GiellaLT: an infrastructure for rule-based language technology tool development

Pirinen, Flammie A
UiT Arctic University of Norway
flammie.pirinen@uit.no

Trosterud, Trond
UiT Arctic University of Norway
trond.trosterud@uit.no

Moshagen, Sjur N.
UiT Arctic University of Norway
sjur.n.moshagen@uit.no

Wiechetek, Linda
UiT Arctic University of Norway
linda.wiechetek@uit.no

Abstract

Currently, machine learning is presented as the ultimate solution for language technology regardless of use case and application, however, it requires as a starting point a massive amount of curated linguistic data in electronic form that is expected to be high quality and representative of the kind of language usage that the tools will follow. For minority and indigenous languages, this can be an insurmountable task, as digital materials of the necessary sizes do not exist and can not easily be produced. In this article we present an approach we have successfully used for supporting indigenous languages to survive and grow in digital contexts for years, and describe the potential of our approach for African contexts. Our technological solution is a free and open-source infrastructure that enables language experts and users to cooperate on creating linguistic resources like dictionaries and grammatical descriptions. In addition we provide language-independent frameworks to build these into applications that are needed by the language community.

Keywords: Rule-based LT, keyboards, proofing tools, LT infrastructure, Extremely low-Resource LT

Introduction

Machine learning (ML) approaches have dominated *Natural Language Processing* (NLP) during the last two decades. They typically require a big amount of noise-free data, which does not exist for 99% of the 7,000 world's languages, either due to the small number of writers or a young written norm. Even if a small language community like the Inari Sámi one in Finland is exceptionally productive in text writing (10,533 words/speaker as opposed to 1,440 words/speaker for Swedish) the amount of text recommended for regular machine learning approaches ("A 3.4 billion word text corpus was used for the original BERT-Large, so it is worth training with a data set of this size."[1]) cannot be reached. The same counts for language communities with many speakers but with a low degree of literacy in this language or simply missing written language domains or a bilingual context where the majority language only is used as a written language.

In this article, we present óur infrastructure – *GiellaLT* – presently hosting language models for more than 130 languages and building a number of language technology tools that are useful for any language community. *GiellaLT* is based on an alternative to corpus-based language technology – knowledge-based, also known as rule-based, language technology. The technology is chosen for its ability to support languages with no earlier digital presence. As long as a project has access to a native speaker and a linguist, maybe even in one and the same person, useful tools can be made. Digital resources are always welcome and will help speed up the development work, but they are not a requirement.

A language community with no or little earlier digital presence also needs different types of tools than languages with billion-word corpora. Every language community is unique, but for ones that have or are aiming for a written tradition digitised, typically the first tool to develop is a keyboard, including mobile keyboards nowadays, to be able to enter text correctly and efficiently. No machine learning can create a keyboard layout or discuss language com-

munity needs.

The GiellaLT infrastructure provides all the building blocks to get started on the language work right away, based on open-source technologies and solutions. There is proven integration with most existing systems and platforms, saving huge amounts of time, money, and resources for newcomers. This is especially important for many indigenous language communities, which would not have the resources to develop underlying technologies and integration solutions on their own. By separating language-independent parts from language-specific ones, the cost of developing the language-independent parts can be shared by everyone, or be carried by communities with enough resources. Being open source we are also maintaining an approach that empowers the language communities in a way that retains their ownership of the language and linguistic data they are working on. In the free open-source model, there is no large danger that someone working on the language models simply acquires the linguistic data from the community for free in order to sell it back to them at a higher price.

One final consideration that our rule-based approach has over the machine-learnt models is one of efficiency: a neural language model has to be trained on a system with at least one GPU for over a period of days or weeks and should probably also run on similar hardware, or accessed over high-speed internet. The rule-based language tools can be compiled on a low-end home computer and run locally on even lower-end mobile phones, which is almost a necessity for many writers' tools. As an added bonus the approach is energy-efficient which is something neural models at the moment still struggle with, c.f. Treviso et al. (n.d.).

GiellaLT – A multilingual infrastructure for everyone

The foundation for the work presented in this article is the multilingual infrastructure *GiellaLT*, which includes numerous languages that have little or no data, a rare case in the NLP world. Everything produced in the *GiellaLT* infrastructure is under free and open licences and freely available. The corpora are available with free licensing where possible. The infrastructure is split code-wise into three GitHub organisations: *GiellaLT* containing the language data for each language, *Divvun* containing language-independent code for the infrastructure, and *Giellatekno* for corpus infrastructure. End user tools served by the Divvun group are at *divvun.no* & *divvun.org*, and tools served by the Giellatekno group at *giellatekno.uit.no*, both at *UiT—Norway's Arctic University*.

