*Nordic Journal of African Studies 14(4): 479 – 491 (2005)*

# Toward a Widely Usable Finite-State Morphology Workbench for Less Studied Languages — Part I: Desiderata

ANSSI YLI-JYRÄ
*University of Helsinki* and *CSC - Scientific Computing Ltd.*

## ABSTRACT

Most of the world's languages lack electronic word form dictionaries. The linguists who gather such dictionaries could be helped with an efficient morphology workbench that adapts to different environments and uses. A widely usable workbench could be characterized, ideally, as **generally applicable**, **extensible**, and **freely available** (GEA). It seems that such a solution could be implemented in the framework of finite-state methods. The current work defines the GEA desiderata and starts a series of articles concerning these desiderata in finite-state morphology. Subsequent parts will review the state of the art and present an action plan toward creating a widely usable finite-state morphology workbench.

*Keywords: computational morphology, lexical resources, finite-state methods*

## 1 INTRODUCTION

There is a fundamental need for a **common research infrastructure** for natural language processing. In relation to this, the current work motivates some desiderata for widely usable finite-state based methodology for representing morphological knowledge.

Digital knowledge bases of the morphology of languages – **computational lexicons** and **morphological grammars** are foundational resources for many language technology applications (Daille et al., 2002; Beesley, 2004). They have important applications in written communication, multi-lingual learning environments, speech interfaces, information management tools, and translation of closely related languages. The advent of such utilities as spellers, dictionaries and lemmatizers have already opened new possibilities for internationalization of the basic IT infrastructure, value-added information services and linguistic research.

There is still a large number of **less studied languages** (Oflazer and Nirenburg, 2003) that have not been a subject of study in the discipline of Human Language Engineering. Morphological knowledge bases could still be built for some 7000 natural languages of the world, and to even more dialects and stylistic varieties of languages. We are currently close to the point where we could demonstrate that 100 languages have been studied in the framework of computational morphology, but only a portion of these languages have comprehensive computational lexicons.

Nevertheless, construction of linguistically adequate lexicons for less studied languages requires a lot of human effort. The need to rationalize the human work prompts development of a computerized workbench that could help the linguists create lexicons more efficiently if not fully automatically. Although some lexicons and morphological grammars can be learned automatically from texts (Itai, 1994; Yarowsky and Wicentowski, 2000; Creutz et al., 2005) fully automatic or unsupervised methods are not sufficient. This is due to two reasons. First, the amount of freely available corpora is limited for many of the less studied languages. Second, many of the less studied languages have rich morphologies that are difficult to learn accurately with unsupervised methods.

The article is structured as follows. The next two sections describe the need for common workbench infrastructure in computational morphology. In section 4, we operationalize the wide usability of a finite-state morphology workbench in terms of three desiderata: generality, extensibility and availability. In section 5, we propose a licensing model to be adopted in the software development and discuss strategies for software patents. In section 6, we describe our ongoing research on the paper's topic. We conclude with section 7.

## 2 LINGUISTIC PREFERENCES ARE DIVERSE

The tools for constructing new lexicons should adapt to different linguistic applications. For example, a field linguist building a lexicon or a dictionary of a less studied language needs methodological workbench that helps him or she to be very effective in the lexicon building task, but also in creation of a **human-readable linguistic descriptions** of the morphology and phonology of the language. Some interactive tools for field linguists and lexicographers exist (ShoeBox/Toolbox, WordManager, etc.), but many of them fail to include an adequate support for description of morphology and phonology. Other users of a morphology workbench would emphasize the need for automatic compilers that produce spellers and other simple applications from linguistic descriptions and lexical databases. These users are interested in the performance of the resulting **computational utilities** rather than linguistic elegance. The nature of such utilities vary from spell checking to generation and analysis of phonetic forms in speech processing.

Mechanisms for linguistic description of word forms has been a long-term research area in general linguistics and computational linguistics. As early as in the 5th century B.C., an ancient grammarian, Pāṇini[1] invented a very powerful rule formalism (Deo, 2005). With the formalism, he described Sanskrit morphology with 3959 rules. In the modern times, the quest for the most appropriate methodology for morphological description has inspired new linguistic theories. In addition to three cardinal **models of morphology** (Item and Arrangement, Item and Process,

[1] http://en.wikipedia.org/wiki/Panini_(scholar)

and Word and Paradigm) (Hockett, 1954; Matthews, 1974), there are **phonological theories** (e.g. Generative Phonology, Declarative Phonology, Optimality Theory, Auto-Segmental Phonology). Computational models (e.g. Finite-State Morphology, Unification Morphology, Functional Morphology, DATR Morphology) employ simple mechanisms, such as functions, feature structure unification and **finite transducers** (also known as generalized sequential machines).

