Proceedings of the Language Technologies for All (LT4All) , pages 379–383
Paris, UNESCO Headquarters, 5-6 December, 2019.

Rich Morphology, No Corpus – And We Still Made It. The Sámi Experience

Sjur Nørstebø Moshagen, Trond Trosterud
UiT The Arctic University of Norway
Department of Language and Culture
{sjur.n.moshagen, trond.trosterud}@uit.no

Abstract

The article presents an infrastructure for building grammar models and language technology applications for indigenous languages, i.e. for languages with a too complex grammatical structure and a lack of the huge amount of corpus material being necessary for mainstream language technology approaches to work. The infrastructure and grammar models provide a large array of applications for indigenous languages. The main problem for indigenous languages turns out to be structural hurdles set up by the major software providers. The article presents these hurdles, as well as a way of overcoming them.

Keywords: language technology, indigenous languages, morphology-rich languages, integration, application programming interfaces, localisation, language resource policies, standards

Čoahkkáigeassu (in North Saami)

Artihkal čájeha infrastruktuvrra, mii hukse grammatihkkamálliid ja giellateknologalaš prográmmaid eamiálbmotgielaide dahje gielaide, main lea kompleaksa giellaoahpalaš struktuvra. Eamialbmotgielain váilot maid teakstačoakkáldagat, mat leat dárbbašlaččat váldogielaid giellaoahpalaš reaidduid huksemii. Infrastruktuvrra ja giellamálliid bokte lea vejolaš hukset máŋggaid prográmmaid dávviguovlluid eamiálbmotgielaide. Váldováttisvuohta dál lea, ahte stuorra prográmmafitnodagat leat huksen prográmmaideaset nu, ahte ii leat vejolaš lasihit eamiálbmotgielaid heivehemiid fitnodagaid prográmmaide. Artihkal čájeha makkár váttisvuođaid birra lea sáhka, ja mo daid sáhttá čoavdit.

1. Introduction

This article presents a way of making grammatical analysers and language technology tools for minority languages with a complex grammatical structure. The infrastructure is hosted at The Arctic University.

The article is organised as follows: First we present our modus operandi, philosophy, scope of work and our major results. The next section presents the major problems facing language technology for minority languages, where integration of our solutions in major writing tools is a key point. The following section presents a solution to these problems, in the form of a manifesto for open language technology. Finally comes a conclusion.

2. Our modus operandi

We work together with the language community, and use rule-based technology and traditional linguistic analysis. In principle, one native speaker is enough, although we always try to involve the people or organisations that have the confidence of or are appointed by the language community to handle language planning.

Our models are not trained on any corpus, and the data sparsity bottleneck thus does not exist. Our technology supports the most complex morphology and morphophonology, but also works well for less complex languages.

By being grammar-based, we are able to offer a three-part cooperation, where all parties have something to gain: Participating in a project building language models within our infrastructure, university-based linguists will get the opportunity to test their grammatical generalisations on a full scale. Language activists will get the tools they need in order to revitalise the language, and programmers will get the opportunity to apply their programs on a new language. Opening up for a cooperation with university linguists is of crucial importance: Each language with a formalised orthography has at least one linguist devoting his or her life to this language. A grammar-based approach will offer this linguist good reasons for joining the project.

2.1. Open source, rule-based technology

In order to model the complex grammar of indigenous languages, we use the Helsinki Finite State Technology, Hfst[1], a tool set being source-code compatible with (Beesley and Karttunen, 2003), and ultimately going back to (Koskenniemi, 1983). It is capable of modeling natural languages in all its complexity, in an explicit and transparent way. To disambiguate morphological homonymy and adding syntactic function and thematic roles, we use Constraint Grammar ((Karlsson, 1990), (Karlsson et al., 1995)) and the compiler VISLCG3 [2].