The basic requirement for developing language tools in GiellaLT is an orthography that is either defined beforehand or is being defined in making the language tools. Access to a printed dictionary is of great help, even more so if it is available electronically. An existing grammatical description is also of tremendous help, but not a requirement.

We build systems that include lexical data as well as rules governing morphophonology, syntax, and semantics as well as a number of application-specific information, e.g. grammatical rules for grammar checking, phonetic rules for *Text-To-Speech* (TTS), and so forth.

The language-dependent work is done within the infrastructure in language-specific repositories, the language-independent features and updates that are relevant to all languages are semi-automatically merged as they are developed. To ensure that language-independent and common features and updates do not destroy existing language data or degrade the language tools, we enforce a rigorous continuous integration-based testing regime. The current system for testing is a combination of our long-term investment in testing within the infrastructure locally for developers—combined with modern automatic testing currently supplied by GitHub actions.

Another part of the *GiellaLT* philosophy is that of reusable and multi-purposeful resources, cf. Antonsen et al. (2010). This is true for all of our work, from corpus collection to cross-lingual cooperation and is crucial for the sustainability of the work in indigenous and lower-resourced languages

where language experts' work time is scarce and precious.

Despite the lack of data, there are high-level tools in *GiellaLT* such as *machine translation* (MT), text-to-speech *TTS*, spelling and grammar checkers, and more, that have been very well received in the language communities. This would not have been possible without first developing basic tools such as keyboards, morphological analysers, and spelling checkers.

GiellaLT for African languages

In this section, we go through the NLP tools GiellaLT infrastructure provides for language users. We list here a subset of the applications that we have found are most useful for the language communities we work with, in language support and revitalisation work: Keyboards and spelling and grammar checking and correction, dictionaries and machine translation. We also provide tools for speech technology, however, this is not discussed in detail in this article, for example, c.f. Hiovain-Asikainen & Moshagen (2022). Furthermore, the linguistic resources that are used as a basis of end-user tools like morphological analysers, are a key resource for digital humanities work on the language: tokenisation, morphosyntactic analysis and glossing for example.

Languages for which Microsoft and Google do not make language technology solutions are vulnerable to technological changes. Closed source programs for such languages run the risk of becoming unusable as word processors or operative system change. Companies behand these programs may then either go bankrupt or change their focus to other areas. Being closed source, the work behind these solutions is then lost. The GiellaTL solution to this is to keep both the linguistic software and the software needed for integrating it in various applications as open source. This means that scarce resources may be reused, without the risk of loosing work. Open access to language independent software also makes it easier to build solutions for lesser-used languages.

Keyboards

Most African languages are written with the Latin alphabet. Many of them have letters outside the A-Z range and even more, have extensive systems of diacritical symbols. On top of that comes the challenge of how click sounds are treated in the San languages, with letter symbols resembling punctuation marks.

Each of these languages does need its own keyboard setup. When language technology tools to an increasing extent are linked to keyboard setups, languages using only the letters A-Z will need their own keyboard to invoke language technology tools for the appropriate language.

In order to meet this challenge, the GiellaLT infrastructure comes with a pipeline for making keyboards and installing them and their corresponding language technology tools on different platforms. The core of the pipeline is the `kbdgen` tool, with which one can easily specify a keyboard layout in a *YAML* file, mimicking the actual layout of the keyboard. The listing below shows the definition of the Android mobile keyboard layout for Lule Sámi. The tool takes this definition and a bit of metadata combines it with code for an Android keyboard app, compiles everything, signs the built artefact and uploads it to the Google Play Store, ready for testing.

```
modes:
    android:
      default: |
            á w e r t y u i o p å
            a s d f g h j k l ø æ
              z x c v b n m ŋ
```

`kbdgen` supports generating keyboard apps or installer packages for Android, iOS, macOS, Windows, Linux (X11 and m17n) and Chrome OS. There is experimental support for generating *Common Language Data Repository* (CLDR) XML

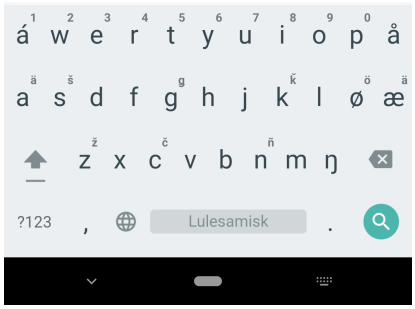

Figure 1: Screenshot of Lule Sámi keyboard for Android, as defined in the listing above.

files, *Scalable Vector Graphics* (SVG) files for fast layout debugging, and finite state transducers for neighbour key mistyping error models. The Windows installer includes a tool to register unknown languages, so that even languages never seen on a Windows computer will be properly registered, thus making it ready to support proofing tools and other language processing tools.

The mobile keyboard app also includes support for spellers (see later), to support the writing process.