Theoretically, finite transducers are inferior to the rewriting system used in Generative Phonology. In early 1970's, C. Douglas Johnson made, however, a practically significant discovery:

> There seem, in fact, to be few phonological processes that exceed the capacity of finite transducers; the ones known to me belong to the very restricted types to be discussed in Chapter 7. (Johnson 1972:56).

The restriction discovered by Johnson surfaced in computational linguistics only after an independent re-discovery of the same restriction in 1981 (see Kaplan and Kay 1994): Kaplan and Kay had designed a transducer compiler for a restriction of Generative Phonology, thus creating a **finite-state based programming language** for linguists.

Kaplan and Kay's experiments on a finite-state based programming language for phonological rules shaped the work of Koskenniemi (1983). Koskenniemi rejected generative formalism and adopted a different theoretical framework, a two-level grammar that is now known as Two-Level Morphology. Today, Koskenniemi's Two-Level Morphology (Koskenniemi, 1983, 1984) and the more general framework of finite transducers (Kaplan and Kay, 1994; Karttunen et al., 1992; Karttunen, 1994, 1993; Beesley and Karttunen, 2003) are undoubtedly the most successful approaches in computational morphology and phonology, providing a well-understood solution for efficient compilation and representation of computational lexicons of word forms. Currently, there are efficient techniques – flag diacritics and compile and replace algorithm – that can handle even non-concatenative morphology.

The technology build on the notion of finite transducers could be used also by general linguists. The need to facilitate linguists at their work has inspired a number of interactive computational workbenches for doing morphology and phonology (Ashby et al., 2001; Bird and Ellison, 1992; Maxwell, 1999; Maxwell et al., 2002; Maxwell, 2003; Bharati et al., 2004; Németh et al., 2004; Novák, 2004).[2]. If computational models based on finite transducers were embedded to an interactive grammar development environment, the well-defined calculus of finite transducers could be used for validation of human-readable grammars. At the same time, it would be crucial that the development environment would abstract away from the low level notions of the finite-state framework that are not familiar to typical general linguists.

[2]Some of the tools mentioned in these bibliographical references are not based on the finite-state framework.

Obviously, a widely usable workbench should be suitable for different tasks regardless of linguistic preferences and approaches, ranging from interactive use by pure linguists to applied language engineering.

## 3 Business Contexts Are Diverse

There are now many finite-state based programming languages (e.g. XFST regular expressions, SFST, LEXC, FSA Utilities) and finite-state based implementations of phonological and morphological theories for lexicon building. Nevertheless, none of them seems to be prevailing at the moment. Apart from the diversity of linguistic preferences in the field, this has to do with the fact that there are both commercially important and commercially less important languages, as well as different user communities:

1. **Commercial users** create proprietary resources — they aim at optimized products with high efficiency and coverage.

2. **Academic linguists** write human-readable grammars of languages — they are interested in linguistic elegance and application of linguistic theories.

3. **Open source developers** create free lexical resources — they are aim at independent and open standards[3] and fast development of lexicons.

It seems to me that the traditional division to commercially important and commercially less important languages is no more clear-cut. On one hand, new resources for commercially important languages are becoming publicly available.[4,5] This happens because of the growing awareness of common good in establishing shared research infrastructures (see e.g. Raffenspenger 2004). On the other hand, minority languages have influenced commercial software development by accelerating their internationalization. For example, spell checking capabilities of word-processing software are being extended to these languages sometimes before the official languages of the same countries.

The following metaphor illustrates why the infrastructure for creating electronic lexicons of the languages in the world must be made available freely to the public:

> The availability of good things should grow according to their importance rather than vice versa: human creatures need oxygen too much to pay for it, but they can pay something for a car.

Obviously, a widely usable morphology workbench should be usable in different situations ranging from open-source projects to proprietary projects.

[3] See e.g. the OpenDocument standard (`http://en.wikipedia.org/wiki/Opendocument`).

[4] `http://www-igm.univ-mlv.fr/~unitex/linguistic_data.html`

[5] `http://www.lexique.org/outils/Lexique-BRMIC.pdf`

## 4 THE DESIDERATA

In order to operationalize the degree of wide usability of the morphology workbench, we will split the criterion of wide usability of the workbench into three parallel desiderata as follows:

1. **Generality:** The workbench should be generally applicable both from the application point of view as well as from the point of view of compatibility. In other words, the workbench should reflect generality in the following ways.