The net result of this is a set of powerful grammatical analysers and generators for circumpolar and other languages[3]. In addition to providing a source of information on the grammar of these languages, the models are integrated in a language-independent infrastruc-

[1] `http://github.com/hfst`
[2] `http://visl.sdu.dk/cg3.html`
[3] An interface for using these programs can be found at `http://giellatekno.uit.no`

ture ((Moshagen et al., 2013)). By means of this infrastructure we are able to build a wide range of applications and tools. This includes a set of text proofing tools[4], programs for language learning[5] and for machine translation[6].

2.2. Mainly circumpolar languages

We primarily work with Saami languages, but also with other Uralic languages, with Greenlandic and Faroese, and with First nation languages in Canada (see e.g. (Arppe et al., 2016). All but Faroese are known for very to extremely rich morphology.

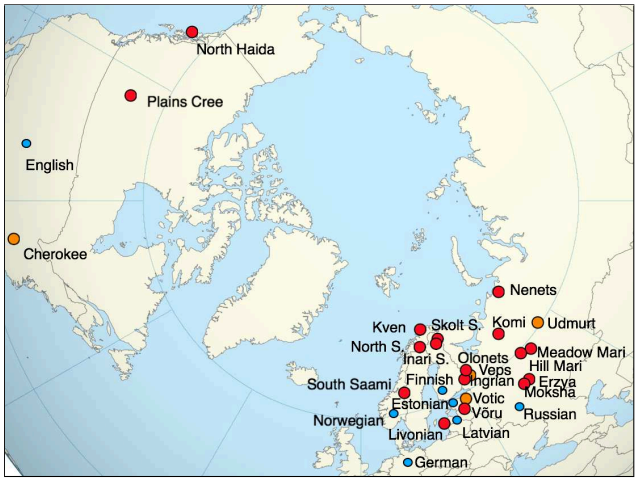

Figure 1: Morphology-enabled dictionaries in the present infrastructure

2.3. Lots of tools

Using these language models, we build spelling checkers, grammar checkers, mobile keyboards with spellers, rule-based machine translation, and offer analysed (i.e. machine-annotated) corpora[7]. We also make intelligent computer-assisted language Learning tools and dictionaries with grammars (cf. (Johnson et al., 2013), an overview of the languages is given in Figure 1).

Both the infrastructure, the linguistic resources, and most of the corpus texts are available as open source. Some corpus texts are not open, due to copyright issues, but single quotes are available via the corpus interface.

3. So we made it - but still?

As described above, we have developed a lot of tools, covering key areas for indigenous languages. Still, our tools are not for everyone, not because they are not available – all our tools are open source and free for everyone – but because the platforms and environments people are using are not open for all languages.

There are numerous examples of such hindrances, from the very low-level language codes to the highest level speech processing API's. The following are but a few of the issues we have met.

For example, our spelling checkers do not work in Chromebooks, and it is not possible to make our spellers work in Google Docs the same way as Google's own tools with red squiggles and right-click functionality. The same is true for MS Office 365. These are problems created by Google and Microsoft – we know that our spellers work, but we are not allowed to put them to use. There might not be an API for providing speller services (e.g. in Chrome OS), or Microsoft and Google have not made available systems for adding third-party spellers to their web-based Office suites.

Another example is combining diacritics in Unicode. These are often unreadable when used in indigenous languages, cf. Figure 2. The explanation is an example of colonial structures manifested in basic language technology: the Unicode consortium has established a principle that no new precomposed letters can be added to the standard, and that new base character + diacritic combinations should be handled by a dynamic composition mechanism – the diacritic is automatically placed on the base character in the optimal position.

Although this sounds like a nice way of keeping the size of Unicode within managable boundaries, in practice it does not work, and it hits indigenous and minority languages only. The problem is that the combining machinery varies by each implementor, giving inconsistent results. One can never trust that the visual appearance on one own's computer will be the same as on the reader's. And often the result is just gibberish.

