(\mathbf{t}|\mathbf{x}) = \arg\max_{\mathbf{t}} \sum_{s} P(\mathbf{t}, \mathbf{s}|\mathbf{x}) \tag{1}$$

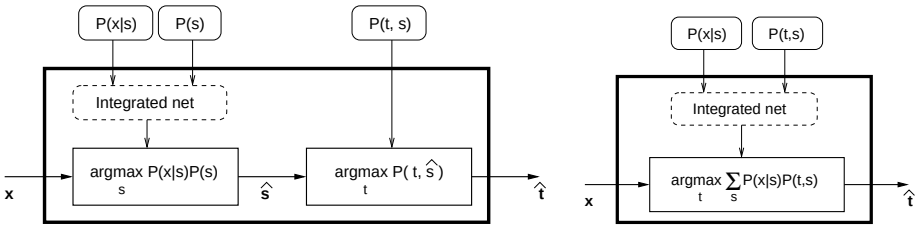
Two architectures [1] can be used in order to build the speech translation system (Fig. 1): the serial and the integrated one.

3 Experimental results and concluding remarks

Two series of experiments have been carried out on two synthetic bilingual corpora: EuTrans for Spanish to English translation [3], and Euskal Turista (ET) for Spanish to Basque translation. Spanish is the source language in both tasks. There are around 10.000 sentence pairs in the training set. The vocabulary size is around 650 word-forms in Spanish, 500 in English and 850 in Basque. In the text-test there are 3.000 pairs for Spanish to English translation, and 1.000 for Spanish into Basque. The speech corpora

⋆ This work has been partially supported by the Industry Department of the Basque Government and by the Universidad del País Vasco under grants INTEK CN02AD02 and 9/UPV 00224.310-15900/2004 respectively.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 40 – 42



(a) Serial architec. (b) Integrated architec.

Fig. 1. 1(a) A text translator after a speech decoder. 1(b) Integrated speech translator built on the basis of a text to text translator, expanding the source word on one edge by its phonetic transcription.

was recorded at 16 KHz in laboratory environment. It is composed of a training corpus of 1264 utterances by 16 speakers, and a test corpus of 336 utterances by 4 speakers [3].

The Spanish text-set perplexity is similar in both tasks (5.0), however, the speech perplexity is 6.9 for EuTrans and 13.5 for ET. Translation results are shown in table 1.

4 Concluding remarks

A speech translation system supported on the grammatical structure provided by the k-TSS models is presented. It is a finite state transducer, learned on the basis of the so called GIATI algorithm.The finite state methods allow for an easy integration of both acoustic and translation models. Experimental results over two different corpora representing the same application task (EuTrans and ET) have been reported. In spite of the reduced vocabulary, ET has proved to be a quite difficult task (compared to EuTrans) because of great inflection and word reordering of the Basque language. Further work is needed in order to improve Basque translation models, both changing alignment methodology and including linguistic information.

Task	speech-WER	text-TWER	serial-TWER	integrated-TWER
EuTrans	4.4	8.1	8.9	9.1
ET	8.6	39.7	58.3	54.7

Table 1. Experimental results in means of recognition word error rate (WER) and translation word error rate (TWER).

References

1. Casacuberta, F., Vidal, E.: Machine translation with inferred stochastic finite-state transducers. Computational Linguistics **30** (2004) 205–225
2. Torres, I., Varona, A.: k-tss language models in a speech recognition systems. Computer Speech and Language **15** (2001) 127–149
3. Amengual, J., Benedí, J., Casacuberta, F., Castaño, M., Castellanos, A., Jiménez, V., Llorens, D., Marzal, A., Pastor, M., Prat, F., Vidal, E., Vilar, J.: The EuTrans-I speech translation system. Machine Translation **1** (2000)

Unsupervised Morphology Induction Using Morfessor

Mathias Creutz, Krista Lagus, and Sami Virpioja

Neural Networks Research Centre, Helsinki University of Technology
P.O. Box 5400, FIN-02015 HUT, Finland
{mathias.creutz, krista.lagus, sami.virpioja}@hut.fi

Abstract

We present *Morfessor*, an unsupervised algorithm and software that induces a simple morphology of a natural language from a large corpus. Morfessor simultaneously builds a morph lexicon and represents the corpus with the induced lexicon using a probabilistic maximum a posteriori model.

Morfessor has been designed to cope with languages having predominantly a concatenative morphology and where the number of morphemes per word can vary much. This distinguishes Morfessor from resembling unsupervised models, which assume a much more restricted word structure.