   **(a) Applicability**

   - It supports creating an adequate description of any language.
   - It is an attractive tool for general linguists.
   - It directly support linguistic formalisms and theories and can compile them to regular expressions.
   - It is of the state of the art in linguistics (phonological and morphological theories)
   - It is of the state of the art in computational linguistics (parameter estimation, special operators etc.).
   - It implements new finite-state techniques (weighted automata, semirings, on-demand computation, network optimizations, multiple tapes, etc.)
   - A wide user community can learn to use it.

   **(b) Compatibility**

   - It uses elegant and generally correct algorithms.
   - It is computationally efficient.
   - It can share and exchange data with related tools.
   - It can be ported to different machines.

2. **Extensibility:** The workbench should be easily extensible. This requirement includes the following:

   - It contains interfaces for plug-ins and library calls.
   - It supports development of new ideas for algorithms and formalisms.
   - It contains components that can be reused in new software.
   - It allows adding or changing features of the rule formalism.

- It allows alternative implementations of algorithms and data structures via encapsulation techniques such as generic programming and wrappers.
- Anybody (knowledgeable) can extend all layers of the workbench.
- Anybody can make commercial and non-commercial derivatives of the software.
- It contains necessary plug-in and application programmer's interfaces to non-GNU licensed components.

3. **Availability:** The workbench should be freely available. This requirement includes the following:

   - The tool is freely available in source code and other formats.
   - Anybody has permission to use the source code for any purpose.
   - Everybody has access to developer's and user's essential documentation.
   - It can be ported and rebuilt by the user after hardware upgrades.
   - The user may redistribute the workbench to new users.
   - Commercial products can be developed with or from the workbench.
   - It does not contain patent-restricted methods in essential parts that cannot be dropped.
   - The development version is available to everybody.
   - Anybody is entitled to produce free or commercial support for it.
   - It can be varied for different business situations and needs.
   - It has an OSI approved license[6].

It is possible to argue that the properties of generality, extensibility and free availability (GEA) are closely related with each other and present a successful concept. For example, the TEX typesetting program[7] is a successful piece of software that is publicly **available**. While TEX itself has a fixed design, it is both designed to be programmable and to allow reuse of its source code in new programs – resulting in very powerful **extensibility**. Through its extensibility[8], TEX has been proven very **general**. While it is true that good programs are more than just a combination of the three desiderata, the case of TEX still suggests that generality comes best with extensibility and availability:

[6] `www.opensource.org`

[7] `http://en.wikipedia.org/wiki/TeX`

[8] For example, this article has been typeset using new macros for TEX/LATEX.

| **ad-hoc solutions** ⟶ | **professional solutions** ⟶ | **ubiquitous solutions** |
|---|---|---|
| specific | multi-purpose | general (G) |
| fixed | tailorable | extensible (E) |
| in-house | proprietary | available, common standard (A) |

## 5 SOME FOUNDATIONAL ISSUES

A discussion on the technical requirements and the design architecture of the envisioned workbench would have to answer to the question how a widely usable workbench would differ from the existing proprietary or free solutions. Such **technical discussion** has been, however, postponed to a later article under the current topic.

Regardless of technical design details, the definitions of the GEA desiderata — especially the desiderata of extensibility and availability — bear high resemblance to typical properties of open source software. Therefore, it is clear that the desiderata of extensibility and availability can be satisfied only if the workbench is created in the **open source community**.

Because the envisioned GEA-based workbench will be free software, we need a widely acceptable policy to handle such issues as open source licensing, copyleft compatibility, and software patents. In the following, I will outline how a practical policy could be developed.

### 5.1 CREATING FREE SOFTWARE

We must choose a **licensing policy** that (i) encourages wide contributions to the workbench development, (ii) facilitates a latent copyleft relaxation for compatibility and (iii) empowers the preservation of the workbench as free software when compatibility is not an issue.

Open source licenses are divided to two categories, copyleft licenses[9] and the others. When a license is a **copyleft** license, it requires all modified and extended versions of the licensed software to be free. It does not mean that the software is not copyrighted. Instead, the copyright holder has the right to decide whether the work is licensed with a copyleft license. Within the open source community, many contributors do not like that a modified version of their code is made proprietary by a commercial developer. Therefore, many open source developers choose a copyleft license, GNU General Public License (GPL) or Lesser General Public License, depending on the type of the component they develop.