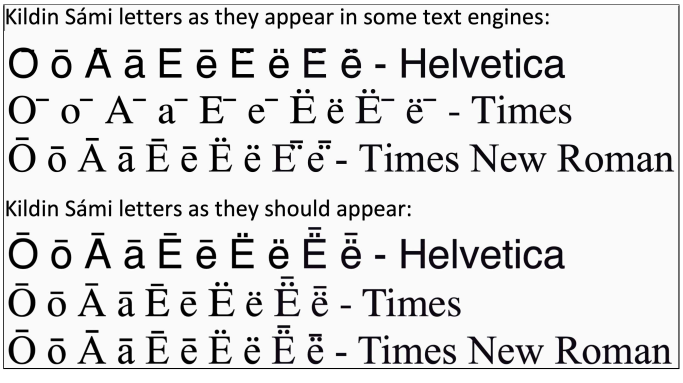

Figure 2: How dynamic diacritics often render in popular text engines (top). Correct rendering below.

Why is it only hitting minority and indigenous languages? Because all majority languages are covered by the existing precomposed letters, so none of the developers really get to see the problems in daily use. It is clearly so, since the bugs illustrated above have been consistent and persistent for at least 10 years.

While it is easy to criticise the makers of the text ren-

[4]Cf. Divvun, `http://github.com/divvun`, see also (Antonsen, 2012), (Wiechetek et al., 2019)

[5]`http://oahpa.no`

[6]For machine translation we use the Apertium formalism (`http://github.com/apertium`), cf. (Antonsen et al., 2017) for a presentation

[7]`http://gtweb.uit.no/korp`

dering engines, one could as well blame the Unicode standard for the situation. Why should new precomposed letters be banned from the standard? Space is cheap nowadays, and adding some extra letters based on existing symbols to a font should also be quite cheap. One could assume that in high-quality, broad-coverage fonts the issue would then be resolved. One could still keep the present, dynanmic composition as a fall-back for less developed fonts, and there will still be cases where dynamic compounding is an acceptable solution. Seen from the perspective of minority languages, whether the issue is solved by better font rendering or by extending the characters repertoire of Unicode is not important. The point is that it should be solved, and not be kept in a situation where neither solution works.

Dictionaries and easy access to definitions, translations and usage examples are very important to minority and indigenous language communities, much more so than for majority languages. And the ability to look up a word in text and immediately get an explanation and translation is crucial in language revitalisation, all the while these languages typically have complex inflections making a direct lookup based on the word form often challenging.

Many operating systems, both desktop and mobile, do have such functionality built-in. Unfortunately it is often either locked down, or of very limited use because there is no way to provide morphological analysis and disambiguation of the input text. So while there is plenty of support for English speakers wanting to learn or needing help in understanding German on e.g. iOS, there is no way the same service can be provided to indigenous and minority languages. It *is* possible to provide *similar* functionality via other means, but that includes extra steps for the users, steps that in practice is a blocker for actual use. Why can't there be a dictionary lookup API available to anyone and with hooks for morphological analysis, providing user services the same way as Apple's own licensed content? Why does third party content have to be treated like third class citizens?

There are a number of settings in which it is desirable to have localised software, including whole operating systems. But most software is not easily localisable, or independently localisable at all. And if it is, it is usually a non-trivial task to distribute the localisation to the users. And if you are able to overcome all these hurdles, you will eventually find that your language is not listed in the language preferences of your operating system, so that the localisation is either unavailable, or the localised software has to resort to special settings to make it available. On top of that the localisation process varies by software and operating system, and is very time consuming despite software text strings usually following a simpler syntax and a lot of repetition. The end result is that most software and operating systems are not localised beyond the dominating languages, and often they can't be localised even if one wants to.

Speech technology is one of the hotspots of language technology these days, and there are academic papers on how speech technology can help overcome the digital divide for indigenous languages by just skipping the written mode, cf. (Palkar et al., 2012). The problem is just that – even if you succeed in building that fantastic speech recognition + machine translation system – you can't make it work where the users are. That is, you can't add your own voice to the Android phones used by the language community, and chances are that the language as such is not even recognised by the operating system. And the speech technology API's are most likely closed behind an unfriendly license, or not available at all.

