Proceedings of ELS 2011

4th European Lisp Symposium

Special Focus on Parallelism and Efficiency

March 31 – April 1 2011
TUHH, Hamburg, Germany

Preface

Message from the Programme Chair

Welcome to ELS 2011, the 4th European Lisp Symposium.

In the recent years, all major academic events have suffered from a decreasing level of attendance and contribution, Lisp being no exception to the rule. Organizing ELS 2011 in this context was hence a challenge of its own, and I'm particularly happy that we have succeeded again.

For the first time this year, we had a "special focus" on parallelism and efficiency, all very important matters for our community with the advent of multi-core programming. It appears that this calling has been largely heard, as half of the submissions were along these lines. Another notable aspect of this year's occurrence is the fact that four dialects of Lisp are represented: Common Lisp, Scheme, Racket and Clojure. This indicates that ELS is successful in attempting to gather all crowds around Lisp "the idea" rather than around Lisp "one particular language". The European Lisp Symposium is also more European than ever, and in fact, more international than ever, with people coming not only from western Europe and the U.S.A., but also from such countries as Croatia and Bulgaria.

While attending the symposium is just seeing the tip of the iceberg, a lot have happened underwater. First of all, ELS 2011 would not have been possible without the submissions we got from the authors and the careful reviews provided by the programme committee members; I wish to thank them for that. I am also indebted to the keynote speakers who have agreed to come and spread the good word. I wish to express my utmost gratitude to our sponsors who contributed to making the event quite affordable this year again. Ralf Möller was our local chair, the "Grey Eminence" of the symposium, and we owe him a lot. Finally, my thanks go to Edgar Gonçalves for taking care of the website with such reactivity and attentiveness.

I wish you all a great symposium!

Didier Verna

Message from the Local Chair

Welcome to Hamburg University of Technology (TUHH). We hope you will enjoy your stay at our university for the 2011 European Lisp Symposium. Not only interesting presentations will be part of the programme, but also social events such as the the social dinner at the Feuerschiff (`http://www.das-feuerschiff.de`) and the Welcome Reception at Freiheit (`http://www.freiheit.com`). We would like to thank all sponsors for making the event possible. For those of you staying over the weekend, a tour to Miniatur-Wunderland (`http://www.miniatur-wunderland.de`) will be offered.

Yours sincerely,

Ralf Möller

TUHH (`http://www.tuhh.de`) is a competitive entrepreneurial university focussing on high-level performance and high quality standards. TUHH is dedicated to the principles of Humboldt (unity of research and education). TUHH has a strong international orientation and also focusses on its local environment. It contributes to the development of the technological and scientific competence of society, aiming at excellency at the national and international level in its strategic research fields, and educating young scientists and engineers within demanding programmes using advanced teaching methods.

Let's not forget Hamburg, for why are we all here? People say Hamburg is Germany's most attractive city combining urbane sophistication, maritime flair, the open-mindedness of a metropolis, and high recreational value. The second-largest city in Germany, it has traditionally been seen as a gateway to the world. Its port is not only the largest seaport in Germany and the second-largest in Europe, but a residential neighborhood with leisure, recreational and educational facilities. Hamburg is a very special place.

Organization

Programme Chair

- Didier Verna, EPITA Research and Development Laboratory, France

Local Chair

- Ralf Möller - Hamburg University of Technology, Germany

Programme Committee

- António Leitão, Instituto Superior Técnico/INESC-ID, Portugal
- Christophe Rhodes, Goldsmiths College, University of London, UK
- David Edgar Liebke, Relevance Inc., USA
- Didier Verna, EPITA Research and Development Laboratory, France
- Henry Lieberman, MIT Media Laboratory, USA
- Jay McCarthy, Brigham Young University, USA
- José Luis Ruiz Reina, Universidad de Sevilla, Spain
- Marco Antoniotti, Universita Milano Bicocca, Italy
- Michael Sperber, DeinProgramm, Germany
- Pascal Costanza, Vrije Universiteit of Brussel, Belgium
- Scott McKay, ITA Software, USA

Sponsors

EPITA
14-16 rue Voltaire
FR-94276 Le Kremlin-Bicêtre CEDEX
France
`www.epita.fr`

LispWorks Ltd.
St John's Innovation Centre
Cowley Road
Cambridge
CB4 0WS
England
`www.lispworks.com`