Morph segmentations produced by Morfessor have been applied in language modeling for unlimited-vocabulary Finnish and Turkish speech recognition. In the experiments on Finnish, the word error rate was nearly halved compared to the traditional word-based approach. Furthermore, the use of Morfessor produced fewer recognition errors than the use of the manually designed Finnish two-level morphological analyzer, due to the better coverage of the word forms occurring in the data. Besides automatic speech recognition, further possible applications of Morfessor include information retrieval and machine translation.

The figure below shows some sample segmentations of Finnish and English word forms. An on-line demonstration and a software package implementing an earlier version of the method can be found at `http://www.cis.hut.fi/projects/morpho/`. Additionally, links to a number of related publications can be found on the web site.

aarre + **kammio** + *i* + *ssa*, **aarre** + **kammio** + *nsa*, **bahama** + **saar** + *et*,
edes + **autta** + *ma* + *ssa*, taka + **penkki** + *lä* + *in*+ *en*, **voi** + *mme* + *ko*
abandon + *ed*, **abandon** + *ing*, **abandon** + *ment*, **beauti** + *ful*,
beauty + *'s*, **long** + **fellow** + *'s*, **master** + **piece** + *s*, un + **expect** + *ed* + *ly*

Fig. 1. Examples of segmentations learned from Finnish and English data set. Suggested prefixes are underlined, stems are rendered in **boldface**, and suffixes are *slanted.*

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 43 – 43

SProUT – a General-Purpose NLP Framework Integrating Finite-State and Unification-based Grammar Formalisms

Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer

German Research Center for Artificial Intelligence (DFKI), Language Technology Lab
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
http://sprout.dfki.de

In these days, we are witnessing a growing trend of exploiting lightweight linguistic analysis for converting of the vast amount of raw textual data into structured knowledge. Although a considerable number of monolingual and task-oriented NLP systems have been presented, relatively few general-purpose architectures exist, e.g., GATE [1] or ELLOGON [2].

This presentation introduces SProUT – a novel general-purpose multilingual NLP platform [3][1]. The main motivation for its development comes from (i) the need of having one modular system for multilingual and domain-adaptive text processing, which is portable across different platforms and (ii) to find a balance between efficiency and expressiveness of the grammar formalism.

SProUT is equipped with a set of reusable online processing components for basic linguistic operations, ranging from tokenization, morphology, gazetteer etc. to text coreference resolution. They can be combined into a pipeline that produces several streams of linguistically annotated structures, which can serve as input for the grammar interpreter, applied at the next stage.

The grammar formalism in SProUT is a blend of efficient finite-state techniques and unification-based formalisms, guaranteeing expressiveness and transparency. To be more precise, a grammar in SProUT consists of pattern-action rules, where the LHS of a rule is a regular expression over typed feature structures (TFS) with functional operators and coreferences, representing the recognition pattern, and the RHS of a rule is a TFS specification of the output structure. Coreferences express structural identity, create dynamic value assignments, and serve as a means of information transport. Functional operators provide a gateway to the outside world and are utilized for introducing complex constraints in rules, for forming the output of a rule, and for integrating external processing components.

Grammars, consisting of such rules, are compiled into extended finite-state networks with rich label descriptions (TFSs). For their efficient processing, a handful of methods going beyond standard finite-state techniques have been introduced. Grammar rules can even be recursively embedded, which in fact provides grammarians with a context-free formalism. The following rule for recognizing prepositional phrases gives an idea of the syntax of the grammar formalism:

```
pp :>  morph & [POS  Prep, SURFACE #prep, INFL infl & [CASE #c]]
```

[1] This publication is supported by a research grant COLLATE II 01 IN C02 from the German Federal Ministry of Education and Research.

```
   (morph & [POS  Adjective, INFL infl & [CASE #c, NUMBER #n, GENDER #g]]) *
   (morph & [POS  Noun, SURFACE #noun1, INFL infl & [CASE #c, NUMBER #n,
                                                     GENDER #g]])
   (morph & [POS  Noun, SURFACE #noun2, INFL infl & [CASE #c, NUMBER #n,
                                                     GENDER #g]]) ?
-> phrase & [CAT  pp, PREP #prep, CORE_NP #core_np,
             AGR  agr & [CASE #c, NUMBER #n, GENDER #g]],
where #core_np = Append(#noun1, " ", #noun2).
```

The first TFS matches a preposition. It is followed by zero or more adjectives. Finally, one or two noun items are consumed. The variables `#c`, `#n`, `#g` establish coreferences, expressing the agreement in case, number, and gender for all matched items (except for the initial preposition item which solely agrees in case with the other items). The RHS of the rule triggers the creation of a TFS of type phrase, where the surface form of the matched preposition is transported into the corresponding slot via the variable `#prep`. The value for the attribute `core_np` is created through a concatenation of the matched nouns (variables `#noun1` and `#noun2`). This is realized via a call to the functional operator Append.

