Proceedings of the EURALI Workshop @LREC2020, pages 36–41
Marseille, 20 June 2022

Bringing Together Version Control and Quality Assurance of Language Data with LAMA

Aleksandr Riaposov, Elena Lazarenko, Timm Lehmberg
Universität Hamburg
Hamburg, Germany
{aleksandr.riaposov, elena.lazarenko, timm.lehmberg}@uni-hamburg.de

Abstract

This contribution reports on work in process on project specific software and digital infrastructure components used along with corpus curation workflows in the the framework of the long-term language documentation project INEL. By bringing together scientists with different levels of technical affinity in a highly interdisciplinary working environment, the project is confronted with numerous workflow related issues. Many of them result from collaborative (remote-)work on digital corpora, which, among other things, include annotation, glossing but also quality- and consistency control. In this context several steps were taken to bridge the gap between usability and the requirements of complex data curation workflows. Components of the latter such as a versioning system and semi-automated data validators on one side meet the user demands for the simplicity and minimalism on the other side. Embodying a simple shell script in an interactive graphic user interface, we augment the efficacy of the data versioning and the integration of Java-based quality control and validation tools.

Keywords: corpus curation, quality assurance, workflow management

1. Introduction

Having started in 2016, the long-term project INEL (Grammar, Corpora, Language Technology for Indigenous Northern Eurasian Languages)[1], spanning 18 years, aims at a broad and comprehensive empirical analysis of language data coming from endangered languages and varieties of the Northern Eurasian Area[2]. For this purpose it generates deeply annotated digital language resources (language corpora and further accompanying resources) from existing as well as newly acquired language material. As an integral part of the project these resources are made long-term available after their finalization and also become part of on-the-fly analysis already during the process of their curation. This unique property enriches the project research by adding a dynamic momentum to the empirical work but also puts high demands on the digital workflows and tools being used along with the corpus creation.

In this paper we will first outline the corpus curation workflows that have been established in the initial six years of the project run-time by focusing on the establishment of a versioning and semi-automated quality checks. Based on this we will present the latest development steps that aim at a more user-friendly and seamless integration of both aspects into linguists everyday work.

[1]https://www.slm.uni-hamburg.de/inel.html

[2]The project is funded by the German Federal Government and Federal States in the Academies' Programme, with funding from the Federal Ministry of Education and Research and the Free and Hanseatic City of Hamburg. The Academies' Program is coordinated by the Union of the German Academies of Sciences and Humanities.

2. Preliminary work

Up to the present day corpora in Selkup (Brykina et al., 2021), Dolgan (Däbritz et al., 2019), Kamas (Gusev et al., 2019), and Evenki (Däbritz and Gusev, 2021) languages have been finalized following strict quality and consitency criteria and published under open access conditions by the language specific sub-projects. All resources as well as multiple graphic user interfaces are available via the INEL-Portal[3]. Furthermore a comprehensive description of the project structure can be also found at (Arkhipov and Däbritz, 2018). One of the most important contributions that made this outcome possible was the adaption of *continuous integration* principles from software development projects to linguistic data curation workflows as described by (Hedeland and Ferger, 2020). Put simply, this includes the establishment of workflows and technologies, that allow for a continuous manipulation of content (in this case language data instead of programming code) whereas the data itself is kept in a state that it can be used as an empirical base for language analysis during the entire process of its creation. In the following one important core componemt of these these workflows, a versioning system, is introduced.