Franz Inc.
2201 Broadway, Suite 715
Oakland, CA 94612
`www.franz.com`

NovaSparks
86 Sherman Street
Cambridge, MA 02140
USA
`www.hpcplatform.com`

Freiheit Technologies gmbh
Straßenbahnring 22
20251 Hamburg
Germany
`www.freiheit.com`

TUHH
Schwarzenbergstraße 95
D-21073 Hamburg
Germany
`http://www.tu-harburg.de`

Contents

The Scheme Natural Language Toolkit (SNLTK)

NLP libraries for R6RS and Racket

D. Ćavar, T. Gulan,
D. Kero, F. Pehar,
P. Valerjev
University of Zadar

ABSTRACT

The Scheme Natural Language Toolkit (SNLTK) is a collection of procedures, example scripts and programs for natural language processing (NLP). The SNLTK library is fully documented and includes implementations of common data-structures and algorithms for text processing, text mining, and linguistic, as well as statistic analysis of linguistic data. Additionally, it also provides basic language data, word lists and corpus samples from various languages.

The SNLTK target group is Scheme and Lisp enthusiasts, computational linguists, linguists and language technology software developers. It is aiming at researchers and teachers, as well as students, who are interested in language related cognitive science, psychology, and linguistics.

Categories and Subject Descriptors
I.2.7 [**Computing Methodologies**]: Natural Language Processing; D.2.8 [**Software Engineering**]: Design Tools and Techniques—*Software libraries*

General Terms
Theory

Keywords
Scheme, Racket, NLP, SNLTK

1. INTRODUCTION

The SNLTK project started as a joint activity of faculty, assistants, and students from various departments at the University of Zadar, i.e. the Schemers in Zadar:

`ling.unizd.hr/~schemers`

The Schemers in Zadar is an open group of Scheme and Lisp enthusiasts. The goals of the group include, among others, the development of practical tools for computational linguistics, language related informatics and cognitive science in Scheme and Lisp. The resulting material should also serve as educational material for courses in the domain of Natural Language Processing, statistical language analysis and machine learning models.

The SNLTK is an outcome of joint NLP coding activities, and an attempt to aggregate the developed code and examples in an openly available general and specific text and language processing library.

`www.snltk.org`

The SNLTK is a collection of Scheme modules for various tasks in natural language processing (NLP), text mining, language related machine learning and statistical analysis of linguistic data.

The core libraries are written in R6RS Scheme, as well as in Racket (racket-lang.org). The code is tested for compatibility with common interpreters and compilers, e.g. Larceny.

The libraries are kept independent of external extensions and modules as much as possible, using the SRFI libraries where necessary. Additional programs, libraries and scripts are made available based on Racket. Racket is the recommended working and learning environment.

2. EXISTING TOOLKITS

Numerous other natural language processing tools, libraries and resources are implemented and available in various programming languages, e.g. Java, Perl, Python. Given the vast amount of NLP components and toolkits, we cannot discuss all the existing tools and libraries here. We will focus on the three most prominent packages and toolkits available for Java, Perl, and Python that are related to SNLTK and some of its goals.

In general, we should differentiate between speech and language processing. These two domains differ with respect to their nature and formal properties. While speech is concerned with the spoken signal, a non-discrete continuous event or phenomenon along the time axis, with specific issues related to its digitization, feature recognition and extraction, and consequently specific technologies, approaches, and algorithms, language refers to the indirectly observable properties of natural language that are related to combinatory and order relations and restrictions of sound groups, syllables, morphemes, words, and sentences. It is the language domain that the SNLTK is concerned with, and textual representations of natural language, rather than speech and signal processing.

Among the most popular of NLP toolkits is the Python Natural Language Toolkit (NLTK) [1]. The Python NLTK is a large collection of many common and popular algorithms for various text and natural language processing tasks. It contains algorithms for statistical NLP, as well as for rule-based analysis using common grammar types and symbolic approaches. Among the available libraries and tools it is the one widely used in educational computational linguistics programs worldwide. It is well documented, and significant amounts of teaching material and examples for it are freely available online. It contains implementations of the most important algorithms used in computational linguistics, as well as samples of common and valuable language data and corpora.