SProUT comes with an integrated graphical development and testing environment. The grammars can be either created in text or XML editing mode, and can be visualized in a graphical mode. The grammar GUI resembles state-of-the-art development environments for programming languages, e.g., errors and warnings listed in the error message window are linked to the corresponding piece of grammar in the editor. Several user interfaces for inspecting the output of the linguistic processing components and for testing the grammars are provided.

SProUT grammars can be cascaded in order to structure and combine different recognition strata. A declarative description of an architecture instance can be compiled to and encapsulated in a Java class and for example plugged into the Heart of Gold NLP middleware [4].

Currently SProUT has been adapted to processing 11 languages, including major Germanic, Romance, Slavonic, and Asian languages. It has been deployed as the core IE component in several industrial and research projects [3]. In our presentation we showcase the development of the SProUT named entity grammars.

References

1. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to Meet New Challenges in Language Engineering. In Natural Language Engineering, **12**, No. 3/4, (2004) 349–373.
2. Petasis, G., Karkaletsis, V., Paliouras, G., Androutsopoulos, I. Spyropoulos, C.: Ellogon: A New Text Engineering Platform. In Proceedings of LREC 2002, Canary Island, (2002) 72–78.
3. Drożdżyński, W., Krieger, H.-U., Piskorski J., Schäfer, U.: Shallow Processing with Unification and Typed Feature Structures – Foundations and Applications. In Künstliche Intelligenz, **1/04**, (2004) 17–23.
4. `http://heartofgold.dfki.de`

Tool Demonstration: Functional Morphology

Markus Forsberg and Aarne Ranta

Department of Computing Science
Chalmers University of Technology and the University of Gothenburg
SE-412 96 Gothenburg, Sweden
{markus, aarne}@cs.chalmers.se

1 System Description

We will present *Functional Morphology*[1] [5], abbreviated *FM*, which is a tool that implements a methodology for constructing natural language morphologies in the functional language Haskell [8]. FM has its own runtime system that supports morphological analysis and synthesis. Moreover, a morphology implemented in FM can be compiled to many other source formats.

FM adopts a *word-and-paradigm* view of morphology: it represents a morphology as a set of inflection tables, *paradigms*, and the lexicon as a set of dictionary words each tagged with a pointer to an inflection table.

The basic idea behind FM is simple, instead of working with untyped regular expressions, which is the state of the art of morphology in computational linguistics, we use finite functions over hereditarily finite algebraic data types. These data types and functions constitute the language-dependent part of the morphology. The language-independent part consists of an untyped dictionary format which is used for synthesis of word forms, translations to other formats, and a decorated trie, which is used for analysis.

Functional Morphology builds on ideas introduced by Huet [6] in his computational linguistics toolkit Zen, which he has used to implement the morphology of Sanskrit. In particular, Huet's ideas about sandhi in Sanskrit have been adopted to a language independent description of compound analysis in FM.

The goal of FM has been to make it easy for linguists, who are not trained as functional programmers, to implement the morphology of a new language. In addition to the ease of programming, FM attempts to exploit the high level of abstraction provided by functional programming to make it possible to capture linguistic generalizations.

A morphology written in FM has a type system, which defines the inflectional and inherent parameters of the language described. By using algebraic data types, the type system can guarantee that no spurious parameter combinations appear in the morphology description, at the same time as all meaningful combinations are defined.

The use of the functional language Haskell provides, besides typing, many other language features that simplify the development of a morphology: *higher-order functions*, functions as first class objects, give the possibility of defining a paradigm in terms of another paradigm; the *class system* is used for sharing code between morphologies of different languages.

[1] FM homepage: `http://www.cs.chalmers.se/~markus/FM/`

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 46 – 47

The lifetime of a digital linguistic resource such as morphology depends on which system it has been developed in [3]. If the resource has been developed in a proprietary system with a binary-only format, and the system is no longer supported after some years, it may be impossible to access the resource. FM offers a solution to this problem by supporting translation to a multiple of different formats, such as XFST source code [2], GF [10] source code, SQL source code, full form lexicon, full form tables etc. This feature will hopefully prolong the lifetime of a morphology developed in FM. Furthermore, the system is completely open source, which should improve the situation even more.