These are just some examples of issues meeting indigenous and minority langauge communities. The language technology groups at UiT are just two of many working to improve the situation for these language communities, but due to the issues mentioned above, *we just can't provide the tools and services our users want!*

To be clear, issues like the above are *not* restricted to the software providers mentioned, the providers are just examples. The problems are found everywhere in the software industry, including in major open-source projects.

4. Solution: A Manifesto for Open Language Technology

To solve the issues discussed in the previous section, we propose a set of simple software development principles, dubbed a *Manifesto for Open Language Technology*. The four principles are:

Open localisation: all software should be localisable independently of the producer of the software

Open interfaces: all language-related programming interfaces should be open by default

Open resources: all language resources should be open and accessible for everyone, given the permission of the language community

Accessible standards: language-related international standards should be respected, fully implemented and implementations should be regularly updated

We'll elaborate on these principles in the following sections.

4.1. Open localisation

The software belongs to its creators, but one could argue that the user interface language belongs to the language community, or rather, that any langauge community should have an independent right to localise any software they deem necessary to their language, *without asking the creator of the software*. In fact, the localisation of software and access to localisations should be made such that the software creator should not need to be involved at all.

Given that the major software platforms nowadays have their own app stores, it should be possible to add something like a *locale store* to it, a place where users can get and install localisations for any language they want. And the localisation packages should include localisations for any piece of software that has been localised into that language, be it the OS itself or any first or third party software package.

The first principle says that the language belongs to the language community, and that should also be true for localisation.

4.2. Open interfaces

Modern digital devices usually come with a lot of linguistic features, from spellers to digital assistants. Most, if not all, of these features have an API at some level. But these API's are not equal: some are open and free, such as speller engine API's, others are behind license bars, such as a lot of speech technology API's, and some are not public at all, such as the API's for adding speech assistant support for new languages.

The major software houses will never make language technology for most of the world's languages, and that is fine, they don't have to. What is *not* fine is that they don't allow the language communities to develop that technology themselves, by locking all needed API's behind bars of various kinds.

The second manifesto principle says that all API's related to language technology should be open and accessible to any language, no questions asked.

This principle does not mean that the language technology itself has to be open, it just says that if the OS vendor does not provide support for language A, it should not stop others from providing that support, and the support should be transparent for the users.

4.3. Open resources

Building linguistic resources for a language is time consuming independently of the actual technology being used. It is vital that the language resources belongs to and are in control of the language community, both for legal and ethical reasons. Too many times it has happened that language resources for an indigenous language has been owned by a private entity, blocking reuse of those resources in settings not directly benefitting that private entity.

The best way to ensure this is to always develop linguistic resources using an open license, although that must in the end be decided by the language community. The third manifesto principle says exactly that: language resources should be open independently of private entities, and the license and openness should be decided by the language community.

4.4. Accessible standards

There are a number of digital standards relating to human languages. And the standards usually take all languages of the world into consideration. But unfortunately these standards are not always implemented in full, and they are thus unreliable and unaccessible.

The most visible example is the ISO 639 series of language codes, especially the 639-3 language codes that cover in principle every language on earth. What is lacking is support from the software industry, especially the operating system developers.

All OS's do recognise these codes as language codes, but that's it. For most languages, its code is not recognised, just that it is *some* language code. This means, for example, that most languages:

- are not known to the system, and can't be a preferred locale
- show up as language *codes* if you install tools for them, not with the language *name*
- have to be hidden behind the *other* language code to be recognised on some systems
- can not be used for speech technology applications

All digital standards relating to human languages, especially the ISO 639 standard, should be treated the same way as Unicode: be fully implemented, and updated regularly both by the standard bodies and by the OS vendors. This is what the last principle is all about.