For Perl, a rich collection of tools and algorithms can be found in the CPAN archive (see www.cpan.org). Numerous sub-modules are available under Lingua::* module, including parsers for Link Grammars [9], WordNet ([14], [6]) access, and many other useful text mining and language processing tasks. Numerous valuable introductions to Perl for text processing and computational linguistic tasks are freely available online, or published as books, see e.g. [10].

Various tools and algorithms implemented in Java can be found online. Among the most prominent might be the OpenNLP (http://incubator.apache.org/opennlp/) collection of NLP tools, and the Stanford NLP software (http://nlp.stanford.edu/software/).

The coverage of the Python NLTK is most impressive, as well as the quality of the implementation. The mentioned Perl modules do as well offer impressive functionalities for various NLP oriented tasks and problems. Nevertheless, these implementations lack overall transparency at the implementation level, and provide less documentation and instructions related to the efficient and ideal implementation of particular algorithms and tools in the particular languages. It appears that they seem to have been designed with a clear usage orientation, rather than focusing on the educational goals of a deeper understanding of the particular algorithms and their implementation in the respective language or programming language paradigm.

Besides all the different implementations of computational linguistic algorithms for text processing, language and speech analysis, there are also numerous frameworks for the integration of specific language processing components for text mining. Among the most popular environments are Gate (cf. [21], [11]) and UIMA . These environments do not focus on particular language processing tasks, but provide an architecture for handling of sub-components in a language processing chain.

3. SNLTK GOALS

As mentioned in the previous section, for various computational linguistic tasks and problems, related to language, or even speech processing, many tools, libraries and components can be found online. The SNLTK is not intended to compete with these tools for applied computational linguistics tasks. It does not even intend to provide better, faster, or new solutions for common problems, or new types of implementation strategies for known or new algorithms. Its main goal is in fact an educational, experimental, and research oriented one. In addition, it provides alternative functional implementations of common, and potentially also new NLP algorithms.

One the one hand, many algorithms that we implemented for research and experimental purposes in the past, have been generalized and added to the library, some have been simplified in order to be easier understandable and analyzable. Many more will be prepared and added in the near future. Thus, an initial set of algorithms and tools in SNLTK is based on implementations from experiments and research projects that had a potential of being usable elsewhere.

On the other hand, we have chosen Scheme as our development and educational language (in addition to, and also replacing Python) for various reasons. Scheme is a very simple, but powerful language. It is easy and fast to learn, and simple to use. Further, various tools, in particular the intuitive IDE DrRacket (former DrScheme) for learning Scheme is available as a cross-platform environment, free of charge, and without runtime restrictions. It contains various learning packages, and freely available books and tutorials are distributed with it, and also available elsewhere online. DrRacket appeared to be the ideal environment for educational purposes. It is used in many programs in computer science, and large amounts of teaching material and examples are available online. However, it has not been widely used for computation linguistic courses, which made it necessary to collect and also re-implement algorithms for educational purposes.

In addition to being a very good educational and research language, with very good development tools like DrRacket, there are many useful tools for Scheme that allow it to be used for professional software development as well. In particular, compilers and interpreters exist that generate binaries, or code translation into other languages (e.g. C), or very good connectivity between languages like Java and C♯. Among the most interesting implementations of such interpreters and compilers we should mention Gambit-C (e.g. [5]), Larceny (e.g. [2]), Bigloo ([20], 1995; [19]), and Chicken Scheme (see www.call-cc.org). The possibility to generate standalone binaries, or translate code automatically into other programming languages is rather limited or non-existent in some of the other languages (e.g. Python and Perl) that natural language toolkits exist for.

Various books and educational material for computational linguistics based on Common Lisp are already available. One of the seminal publications on Lisp and Computational Linguistics is [8]. In addition, [18] offer many computational linguistics related Lisp implementations of algorithms. While we consider these textbooks extremely valuable and important, they nevertheless lack a discussion of modern approaches and implementation strategies in the domain of computational linguistics.

While we focus on the implementation of Scheme libraries, future releases or parallel versions might be geared towards ANSI Common Lisp. The use of CUSP and Eclipse as a development environment is a possible path, since affordable commercial Lisp development environments tend to be outside of the range for the academic and research community.