2 Results

The following morphologies have been implemented in Functional Morphology: a Swedish inflection machinery and a lexicon of 20,000 words; a Spanish inflection machinery + lexicon of 10,000 words [1]; major parts of the inflection machinery + lexicon for Russian [4], Italian, Estonian [7], and Latin. Comprehensive inflection engines for Finnish, French, German, and Norwegian have been written following the same method but using GF as source language [9]. Since FM can generate GF source code, there exists a seamless connection between GF grammars and morphologies defined in FM.

References

1. I. Andersson and T. Söderberg. Spanish Morphology – implemented in a functional programming language. Master's Thesis in Computational Linguistics, 2003. `http://www.cling.gu.se/theses/finished.html`.
2. K. R. Beesley and L. Karttunen. *Finite State Morphology*. CSLI Publications, Stanford University, United States,, 2003.
3. S. Bird and G. Simons. Seven dimensions of portability for language documentation and description. *Language*, 79:557–582, 2003.
4. L. Bogavac. Functional Morphology for Russian. Master's Thesis in Computing Science, 2004.
5. M. Forsberg and A. Ranta. Functional Morphology. *Proceedings of the Ninth ACM SIGPLAN International Conference of Functional Programming, Snowbird, Utah*, pages 213–223, 2004.
6. G. Huet. The Zen Computational Linguistics Toolkit. `http://pauillac.inria.fr/~huet/`, 2002.
7. M. Pellauer. A Functional Morphology for Estonian. Term Paper, 2005.
8. S. Peyton Jones and J. Hughes. Report on the Programming Language Haskell 98, a Non-strict, Purely Functional Language. Available from `http://www.haskell.org`, February 1999.
9. A. Ranta. Grammatical Framework Homepage, 2000–2004. `http://www.cs.chalmers.se/~aarne/GF/`.
10. A. Ranta. Grammatical Framework: A Type-theoretical Grammar Formalism. *The Journal of Functional Programming*, 14(2):145–189, 2004.

From Xerox to Aspell: A First Prototype of a North Sámi Speller Based on TWOL Technology

Børre Gaup[1], Sjur Moshagen[1], Thomas Omma[1], Maaren Palismaa[1], Tomi Pieski[1], and Trond Trosterud[2]

[1] The Saami Parliament, Norway
[2] Faculty of the Humanities, University of Tromsø

Keywords: Sámi, transducers, language technology, spelling, proofing, minority languages

Our demo presents work from a joint project with a twofold goal: To build a parser and disambiguator for North and Lule Sámi, and to make a practical spellchecker for the same languages. The core analyser is written with the Xerox tools `twolc`, `lexc` and `fst`, and the disambiguator uses costraint grammar (`vislcg`).

The spellchecker is intended to work on 3 platforms, for a wide range of programs. One of the speller engines we will have to cover is thus the `spell` family of spellcheckers (here represented by `Aspell`). This implies making a finite state automaton, rather than a transducer.

In the demonstration, we will show the present (alpha) version of the speller. We use the Xerox tools to generate a fullform list of the whole lexicon (we exclude all circular entries), which initially created a whopping 580 Mb text file, corresponding to 150 million word forms. Aspell took that whole wordlist and could use it as it was, but it was hardly practical (it worked, though!). After some modifications our present transducer creates «only» 290 Mb of data. This list is then reduced to a set of inflection stems with the Aspell `munch-list` option. It works by passing the `munch-list` option a file of available inflection lexicons, and Aspell will then *munch* through the fullform wordlist and reduce all wordforms that fits an inflectional lexicon to one stem plus the identified inflection lexicon. Finally we compress the lexicon into an Aspell specific binary format. Its size is at the moment about 65 Mb.

This way of creating a finite state automaton is quite different from how the transducer itself works. Just like Finnish, Sámi has consonant gradation, but unlike in Finnish, the Sámi consonant gradation affects almost all consonant groups of the stressed syllable, in most stem classes (some stem classes are never altered). Moreover, in several word forms, the diphthong of the stressed syllable is altered as well. This gives us as much as 4 surface stems for one and the same lexeme. For the two-level transducer, this is not a problem, since these morphophonological processes are handled by our two-level rules, but it becomes a complicating factor when reverting to the single-level model of Aspell.