Morphological analysers and dictionaries

Not counting Swahili and a few other languages, African languages have very limited corpus resources. Many African languages also have complex morphological structures. This holds most notably for Khoisan and (most) Bantu languages in the south and east as well as Afro-Asiatic languages in the northeast. Such languages, having complex morphology and little or no text resources, will be problematic for mainstream language technology.

Not much lnguage technology has been developed for these languages. The main exceptions are the works by Arvi Hurskainen on Swahili (for an overview and references, see Hurskainen (2018)) and the works by Laurette Pretorius, Sonja Bosch and others for Zulu, Xhosa and other South African languages (e.g. Pretorius & Bosch (2009*a*), Pretorius & Bosch (2009*b*)).

The GiellaLT infrastructure was developed to deal with this situation. The challenge of making language technology for complex and marginalised circumpolar languages was solved by representing the morphological structure as *finite state transducers*. Morphological analysers are the core of our language technology tools, they are written in form of *Finite State Morphology* Beesley & Karttunen (2003), Lindén et al. (2013). This means in practice that language modelling is based on dictionaries and hand-written rules for morphotactics as well as syntactic and semantic processing as needed, e.g. with *Constraint Grammar* Karlsson (1990), Didriksen (2010). While writing rule-based models for language processing is contemporarily written off as too slow and labour-intensive, in our experience full-time work on the dictionary and morphotactic rules for three months is enough to create high-coverage usable language models. If one compares this to the work it takes to create and curate corpora for machine learning, three months does not get one very far in the creation of gigaword corpora.

The main language families treated in the *GiellaLT* infrastructure are Uralic, Algonquin, Eskimo-Aleut and various Siberian languages. As for African languages, we have so far only experimented with eight of them[2]. The Somali language model is far beyond alpha level, it contains almost 16000 stems and the core morphology. The other language models are all relatively small, but in some cases, they still give an impression of how central morphophonological challenges may be solved.

Proofing tools

Proofing tools are a crucial piece of software to support normative language writing, i.e. spelling and grammar checkers and correctors. The morphological analysers described above are the basis for about every other tool one can build using the GiellaLT infrastructure, including proofing tools.

Spellers

A speller consists of essentially two parts: a morphological dictionary that contains the information as to whether a given word is part of the language or not. It is assumed by spell-checking that words not in the collected dictionary are misspellings of real words. The second part is an error model that takes the unknown word user inputted and tries to find similar words that are found in the language.

Technologies similar to the ones presented hare have been applied for African languages as well. One example is Bosch & Eiselen (2012), for Zulu, another is the Swahili speller distributed by the Finnish company Lingsoft.

In the GiellaLT infrastructure, both parts are modelled as finite state transducers (FSTs). The morphologically aware dictionary is built directly on the morphological analyser mentioned above after removing unwanted content such as punctuation and non-standard language. The error model is built as a *Levenshtein* edit distance model Levenshtein (1965) plus language-specific weighted replace rules Pirinen & Lindén (2010).

The speller package is distributed through our own desktop installation and updates system *Páhkat*, and a `pahkat` client is also part of the mobile keyboard apps. This means that spellers on all supported systems, both mobile and desktop, are automatically kept up to date.

As part of the mobile keyboard app, spellers help people in the writing process when using their native keyboard. We are currently experimenting with various forms of word completion and prediction models, but nothing has been released yet.

Grammar checking

The GiellaLT infrastructure includes an advanced grammar checker framework. It uses a combination of morphological analysers and Constraint Grammar Didriksen (2010) disambiguation and error detection components. Furthermore, the constraint grammar logic is used to determine where the grammar errors are; the logic is similar to grammar-based syntax parsing Wiechetek (2012). It is all rule-based, which means that it is possible to develop with essentially no pre-existing electronic corpora. The grammar checker features are quite new but are already used for four different languages.

Machine Translation

The GiellaLT infrastructure supports developing machine translation systems in cooperation with Apertium Khanna et al. (2021). The monolingual models developed in the GiellaLT infra are then combined with the transfer rules and lexicons in Apertium to provide an end-to-end MT system. The Apertium system at its core is also a rule-based machine translation toolkit. This means that one can build MT systems for languages with next to no existing resources in bilingual corpora as well, extending the work put in the monolingual dictionaries. The components needed for a rule-based machine translation on top of the rule-based morphological analysis in GiellaLT infra are a bilingual dictionary, i.e. a regular word-to-word translation dictionary and a set of grammatical rules concerning the translation of linguistic differences between languages.

Conclusion

We have presented the GiellaLT infrastructure and its philosophy and goal of supporting indigenous languages around the world, especially languages with complex morphology or phonology – or both. We have shown that a broad range of useful tools for language communities can be built with minimal preexisting electronic resources, thus allowing the creation of language technology tools for any language, and we have stressed the importance of open source as a strategy for avoiding losing language resources. Finally, we have given an overview of existing resources for African languages in the GiellaLT infrastructure.