5. Conclusion

We have developed an infra and tools for a number of morphologically complex languages. We have shown that LT tools are possible for any language, irrespective of available corpus and grammatical complexity, but we have also shown that there is a lot of issues left, issues caused by lack of support and direct neglect from the major players in the software industry. Finally, we propose a short but pointed list of software development principles to address the issues we have identified. The ultimate goal is to achieve *indigenous self-determination in the digital realm*.

6. Acknowledgements

We would like to thank our collegues at Divvun and Giellatekno in Tromsø for the fruitful cooperation over the years, our international partners, a.o. The Techno Creatives, Gramtrans, Trigram, Kvensk institutt, Oqaasileriffik, and FU-lab (Syktyvkar). The work is mainly financed by The Arctic University of Norway and by the Norwegian Ministry of Local Government and Modernisation. Important steps forwards have been made possible by project support from the Research Council of Norway and by Kone Foundation of Finland, as well as by the Social Sciences and Humanities Research Council of Canada.

7. Bibliographical References

Antonsen, L., Gerstenberger, C., Kappfjell, M., Rahka, S. N., Olthuis, M.-L., Trosterud, T., and Tyers, F. M. (2017). Machine translation with North

Saami as a pivot language. In *Proceedings of the 21st Nordic Conference on Computational Linguistics, NoDaLiDa, 22–24 May 2017, Gothenburg, Sweden*, volume 29 of *NEALT Proceedings Series*, pages 123–131. Linköping University Electronic Press.

Antonsen, L. (2012). Improving feedback on L2 misspellings – an FST approach. In *Proceedings of the SLTC 2012 workshop on NLP for CALL, Lund, 25th October, 2012*, volume 80 of *Linköping Electronic Conference Proceedings*, pages 1–10. Linköpings universitet.

Arppe, A., Lachler, J., Trosterud, T., Antonsen, L., and Moshagen, S. N. (2016). Basic language resource kits for endangered languages: A case study of Plains Cree. In *Proceedings of the LREC 2016 Workshop CCURL 2016 – Towards an Alliance for Digital Language Diversity*, pages 1–8. LREC.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology*. Studies in Computational Linguistics. CSLI Publications, Stanford, California.

Johnson, R., Antonsen, L., and Trosterud, T. (2013). Using finite state transducers for making efficient reading comprehension dictionaries. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013); May 22–24; 2013; Oslo University; Norway*, volume 16 of *NEALT Proceedings Series*, pages 59–71. Linköping University Electronic Press.

Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A. (1995). *Constraint Grammar. A Language-Independent System for Parsing Unrestricted Text*. Natural Language Processing. Mouton de Gruyter, Berlin, New York.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In *COLING '90 Proceedings of the 13th conference on Computational linguistics*, volume 3, pages 168–173, Helsinki.

Koskenniemi, K. (1983). *Two-level Morphology. A General Computational Model for Word-forms Production and Generation*, volume 11 of *Publications of the Department of General Linguistics*. University of Helsinki.

Moshagen, S. N., Pirinen, T., and Trosterud, T. (2013). Building an open-source development infrastructure for language technology projects. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013); May 22–24; 2013; Oslo University; Norway*, number 16 in NEALT Proceedings Series, pages 343–352. Linköping University Electronic Press.

Palkar, S., Black, A., and Parlikar, A. (2012). Text-to-speech for languages without an orthography. In *Proceedings of COLING 2012: Posters*, pages 913–922, Mumbai, India, December. The COLING 2012 Organizing Committee.

Wiechetek, L., Moshagen, S., Gaup, B., and Omma, T. (2019). Many shades of grammar checking – launching a constraint grammar tool for north sámi. In Eckhard Bick et al., editors, *Proceedings of the NoDaLiDa 2019 Workshop on Constraint Grammar: Methods, Tools and Applications, Turku, Finland*, volume 33 of *NEALT Proceedings Series*, Linköping, Sweden. Linköping University Electronic Press.