The Aspell munch-list way of creating stems and inflectional suffixes isn't very satisfying, for at least two reasons: we already have an excellent morphological description of North Sámi, and duplicating it in the form of the Aspell inflectional lexicon isn't very elegant and requires redoing the same work; and by having two parallel morphological descriptions the whole system requires more maintenance work and is more error-prone. But for the reasons cited above regarding Sámi morphophonology, we have

at present not found an easy way to generate the correct inflectional stems directly from the Xerox tools.

Aspell is not the optimal speller architecture, but since we need to have an alpha version running in order to test our lexicon (and since Aspell is one of the target spellers of the projecct), it turned out to be a working solution.

A Programming Language For Finite State Transducers

Helmut Schmid

Institute for Natural Language Processing (IMS)
University of Stuttgart, Germany
schmid@ims.uni-stuttgart.de

SFST-PL is a programming language for finite-state transducers which is based on extended regular expressions with variables. SFST-PL is used by the Stuttgart Finite-State-Transducer (SFST) tools which are available under the GNU public license. SFST-PL was designed as a general programming language for the development of tokenizers, pattern recognizers, computational morphologies and other FST applications. The first SFST application was the SMOR morphology [1], a large-scale German morphology which covers composition, derivation and inflection.

An SFST program is basically a regular expression which is optionally preceded by a list of variable and alphabet definitions. The following basic transducer expressions are available:

a:b	defines a transducer which maps the symbol **a** to **b** (in generation mode).
a	abbreviation of **a:a**
a:.	maps the symbol **a** to any symbol that it occurs with it in the alphabet. The alphabet contains a set of symbol pairs and must have been defined before.
.	abbreviation of **.:.**, the union of all symbol-pairs in the alphabet.
[abc]:[de]	identical to **a:d** | **b:e** | **c:e** ("|" is the union operator.)
[a-c]:[A-C]	same as **[abc]:[ABC]**.
{abc}:{de}	identical to **a:d b:e c:<>** This expression maps the string **abc** to **de**.
var	the transducer stored in variable *var*.
"lex"	a transducer consisting of the union of the lines in the file *lex* (Apart from ":" and previously seen multi-character symbols, all symbols in the argument file are interpreted literally.)
"<file>"	is a pre-compiled transducer which is read from *file*

SFST-PL supports multi-character symbols (which are enclosed in angle brackets like <Sg>) and a wide range of operators including concatenation, union '|', intersection '&', composition '||', complement '!', optionality '?', Kleene star '*' and Kleene plus '+', range '^', domain '_', inversion '^_', and replacement operators (<=,=>,<=>). The special symbol <> represents the empty string.

Variables are surrounded by dollar signs. They are defined with a command var = <expression> (where <expression> is some transducer expression). The alphabet is defined with the command ALPHABET = <expression>. The definition of an alphabet is required for the interpretation of the wild-card symbol '.' and for the complement and replacement operators.

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 50 – 51

Comments start with a percent sign and extend up to the end of the line. Whitespace is ignored unless it is quoted by a backslash. Programs can be partitioned into several files which are combined with include commands (like `#include "file"`) which insert the contents of the argument file at the current position. The pre-compilation of component transducers often speeds up the compilation.

The SFST tools comprise a compiler which translates SFST programs into minimized finite-state transducers. The compiler was implemented using a high-level C++ library and the YACC compiler generator, which makes it easy to change or extend the syntax of the programming language. The compiler generates three different transducer formats which are optimized for flexibility, speed or memory and startup efficiency, respectively. The SFST tools also comprise programs for analysis, printing, and comparison.

The following simple SFST-PL program will correctly inflect adjectives like “easy” (easier, easiest) and late (later, latest).

```
% the set of valid character pairs
ALPHABET = [A-Za-z]:[A-Za-z] y:i [#e]:<>

% Read a list of adjectives from a lexicon file
$WORDS$ = "adj"

% rule replacing y with i if followed by # and e
$Rule1$ = y <=> i (#:<> e)

% rule eliminating e if followed by # and e
$Rule2$ = e <=> <> (#:<> e)

$Rules$ = $Rule1$ & $Rule2$

% add inflection to the words
$S$ = $WORDS$ <ADJ>:# ({<pos>}:{} | {<comp>}:{er} |
                       {<sup>}:{est})

% apply the phonological rules to obtain the resulting
% transducer
$S$ || $Rules$
```

A more comprehensive morphology including mechanisms for dealing with derivation, compounding and inflection is available with the SFST tools. It is adaptable to other languages by changing the lexicon, the inflectional classes, and the phonological rules.

