Building language technology infrastructures to support a collaborative approach to language resource building

Tommi A Pirinen[1][0000−0003−1207−5395] and Francis M. Tyers[2][0000−0001−6108−2220]

[1] UiT Norgga árktalaš universitehta
firstname.lastname@uit.no
https://uit.no
[2] Indiana
ftyers@iu.edu

Abstract. Digital infrastructures are a vital part of support for providing a research framework and platform in engineering their digital lexicography and grammars and deploying the to end-users as real NLP software products. In this article we review the usage of two popular free/libre open source infrastructures and give our view on best current practices from few decades of experiences. We find that infrastructures can turn work in digital lexicography and grammars into viable end-user products like machine translators, spell checkers and correctors and so forth.

Keywords: NLP infrastructures · Citizen science · Uralic languages.

1 Introduction

Work in digital humanities, particularly like of digital lexicography, is a very front and central concept in building for language technologies and digital cultures for minority and under-resourced languages. In these contexts, it is common to have very limited access to language experts that are highly necessary for building of the resources necessary, we talk about *citizen science* and *crowd-sourcing*. For this reason, the role of digital infrastructures for building of the natural language processing systems becomes very central. An ideal digital infrastructure makes it easy for a language expert, activist or other contributor with limited technical skills or resources to work with involved technically complex systems, to contribute their expertise with their native language skills. In this article we make an overview of some infrastructures that enable contributing native language skills and describe some recent developments in the field of software engineering that have enabled us to improve our infrastructures so we also lay out a desiderata of a kinds for all sorts of language technology infrastructures.

In this article we use as a case study two infra-structures we have helped to build, that are also central to Uralic natural language processing: GiellaLT[3]

[3] https://github.com/giellalt

In Hämäläinen, M., Partanen, N., Alnajjar, K. (eds.) Multilingual Facilitation (2021), pages 53–60.

and Apertium[4], but we also cast an eyesight towards other kinds of popular infrastructures.

The article is organised in following sections: First we describe some background for the system in section 2. Then we describe and compare different infrastructures and their features in section 3. We discuss the system in section 4.

2 Background

The work on computational lexicography, specifically for finite-state language models and Uralic languages has been carried on for almost 40 years now. Some of the technologies and systems underneath remain largely unchanged, for example Finite-State morphology [2] methods have been used in similar form ever since the beginning. On the other end, the software engineering and computational infrastructures typically have life-cycle of not more than 10 years. The role of infrastructure to support language scientists and contributors is in many cases to reduce the impact of changes of such computational systems to provide smooth continuity for the development of the actual dictionary data etc. The scientists and language experts contributing words and grammars are usually not interested in differences of, e.g. RCS, CVS, SVN and git as version control system, while the infrastructures have been dragged through these all in the past decades—and conversely, the engineers who build the infrastructures may be very interested in such underlying technologies. The linguistic and NLP software is built on lexical data and grammars, this includes software like: *machine translation*, *spell-checking and correction*, *grammar checking*, *text-to-speech* and so forth. The main point of a good infrastructure is that the lexical data can be contributed once and used for all applications alike. In figure 1 there is a diagrammatic representation of ideal work-flow between various stakeholders of NLP software.

One of the core values that makes this all possible is based on the *free/libre open source software* (FLOSS) movement, which in way is turning into free science and open data movements in the academia. In the old times of single scientists keeping their word-lists and grammar sketches in the desk drawers, eventually to be published in a grand tome with a high price and restricting licence schemes, it was not easily possible for other researchers and the language community at large to start contributing and further developing the resources. However, the free and open source movement has already enabled such huge projects like Wikipedia, a massive crowd-sourced online encyclopedia made by humans, and this work is in a way trying to harness the same approaches in the field of linguistics and lexicography.

One of the recent innovations in software engineering is the concept of *Continuous Deployment* and *Continuous Integration* (CD/CI). This means that a software is developed all the time and the updates are made public immediately,

[4] https://github.com/Apertium

In Hämäläinen, M., Partanen, N., Alnajjar, K. (eds.) Multilingual Facilitation (2021), pages 53–60.