The current status of African languages within our infrastructure is limited to several startups and experimental languages, one of the aims of this article is to further survey the potential for future coopera-

tion and engagement in the development of African NLP within our infrastructure.

Notes

[1] Hajdu Róbert 2021: Train BERT-Large in your own language. Towards Data Science.

[2] These are Akan, Amharic, Luo, Ndolo, Pedi, Somali, Tigrinya, and Zulu. The source code is available at *https://github.com/giellalt/lang-xxx*, where *xxx* should be replaced with the ISO 639-3 code of the language in question.

Acknowledgements

All work by the Divvun development group at UiT is funded by the Norwegian Ministry of Local Government and Regional Development. All work by the Giellatekno research group is funded by UiT The Arctic University of Norway.

References

Antonsen, L., Trosterud, T. & Wiechetek, L. (2010), Reusing grammatical resources for new languages, *in* 'Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)', European Language Resources Association (ELRA), Valletta, Malta.
URL: *http://www.lrec-conf.org/proceedings/lrec2010/pdf/254_Paper.pdf*

Beesley, K. R. & Karttunen, L. (2003), 'Finite-state morphology: Xerox tools and techniques', *CSLI, Stanford* .

Bosch, S. & Eiselen, R. (2012), 'The effectiveness of morphological rules for an isiZulu spelling checker', *South African Journal of African Languages* **25**, 25–36.

Didriksen, T. (2010), *Constraint Grammar Manual: 3rd version of the CG formalism variant*, GrammarSoft ApS, Denmark.
URL: *http://visl.sdu.dk/cg3/vislcg3.pdf (Accessed 2017-11-29)*

Hiovain-Asikainen, K. & Moshagen, S. (2022), Building open-source speech technology for low-resource minority languages with sámi as an example – tools, methods and experiments, *in* 'Proceedings of the the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages', European Language Resources Association, Marseille, France, pp. 169–175.
URL: *https://aclanthology.org/2022.sigul-1.22*

Hurskainen, A. (2018), Sustainable language technology for african languages, *in* A. Agwuele

& A. Bodomo, eds, 'The Routledge Handbook of African Linguistics', Routledge Handbooks, Routledge, Abingdon, pp. 359–375.
URL: *https://doi.org/10.4324/9781315392981-19*

Karlsson, F. (1990), Constraint grammar as a framework for parsing unrestricted text, *in* H. Karlgren, ed., 'Proceedings of the 13th International Conference of Computational Linguistics', Vol. 3, Helsinki, pp. 168–173.

Khanna, T., Washington, J. N., Tyers, F. M., Bayatlı, S., Swanson, D. G., Pirinen, T. A., Tang, I. & Alòs i Font, H. (2021), 'Recent advances in apertium, a free/open-source rule-based machine translation platform for low-resource languages', *Machine Translation* pp. 1–28.

Levenshtein, V. I. (1965), 'Двоичные коды с исправлением выпадений, вставок и замещений символов', Доклады Академий Наук СССР **163**(4), 845–848.

Lindén, K., Axelson, E., Drobac, S., Hardwick, S., Kuokkala, J., Niemi, J., Pirinen, T. A. & Silfverberg, M. (2013), Hfst—a system for creating nlp tools, *in* 'International workshop on systems and frameworks for computational morphology', Springer, pp. 53–71.

Pirinen, T. & Lindén, K. (2010), 'Finite-state spell-checking with weighted language and error models: Building and evaluating spell-checkers with wikipedia as corpus', *Proceedings of LREC 2010* .

Pretorius, L. & Bosch, S. (2009*a*), 'Computational aids for Zulu natural language processing', *Southern African Linguistics and Applied Language Studies* **21**, 267–282.

Pretorius, L. & Bosch, S. (2009*b*), Exploiting cross-linguistic similarities in Zulu and Xhosa computational morphology, *in* 'Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages', Association for Computational Linguistics, p. 96–103.

Treviso, M., Ji, T., Lee, J.-U., van Aken, B., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Martins, P. H., Martins, A. F. T., Milder, P., Raffel, C., Simpson, E., Slonim, N., Balasubramanian, N., Derczynski, L. & Schwartz, R. (n.d.).
URL: *https://arxiv.org/abs/2209.00099*

Wiechetek, L. (2012), Constraint Grammar based correction of grammatical errors for North Sámi, *in* G. D. Pauw, G.-M. de Schryver, M. Forcada, K. Sarasola, F. Tyers & P. Wagacha, eds, 'Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages (SALTMIL 8/AFLAT 2012)', European Language Resources Association (ELRA), Istanbul, Turkey, pp. 35–40.