References

1. Schmid, H., Fitschen, A., Heid, U.: SMOR: A German computational morphology covering derivation, composition and inflection. In: Proceedings of the 4th International Conference on Language Resources and Evaluation. Volume 4., Lisbon, Portugal (2004) 1263–1266

FIRE Station

Bruce Watson

Technische Universiteit Eindhoven,
Department of Mathematics and Computer Science,
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
bruce@bruce-watson.com

This demonstration will provide a quick introduction to the FIRE Station [1]. FIRE Station is a "workstation" environment for manipulating FInite automata/transducers and Regular Expressions. It is built on top of the FIRE Works, a computational toolkit (with a programming interface only) for constructing, optimizing, manipulating, and using all sorts of regular language and regular relation objects. Both software systems are in a rather early stage, but the key insights are already apparent.

A key advantage over many other similar toolkits and environments is the close connection between the representation of an automaton (as a transition graph) and the representation of each state's accepted language (as a regular expression); indeed, these two concepts are simultaneously represented in a single abstract data-structure. This allows a unified view of regular languages, easing the way in which users interact with them. Perhaps more importantly, it can (in future versions) be used to allow for reversibility: from automaton back to regular expression/relation, and vice-versa. There are also significant performance advantages (in terms of memory and running time), and advantages in debugging/simulating automata. Finally, both systems are freely available, and we invite other implementors to work with us in creating new "skins" for various domains, such as computational linguistics, security systems, etc.

References

1. Frishert, M., Cleophas, L., Watson, B.W.: FIRE Station: an environment for manipulating finite automata and regular expression views. In Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S., eds.: CIAA 2004. Volume 3317 of LNCS., Berlin and Heidelberg, Springer-Verlag (2005) 125–133

FSMNLP 2005 A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.)
Abstracts of the Workshop, pp. 52 – 52

Author Index

Alegria, Iñaki 38
Amsalu, Saba 27

Bíró, Tamás 4
Barrachina, Sergio 5
Barthélemy, François 3

Carlson, Lauri 18
Casacuberta, Francisco 5, 40
Castaño, José M. 29
Champarnaud,, Jean-Marc 13
Civera, Jorge 5
Cohen-Sygal, Yael 6
Creutz, Mathias 43
Cubel, Elsa 5

de Ilarraza, Arantza Díaz 38
Dinçer Erbaş, Mehmet 19
Drożdżyński, Witold 44

Erdoğmuş, Müge 19

Forsberg, Markus 46

Gaup, Børre 48
Geyken, Alexander 7
Gibbon, Dafydd 27
Gomez, Fernando 25
Guijarrubia, Víctor 40
Guingne, Franck 13

Hammarström, Harald 31
Han, Na-Rae 8
Hanneforth, Thomas 7, 9
Harju, Tero 1
Harjula, Lotta 33
Hasida, Kôiti 15
Hulden, Mans 10
Hurskainen, Arvi 35

Jansche, Martin 11
Johnson, J Howard 12

Karttunen, Lauri 2
Kempe, André 13
Krieger, Hans-Ulrich 44

Labaka, Gorka 38
Lagarda, Antonio L. 5
Lagus, Krista 43
Lakshmanan, Kuppusamy 14
Lersundi, Mikel 38
Louwrens, Louis 35

Mayor, Aingeru 38
Miyata, Takashi 15
Moshagen, Sjur 48

Nasr, Alexis 16, 17
Nicart, Florent 13
Niemi, Jyrki 18, 26

Oflazer, Kemal 19
Omma, Thomas 48

Pérez, Alicia 40
Padró, Lluís 20
Padró, Muntsa 20
Palismaa, Maaren 48
Petersen, Ulrik 21
Pieski, Tomi 48
Piskorski, Jakub 22, 44
Poulos, George 35
Pustejovsky, James 29

Rambow, Owen 16
Ranta, Aarne 46

Sarasola, Kepa 38
Schiller, Anne 23
Schmid, Helmut 50
Schäfer, Ulrich 44

Torres, M. Inés 40
Trosterud,Trond 48

Uí Dhonnchadha, Elaine 24

van Delden, Sebastian 25
van Genabith, Josef 24
Vidal, Enrique 5
Vilar, Juan M. 5
Virpioja, Sami 43
Volanschi, Alexandra 17

Watson, Bruce 52
Wintner, Shuly 6

Yli-Jyrä, Anssi 26