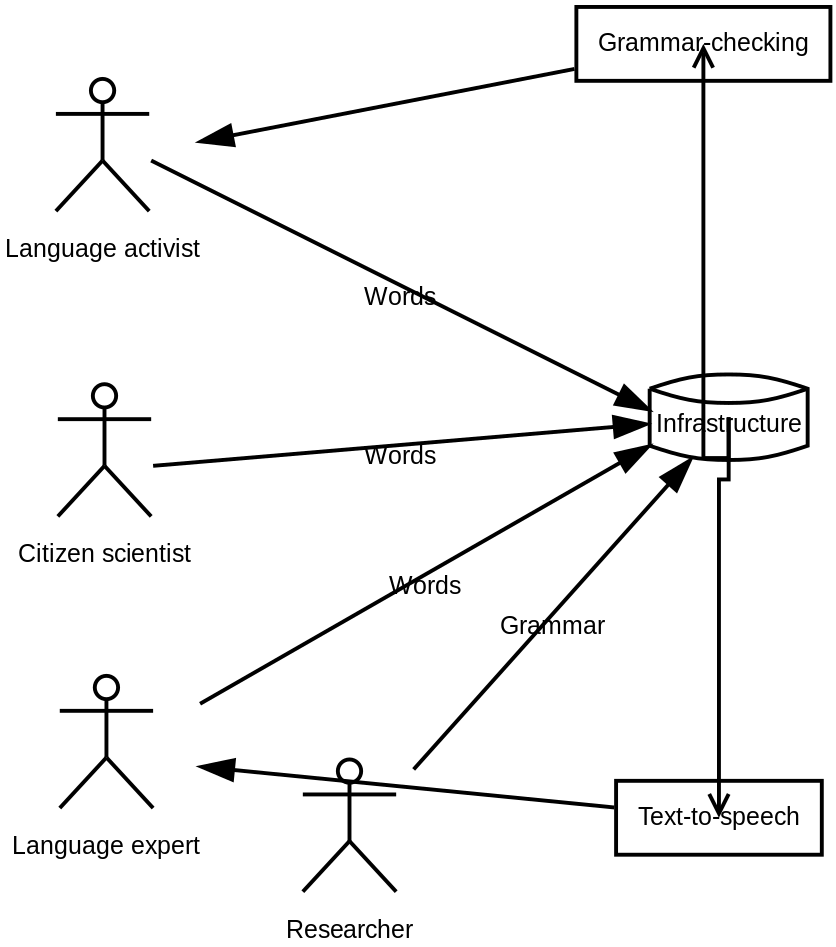

Fig. 1. Ideal workflow for infrastructure in NLP apps (simplified; in reality there is a fully connected graph of dozens of apps and stakeholders)

and there is an automatic infrastructure to test the software quality and send the end-users updated versions. This translates directly to work with language resources; when language users discover a word is not supported by the spell-checker or similar NLP software they can submit it to the infrastructure and get an updated version of the software with the word included within hours instead of after months of development cycle. In this article we study the implementation of such CI/CD infrastructures within the NLP systems.

2.1 Prior Art

The role of computational linguistics infrastructures in projects for cooperating with citizen scientists and language experts to create NLP software has not been very extensively studied. Some existing work has been published on the topic of co-operation of linguists and engineers in large scale infra-structure like projects, e.g. [8, 4, 7, 11]. In the part of involving the audience more in the infra-structure usage some recent projects have built on the existing and documented infrastructures, such as [1, 10] The other part of infrastructure usage and uptake is documented as outreach activities, for such relevant is also software projects, like UralicNLP [5][5], and activities of education networks like COPIUS[6] and labs like FU-Lab.[7]

3 Infrastructures and Resources

In this article we survey some of the main free/open source repositories for Uralic NLP specifically related to lexicography and morphology. The main goal of much of our work is to release NLP-based tools for end-users, be it machine translation or spell-checking and correction, and the lexicography and contributions linguistic research are achieved as a side effect. As one of the goals of the infrastructures is to lower the barrier to entry, we put an emphasis in this article to the infrastructure frameworks built to make it easier and more accessible. In table 1 we show an overview of the languages and resources for our infrastructures: GiellaLT and Apertium and for comparison other FLOSS infrastructures: CommonVoice[8] and Universal Dependencies[9] and UniMorph[10].

One important feature of a language technology infrastructure is separation of the technology and data. This means that the language experts can collect and curate data, while the engineers improve and add NLP systems, and when a new or improved system for a specific NLP application is finalised, it can be applied to all languages providing language data in the infrastructure. In practice for example, this has in past meant, that when new research was published making

[5] https://github.com/mikahama/UralicNLP/
[6] https://www.copius.eu/index.php
[7] https://fu-lab.ru/
[8] https://commonvoice.mozilla.org
[9] https://universaldependencies.org
[10] https://github.com/unimorph/

Table 1. Infrastructures and the languages and resources they have.