4. LIBRARY CONTENT

In its current state the SNLTK contains basic procedures and data from domains like:
– Finite State Automata
– Parsing with Context Free Grammars
– N-gram models
– Vector Space Models and algorithms for classification and clustering
– Basic language data for various languages
– Additional components

Specific statistical procedures for the processing of N-gram models are being developed, as well as document classification and clustering functionality.

In the following we shall describe some of the subcomponents and functionalities implemented in the SNLTK.

4.1 Finite State Automata

Finite State technologies for spell checkers, morphological analyzers, part of speech taggers, shallow parsers, and various other approaches to NLP are well known and discussed in the literature, see for example [15], and [16]. Finite State Automata are used for spell checkers, and shallow parsers, wherever regular grammars or languages are sufficient for natural language processing.

We made various implementations of FSAs for lexicon compression, morphological analyzes (Croatian morphological analyzer), and simple word class recognition available in the SNLTK. The current implementation makes use of table based FSA implementations. Basic functionalities include the generation of acyclic deterministic FSAs (ADFSA) from finite word lists, as well as common operations like unions, minimization, and concatenations over ADFSAs. Missing functionalities include conversions of non-deterministic automata to deterministic ones, the construction of transducers or Moore/Mealy machines ([17], [12]), and various other optimizations and code export.

Currently ADFSAs can be exported as DOT definitions for visualization and interoperability (e.g. code generation, transformation using Graphviz and related tools). A future version should be able to export C code definitions of automata, maybe even directly assembler.

4.2 Parsing with Context Free Grammars

Natural language syntax is formally beyond regular languages and the coverage of regular grammars. Thus, for syntactic processing, various parsing algorithms are implemented that use context free grammars (CFG). Simple algorithms like bottom-up or top-down parsers are part of the parser library, as well as an Earley Parser [4], and other types of chart parsers, using agenda-based processing strategies.

In addition to the simple parser implementations, graphical visualization widgets for Racket have been implemented that display balanced syntactic parse trees.

For higher level syntactic processing that makes use of lexical feature structures, first versions of unification parsers are included in the library as well, see [3].

4.3 N-gram models

Various language processing algorithms make use of statistical models of distributional properties of linguistic units, e.g. sounds, morphemes, words and phrases. An approximation of such distributional properties can be achieved using N-gram models.

Various tools for different types of N-gram models are included in the SNLTK. It is possible to create character-based N-gram models from textual data, that are useful for language identification and approximations of syllable structure. In addition, word token-based N-gram model generators exist as well, that extract statistical distributional models over word co-occurrences.

Besides token co-occurrence patterns, bags of tokens or types can be generated from N-gram models as well. Frequency profiles can be calculated in terms of absolute and relative frequencies. Some other language tools which include filtering of functional words, lemmatization and normalization can be applied in different stages of N-gram model generation. Once a N-gram model is generated frequencies can be weighted by tf-idf weight (term frequencyâĂŞinverse document frequency) before generation of vector space models.

Information theoretic measures are included as well. The Mutual Information score for example can be calculated for models and individual N-grams, which is useful for finding correlation between uni-grams or for bi-gram weighting. Other implemented statistical testing functions include chi-squares and t-tests for testing similarities or differences between models.

4.4 Vector Space Models and algorithms for classification and clustering

For various models of distributional properties of tokens and words in text, as well as complex text models, vector space models appear to be very useful. Not only multivariate statistical properties can be processed using such models, but these models are also language independent, i.e. generalize well over multiple languages with fundamentally different properties at various linguistic levels (e.g. syllable structure, syntactic word order restrictions).

One the one hand, simple mapping algorithms of raw text and N-gram models to Vector Space Models are provided in the SNLTK.

For classification and clustering tasks various similarity metrics algorithms are implemented, that are used among others in a K-Means algorithm for text clustering. As similarity metrics, various distance measures like Euclidean Distance and Cosine Similarity are provided. Further scaling algorithms for vector models are provided, including re-ranking on the basis of tf-idf, as well as statistical significance testing on the basis of e.g. chi-square tests.

4.5 Basic language data for various languages

Besides example corpora and word lists for specific languages, in particular stop word lists for various languages are provided. The creation of additional stop word lists is facilitated by a library which exports them as lists or a hashtable data structure. In addition to hashtable based lookup lists, finite state automata can be generated from word lists.