### 5.2 PREPARING FOR COPYLEFT RELAXATION

By default, GNU LGPL should be preferred to GPL because many potential contributors would also like to reuse the components of the workbench in commercial

[9] `http://www.gnu.org/licenses/licenses.html\#WhatIsCopyleft`

software. Compared to GPL, LGPL allows more freedom in this respect, because an LGPL-licensed library can be linked to a proprietary software.

It may still happen that LGPL-licensed software cannot be combined with other software. In such situations, the **copyright** holder can resolve the conflict by allowing **exceptional uses**. To deal with such situations, it is important that the developers have only a small set of representatives who have the mandate to resolve problematic situations.

If the number of GPL-licensed contributions increases, we must choose a manageable **copyright policy** that allows case-by-case licensing. For this purpose, we might need to adopt a model similar to Sun Microsystem's **Joint Copyright Agreement** (JCA)[10] from the very beginning. Goldman and Gabriel (2005) explain why it is important to adopt such a model from the early stage: it is difficult to impose such agreements later.

A joint copyright agreement works very well in research groups that can jointly choose a representative for its members. However, for individual contributors the legalese of such an agreement might be too much. Because the community and groups need the possibility to make exceptions to the copyleft policy, it is necessary that individual contributions would be licensed under a **permissive open source** license, such as the MIT license[11] or the Boost Software license.[12] The latter license is required, in particular, if the contributed component could later be included to the Boost C++ library.[13] If the individual contributor, however, insists on using a copyleft license, the community can recommend the use of GNU LGPL, because LGPL ensures the largest possible applicability.

## 5.3 Developing a Strategy for Software Patents

We need to develop a strategy for relating the open source code to **software patents**. Many finite-state algorithms (some rewriting and replace operators, algorithms for hyphenation, tokenization, minimization, etc.) are covered by software patents[14] although they may be widely used in the scientific community. If such freely available implementations of patented algorithms were used also in a commercial setting, patent holders can make life very difficult for the distributor of a free software. For example, the Linux distributor Red Hat had to remove all MP3 software from its

[10] `http://www.openoffice.org/copyright/copyrightapproved.html`

[11] `http://www.opensource.org/licenses/mit-license.php`

[12] `http://www.boost.org/more/license_info.html`

[13] `http://www.boost.org/`

[14] Patents for finite-state methods have been granted even in Europe. According to Foundation for a Free Information Infrastructure[15], "While the European Parliament rejected almost unanimously the software patents directive, which would legalese the EPO case law on software patents, the EPO will be able to go ahead with granting these patents, and these patents will now even be Community titles (`http://wiki.ffii.de/ComPat060118En`)."

distribution because of potential licensing conflicts.[16]

Sometimes a holder of patents tactically licenses patented methods to the open source community[17]. A great number of big companies have already started to support open source community and business models based on open source business models (Koenig, 2005).

In these scenarios, the open source community has a passive patent strategy. We may need, however, also an active patent strategy to protect the basis of the open source development.

## 6 FUTURE WORK

The current article discussed the GEA desiderata in finite-state morphology. The space would not allow much wider discussion in one article. The author is working on further articles on the current topic. After completion, the intent is to submit them to this journal. The forthcoming articles would address the following needs:

First, there is a need to demonstrate the success and limitations of the finite-state morphology according to the GEA desiderata. The author has been working on a survey on the state of the art in the finite-state morphology. The survey includes more than 75 languages to which finite-state methods have been applied[18], some 20 widely available finite-state based tools and open projects, and catalogs a number of extensions to the widely implemented finite-state based techniques.[19] According to the survey, an ideal or promising finite-state based realization of the GEA desiderata is still non-existent because the GEA desiderata might be too optimistic to be within the reach of normal corporate software development. The author has gathered, a list of realistic improvements that could be implemented in an open source project. Collaborative implementation of such features could help to realize the GEA desiderata to a greater degree.

Second, there is a need for an action plan. In TWOLDAY 2005, the audience provided a wealth of constructive feedback to the author, on the basis of his initiative (Yli-Jyrä, 2005). Furthermore, the author have consulted networks of finite-state experts. There seems to be a wide interest in collaborative work on a finite-state based infrastructure for morphology. An important aspect of an effective action plan is the integration of the ongoining efforts to build finite-state libraries and finite-state compilers. The layered architecture sketched earlier (Yli-Jyrä, 2005) will be

[16] `http://www.iusmentis.com/computerprograms/opensourcesoftware/patentrisks/`

[17] For example, IBM gave 500 patents to the OS community although this costs 10-12 million dollars per year. According to Bob Sutor from IBM, IBM believes that this is an investment to expected business opportunities, see `http://www.coss.fi/fi/ajankohtaista/sutor_navin.html`.