Infra - stuff	Languages	NLP resources
GiellaLT	121	Morphological analysis
		Text-To-Speech
		Spell and Grammar-checking
Apertium	181	Morphological analysis
	237	Machine translation
Common Voice	78	Speech recognition
		Speech synthesis
Universal Dependencies	104	Treebanking
		Dependency Grammar
Unimorph	122	Example
		word-forms

weighted finite-state spell-checking and correction end-user usable [9], all languages in the infrastructure could have an additional (albeit basic) spell-checker and corrector. Both in GiellaLT infra and Apertium system this is implemented at low level by simply applying the necessary changes to all of the language repositories. Due to potential of complicated interactions and change conflicts in this phase, both infrastructures have built additional tooling to ease the process: gut[11] and Apertium-init[12] In the core both systems are based on partial templating, with main difference being in the implementation technology: Rust and Python respectively.

3.1 Testing

One feature that an infrastructure is useful for as a supporting role for citizen science and crowdsourcing lexicography is quality assurance. In the NLP software this is done by automated testing. The automated tests for an end-user facing software can vary from as simple unit and integration testing as ensuring that you can enter word-forms and get analyses or translations back to as intricate as ensuring that the translation or spell-checking quality does not decrease. The testing methodology follows from software engineering and tends to use the same terminology: Unit testing, regression testing, integration testing and so forth. One of the testing approaches many linguists might be most acquainted with is a form of regression testing [3]. One of the recent developments in software engineering is automated testing of each commit on the server, using *continuous integration*. Continuous integration is set up to ensure that no changes to the lexicon are breaking the system or decreasing the results.

There are two problems that arise with complex NLP systems and automated testing: firstly, a full coverage testing on substantial material can take

[11] https://github.com/divvun/gut/
[12] https://github.com/Apertium/Apertium-init/

much more time and processing resources than is available on a freely available CI services. Secondly, complex NLP systems do not only break from changes within the local lexicographical data, but also from other parts of infrastructure changing. With the theme of rule-based machine translation this is obvious: each project depends at least from three different lexicographical projects: the *source* language dictionary, the *target* language dictionary and the *bilingual* dictionary, and change in any of these can throw the translation off. For other NLP pipelines, similar problems arise from changes in the build infrastructure, for example handling of capitalisation, tokenisation and even tagging standard changes. The first problem of limited processing power we have partially solved with self-hosted and customised build servers, however, this does introduce a non-standard non-trivial component to the mix so it is unoptimal for the purpose of this article (not easily reproducible). The second problem still exists in our build systems, however we are researching on potential solutions.

3.2 Deploying

One feature that infrastructure provides to scientists and language experts is turning their lexicons and grammars into end-user products fro the whole language community: for example languages in GiellaLT and Apertium repository are provided to users of LibreOffice, or Linux-based systems as packages for spell-checking and correction [9].[13]

The most common approach for deployment is, is to simply provide a standard build system, such as auto-tools based one, and have the distributors of systems pick it up for their users; this has been the traditional approach especially within the Linux ecosystem and other free / open systems. This is not always feasible option for minority language technology, e.g. due to lack of interest and necessary expertise on behalf of the distribution providers, for other approaches we have build custom repositories[14], and supplementary parts to packaging systems[15], which, while not ideal for average end users, are good enough to get the software to a majority of the end users.

3.3 Documentation

One of our goals in the original infrastructure designs has been to introduce the concept of *Literate Programming* [6] into NLP. The idea of producing documentation, even books, from well-commented source code is well within reach or natural language processing, lexicographical databases and grammar rules are typically the main meal of linguistic literature in general. For example in Fennistic tradition, it is not untypical to see a grammar book formatted in form of dozens and dozens of examples listed or tabulated accompanying every description of every grammatical phenomenon, this is very near to rule-based NLP

[13] https://divvun.org
[14] https://Apertium.projectjj.com/apt/
[15] http://pahkat.uit.no/

format already. The current implementation of the literate programming scheme in giellaLT infra follows the light-weight literate programming style most programming languages have eventually adopted as well, namely *doc comments*. What this means is we have specially formatted comment blocks within source code, that can be, together with context-relevant code, be re-used as documentation, as well as *unit tests*. Compare this to Python's docstrings[16] and doctests[17]. It is noteworthy, that mostly, in lexicography and grammar, tests, are nothing more than example word forms and their preferred annotations, be it grammatical analysis or perhaps spelling corrections, this, naturally, is in and of itself interesting for a linguist even without the testing feature. As a recent feature for the documentation comments, the output format has been updated to GitHub's github-pages format and a prototypical examples can be found at the time of writing in the GiellaLT github space[18].