4.5.1 Example corpora

Text collections for various languages are being prepared that facilitate language model training and tuning, as well as testing of existing algorithms.

4.6 Additional components

4.6.1 Basic tree visualization (Racket)

The SNLTK also allow us to use Racket interpreter to generate N-gram models and produce DOT representations of directed graphs based on the underlying N-grams that can be visualized with Graphviz or any other alternative tool for DOT-based graph visualization. The current n-gram2dot release represents frequencies via heavier weights and length of edges and also different weight of nodes [7]. Graphviz can generate different output formats for graphs like PDF, PNG, SVG.

4.7 Documentation

The documentation of the SNLTK library and tools is provided initially in English. All documents are translated to German, Polish and Croatian as well.

The SNLTK documentation contains the library description, as well as example documents related to the use of specific library functions, and algorithm implementation issues. One of the main goals of SNLTK is to provide a detailed documentation of the algorithms, their implementation, as well as different examples of use.

5. CONCLUSION

On the one hand, from our experience, we can conclude that Scheme and current Scheme development environments like DrRacket are very useful for the development of NLP tools, as well as for learning and education purposes.

The SNLTK in its current form is subject to further development and extension. Its further directions are guided by the research and educational interests of the participating researchers and developers.

Future releases of the SNLTK will most likely include more NLP algorithms, as well as machine learning algorithms, related, but not restricted to the domain of cognitive modeling and language learning.

References

6. REFERENCES

[1] Bird, S., Klein, E., and Loper, E. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media.

[2] Clinger, W.D. 2005. Common Larceny. In the *Proceedings of the 2005 International Lisp Conference*, June 2005, pages 101–107.

[3] Dybvig, R.K. 2009. *The Scheme Programming Language.* 4th edition. Cambridge, MA: MIT Press.

[4] Earley, J. 1970. An efficient context-free parsing algorithm, *Communications of the Association for Computing Machinery* 13.2, 94–102.

[5] Feeley, M. 2010. *Gambit-C.* Online documentation, http://www.iro.umontreal.ca/ gambit/doc/gambit-c.html.

[6] Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database.* Cambridge, MA: MIT Press.

[7] Gansner, E., Koutsofios, E., and North, S. 2006. *Drawing graphs with dot. dot UserâĂŹs Manual*, 1–40.

[8] Gazdar, G., Mellish, C. 1990. *Natural Language Processing in LISP: An Introduction to Computational Linguistics.* Boston, MA: Addison-Wesley Longman Publishing.

[9] Grinberg, D., Lafferty, J., Sleator, D. 1995. A robust parsing algorithm for link grammars, *Carnegie Mellon University Computer Science technical report* CMU-CS-95-125.

[10] Hammond, M. 2003. *Programming for Linguists: Perl for Language Researchers*, Blackwell.

[11] Konchady, M. 2008. *Building Search Applications: Lucene, LingPipe, and Gate.* Mustru Publishing.

[12] Mealy, G.H. 1955. A Method to Synthesizing Sequential Circuits. *Bell Systems Technical Journal* 34. 1045–1079.

[13] Sperber, M., Dybvig, R.K., Flatt, M., Straaten, A.v., Findler, R., and Matthews, J. 2010. Revised [6] Report on the Algorithmic Language Scheme. Cambridge University Press.

[14] Miller, G.A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11: 39–41.

[15] Mohri, M. 1996. On Some Applications of Finite-State Automata Theory to Natural Language Processing. *Natural Language Engineering* 1.1.

[16] Mohri, M. 1997. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics* 23.2, 269–312.

[17] Moore, E.F. 1956. Gedanken-experiments on Sequential Machines. Automata Studies, *Annals of Mathematical Studies* 34, 129–153.

[18] Russell, S., and Norvig, N. 2009. *Artificial Intelligence: A Modern Approach.* 3rd edition. Upper Saddle River, NJ: Prentice Hall.

[19] Serrano, M. 1996. *Bigloo user's manual.* Technical Report, Inria.

[20] Serrano, M., and Weis, P. 1995. Bigloo: a portable and optimizing compiler for strict functional languages, *SAS* 95, 366–381.

[21] Wilcock, G., and Hirst, G. 2008. *Introduction to Linguistic Annotation and Text Analytics* (Synthesis Lectures on Human Language Technologies). Claypool Publishers.