[18] We have counted even grammar fragments.

[19] The author would gratefully acknowledge further pointers to descriptions of less studied languages and open source code bases using finite-state methods. The current list is available from the author.

discussed and elaborated in the action plan.

## 7 THE SUMMARY

In this article, we have argued that there is a need for a widely usable finite-state morphology workbench. Then, we have tried to analyzed what it might mean to have an ideal, "widely usable" workbench. Finally, we elaborated a few fundamental issues related to licensing of software. There are many relevant issues that we did not try to outline in this article, due to space limitations.

In forthcoming articles, we plan to elaborate some other aspects in the implementation of a widely usable finite-state morphology workbench and research infrastructure.

## 8 ACKNOWLEDGEMENTS

The author is grateful to the TWOLDAY 2005 audience for critical remarks and to many experts for later email correspondence. Kimmo Koskenniemi, Krister Lindén, Andrś Kornai, Arvi Hurskainen, Michael Cochran, Howard Johnson, Arto Teräs and Elias Aarnio have personally provided useful feedback based on some extended, unpublished versions of the article. Portions of an extended version were used as a basis for the current article.

## REFERENCES

Ashby, Simone, Julie Carson-Berndsen, and Gina Joue. 2001.
*A testbed for developing multilingual phonotactic descriptions*, In **Proceedings of Eurospeech 2001**, page 4p.
`http://citeseer.ist.psu.edu/ashby01testbed.html`.

Beesley, Kenneth R. 2004.
*Morphological analysis and generation: A first-step in natural language processing*, In **First Steps in Language Documentation for Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, Proceedings of the SALTMIL Workshop at LREC 2004**, pages 1–8.
`http://isl.ntf.uni-lj.si/SALTMIL/`.

Beesley, Kenneth R. and Lauri Karttunen. 2003.
**Finite State Morphology**, CSLI Studies in Computational Linguistics. Stanford, CA, USA: CSLI Publications.

Bharati, Akshar, Rajeev Sangal, Dipti M. Sharma, and Radhika Mamidi. 2004.
*Generic morphological analysis shell*, In **First Steps in Language Documentation for Minority Languages: Computational Linguistic Tools**

**for Morphology, Lexicon and Corpus Compilation, Proceedings of the SALTMIL Workshop at LREC 2004**, pages 40–43.
http://isl.ntf.uni-lj.si/SALTMIL/.

Bird, Steven and T. Mark Ellison. 1992.
*A phonologist's workbench*, In **Proceedings of the 15th International Conference of Linguists**. Quebec, Canada.
ftp://ftp.cis.upenn.edu/pub/sb/papers/be92/be92.pdf.

Creutz, Mathias, Krista Lagus, Krister Lindén, and Sami Virpioja. 2005.
*Morfessor and hutmegs: Unsupervised morpheme segmentation for highly-inflecting and compounding languages*, In **Proceedings of the Second Baltic Conference on Human Language Technologies**, pages 107–112. Tallinn.
http://www.cis.hut.fi/mcreutz/papers/Creutz05hlt.pdf.

Daille, Béatrice, Cédile Fabre, and Pascale Sébillot. 2002.
**Applications of Computational Morphology**, pages 210–234.
Somerville, MA: Cascadilla Press.

Deo, Ashwini. 2005.
*Derivational morphology in inheritance-based lexica: insights from Pāṇini*, manuscript. Department of Linguistics, Stanford University.
www.stanford.edu/%7Eadeo/morph.pdf.

Goldman, Ron and Richard P. Gabriel. 2005.
**Innovation Happens Elsewhere. Open Source as Business Strategy**. Morgan Kaufmann.
http://dreamsongs.com/IHE/IHE.html.

Hockett, Charles. 1954.
*Two models of grammatical description*, **Word** 10:210–231.

Itai, Alon. 1994.
*Learning morphology – practice makes good*, In R. C. Carrasco and J. Oncina, eds., **Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21–23, 1994**, vol. 862 of *Lecture Notes in Artificial Intelligence*, pages 5–15. Springer-Verlag.
http://citeseer.ist.psu.edu/itai94learning.html.