4 Discussion and conclusions

We have built various infrastructures for enabling various groups of contributors to contribute linguistic data. Currently our infrastructures are used to involve large number of researchers in providing digital language tools for large number of marginalised languages.

In the table 1 we give the amount of languages being worked on within the infrastructures, while the numbers do not directly tell of the quality, there are further ways of determining how much the languages have been worked on in each of the systems. Within GiellaLT and Apertium communities we have opted for self-classifying the production quality of the languages as such: in GiellaLT registry[19] at the time of writing we find 8 full *production* quality and 27 *beta* testing phase languages (out of 128), in Apertium correspondingly 53 *trunk* translators and 39 *nursery* translators (out of 237).[20] The repositories Common Voice, Universal Dependencies and Unimorph on the other hand do not include classifications, but we can study them rather by comparing the sizes: the biggest recording size in English is 1800 hours, and 36 have 10 or more hours of confirmed recordings, out of the 78. For Universal Dependencies biggest data set is German with 3.753 million dependency trees, with 43 having over 100 thousand trees. In unimorph, Finnish is the biggest dataset with over 2 million word-forms, with 27 languages containing over 100 thousand word forms.

We have two infrastructures for NLP systems that have provided people with various tools and software for years. They have also improved continuously and still actively used.

[16] https://www.python.org/dev/peps/pep-0257/
[17] https://docs.python.org/3/library/doctest.html
[18] https://giellalt.github.io, see for example https://giellalt.github.io/lang-fin/fin.html
[19] https://github.com/divvun/registry#languages
[20] at the time of writing, this can obviously change fast in a fast-moving user community

Acknowledgments

Building and using infrastructures takes a large developer base and the authors of the article are just a small part of these communities. Thanks goes to everyone at GiellaLT[21], Divvun[22], and Apertium[23] communities.

References

1. Alnajjar, K., Hämäläinen, M., Rueter, J., Partanen, N.: Ve'rdd. narrowing the gap between paper dictionaries, low-resource NLP and community involvement. In: Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations. pp. 1–6. International Committee on Computational Linguistics (ICCL), Barcelona, Spain (Online) (Dec 2020). https://doi.org/10.18653/v1/2020.coling-demos.1, https://www.aclweb.org/anthology/2020.coling-demos.1
2. Beesley, K.R., Karttunen, L.: Finite-state morphology: Xerox tools and techniques. CSLI, Stanford (2003)
3. Bender, E.M., Poulson, L., Drellishak, S., Evans, C.: Validation and regression testing for a cross-linguistic grammar resource. In: Acl 2007 workshop on deep linguistic processing. pp. 136–143 (2007)
4. Bird, S., Liberman, M.: A formal framework for linguistic annotation. Speech communication **33**(1-2), 23–60 (2001)
5. Hämäläinen, M.: Uralicnlp: An nlp library for uralic languages. Journal of Open Source Software **4**(37), 1345 (2019). https://doi.org/10.21105/joss.01345, https://doi.org/10.21105/joss.01345
6. Knuth, D.E.: Literate programming. The Computer Journal **27**(2), 97–111 (1984)
7. Maxwell, M., David, A.: Joint grammar development by linguists and computer scientists. In: Workshop on NLP for Less Privileged Languages, Third International Joint Conference on Natural Language Processing. pp. 27–34. Hyderabad, India (2008)
8. Moshagen, S., Rueter, J., Pirinen, T., Trosterud, T., Tyers, F.M.: Open-source infrastructures for collaborative work on under-resourced languages. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC. pp. 71–77 (2014)
9. Pirinen, T., Lindén, K., et al.: State-of-the-art in weighted finite-state spell-checking. In: Computational Linguistics and Intelligent Text Processing 15th International Conference, CICLing 2014, Kathmandu, Nepal, April 6-12, 2014, Proceedings, Part II (2014)
10. Rueter, J., Hämäläinen, M.: Synchronized mediawiki based analyzer dictionary development. In: Proceedings of the Third Workshop on Computational Linguistics for Uralic Languages. pp. 1–7. Association for Computational Linguistics, St. Petersburg, Russia (Jan 2017). https://doi.org/10.18653/v1/W17-0601, https://www.aclweb.org/anthology/W17-0601
11. Streiter, O., Scannell, K., Stuflesser, M.: Implementing NLP projects for non-central languages: Instructions for funding bodies, strategies for developers. Machine Translation **20**(4), 267–289 (2006)

[21] https://github.com/orgs/giellalt/people
[22] https://github.com/orgs/divvun/people
[23] https://github.com/orgs/Apertium/people