Johnson, C. Douglas. 1972.
**Formal Aspects of Phonological Description**, no. 3 in Monographs on linguistic analysis. The Hague: Mouton.

Kaplan, Ronald M. and Martin Kay. 1994.
*Regular models of phonological rule systems*, **Computational Linguistics** 20(3):331–378.
http://acl.ldc.upenn.edu/J/J94/J94-3001.pdf.

Karttunen, Lauri. 1993.
*Finite-state lexicon compiler*, Technical Report ISTL-NLTT-1993-04-02, Xerox Palo Alto Research Center.
http://content-analysis/fssoft/docs/lexc-93/lexc93.html.

Karttunen, Lauri. 1994.
*Constructing lexical transducers*, In **15th COLING 1994, Proceedings of the Conference**, vol. 1, pages 406–411. Kyoto, Japan.

Karttunen, Lauri, Ronald M. Kaplan, and Annie Zaenen. 1992.
*Two-level morphology with composition*, In **14th COLING 1992, Proceedings of the Conference**, vol. I, pages 141–148. Nantes, France.
http://acl.ldc.upenn.edu/C/C92/C92-1025.pdf.

Koenig, John. 2005.
*Seven open source business strategies for competitive advantage*, Electronic document, Riseforth Inc.
www.riseforth.com/images/Seven%20Strategies%20-%20Koenig.pdf.

Koskenniemi, Kimmo. 1983.
**Two-level morphology: a general computational model for word-form recognition and production**, no. 11 in Publications. Helsinki: Department of General Linguistics, University of Helsinki.

Koskenniemi, Kimmo. 1984.
*A general computational model for word-form recognition and production*, In **10th COLING 1984, Proceedings of the Conference**, pages 178–181. Stanford, California.
http://acl.ldc.upenn.edu/P/P84/P84-1038.pdf.

Matthews, P. H. 1974.
**Morphology**. Cambridge University Press.

Maxwell, Michael. 1999.
*A new program for doing morphology: Hermit Crab*, **Notes on Linguistics** 2(1):11–36.
www.sil.org/computing/hermitcrab/nol_hc_article.doc.

Maxwell, Mike. 2003.
*Incremental grammar development using finite-state tools*, In **EACL 2003 Workshop on Finite-State Methods in Natural Language Processing, Proceedings of the Workshop**, pages 51–58. Agro Hotel, Budapest, Hungary.
http://papers.ldc.upenn.edu/EACL2003/IncrementalChanges.pdf.

Maxwell, Mike, Gary Simons, and Larry Hayashi. 2002.
*A morphological glossing assistant*, In **Proceedings of the International LREC Workshop on Resources and Tools in Field Linguistics**.

http://papers.ldc.upenn.edu/LREC2002/Morph_Gloss_Asst.pdf.

Németh, László, Viktor Trón, Péter Halácsy, András Kornai, András Rung, and István Szakadát. 2004.
*Leveraging the open source ispell codebase for minority language analysis*, In **First Steps in Language Documentation for Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, Proceedings of the SALTMIL Workshop at LREC 2004**, pages 56–59.
http://isl.ntf.uni-lj.si/SALTMIL/.

Novák, Attila. 2004.
*Creating a morphological analyzer and generator for the Komi language*, In **First Steps in Language Documentation for Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, Proceedings of the SALTMIL Workshop at LREC 2004**, pages 64–67.
http://isl.ntf.uni-lj.si/SALTMIL/.

Oflazer, Kemal and Sergei Nirenburg, eds. 2003.
**Language Engineering of Lesser-Studied Languages**, vol. 188 of *NATO Science Series: Computer & Systems Sciences*. Amsterdam, The Netherlands: IOS Press.

Raffenspenger, Carolyn. 2004.
*Funding research for the common good*, **The Environmental Forum** page 14.
Reprinted by permission at www.sehn.org/pdf/jul-aug04.pdf.

Yarowsky, D. and R. Wicentowski. 2000.
*Minimally supervised morphological analysis by multimodal alignment*, In **38th ACL 2000, Proceedings of the Conference**, pages 207–216. Hong Kong.
http://www.cs.jhu.edu/~yarowsky/pubs.html.

Yli-Jyrä, Anssi. 2005.
*An initiative for an open and extendible finite-state morphology workbench*, Talk presented at TWOLDAY 2005.
www.ling.helsinki.fi/events/TWOLDAY2005/.