2.1. Version control systems

In recent years the use of version control systems (VCS) such as *Git*[4] or *Apache Subversion*[5] has established as a popular way for collaborative data management and exchange. VCS help to track down changes in the data one is working with and make snapshots

[3]https://inel.corpora.uni-hamburg.de/portal/

[4]https://git-scm.com/

[5]https://subversion.apache.org/

of those changes over time; therefore, they are highly beneficial for collaborative work. Originally stemming from software development, principles of versioning have the potential for improving the quality of collaborative work on textual content like in the case of language data curation. As mentioned above, INEL places a high value on the quality and consistency of the corpus data and thus, on data curation and control. In this context versioning with the means of Git has been considered vital for the project workflows. However, this decision naturally leads to a question of how to seamlessly integrate usage of such a highly technical tool into the linguistic research routine. To do so, following aspects should be overseen:

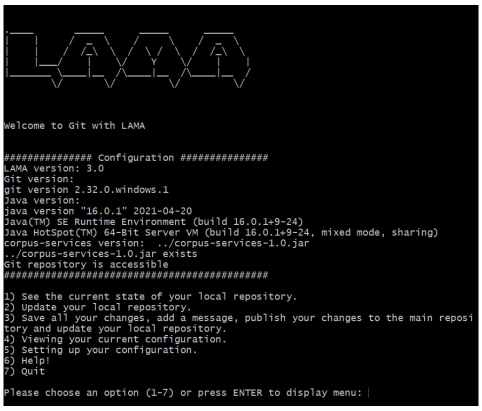
- **Functionality:** Because Git is primarily a tool intended for software development, it encompasses by far more functions than it would be needed to synchronize the process of linguists' work. Moreover, Git commands have a tendency to transcend into being "cryptic" and hard to remember. Thus, a technically inexperienced person might find it hard to use them fully intuitively without diving into the theoretical background behind them and surrounding oneself with cheat-sheets and references. Furthermore, there is a high risk of data damage and even loss if a certain command is misused.

- **Interface.** There are two main ways to interact with Git - by the means of a command-line interface (CLI) or a graphic user interface (GUI). Following on from the functionality aspect, out-of-the-box Git GUI as well as further popular GUI solutions may seem over-saturated with various options and functions. This can lead to a more complex learning curve during the first encounters with VCSs and result into reluctance to use them. On the other hand, using Git CLI may seem to be confusing for those who have little to no previous experience with command-line shells. The confusion provoked by an unfamiliar working environment (merely a terminal window with which non tech-savvy rarely work) combined with the requirement to type in the terminal various commands, in turn, could lead to even more reluctance to learn Git CLI than the Git GUI.

Consequently, while Git could potentially bring quality control workflows on a new level, its full functionality can simultaneously become a reasonable disadvantage when users with a non-technical background are challenged to use it. To overcome this obstacle, it has been decided to develop a Git tool that fulfills several project-specific criteria. First of all, it should operate only with basic Git commands determined as necessary for the linguists' collaborative work and be easy-to-use. This would provide linguists immediate access to the tool and spare time on learning esoteric Git functionality. Secondly, its looks should be balanced and concentrated only on what is really important for the current work: not as "frightening" as the standard CLI and at the same time, not over-saturated with various buttons and working tree visualizations that do not benefit the linguist and could be potentially distracting. Finally, the tool should have a potential to be integrated into existing workflows and complement them.

2.2. LAMA - Linguistic Automation Management Assistant

To satisfy the above listed criteria, a minimalistic Git client LAMA (Linguistic Automation Management Assistant) has been developed in previous years (Ferger and Jettka, 2021a; Ferger and Jettka, 2021b). Although still being a Bash script running in a terminal window, at that point LAMA offered a simple user interface. It was no longer crucial to keep in mind all the necessary Git commands and type them manually because of a simple user menu. Moreover, since LAMA was designed to run in a Unix shell, it did not depend on further software other than Git itself. The tool was a cross-platform and ready-to-use solution, and an immediate and straightforward introduction of LAMA into the research workflows was possible.



```
Welcome to Git with LAMA


############### Configuration ###############
LAMA version: 3.0
Git version:
git version 2.32.0.windows.1
Java version:
java version "16.0.1" 2021-04-20
Java(TM) SE Runtime Environment (build 16.0.1+9-24)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.1+9-24, mixed mode, sharing)
corpus-services version:  ../corpus-services-1.0.jar
../corpus-services-1.0.jar exists
Git repository is accessible
#############################################

1) See the current state of your local repository.
2) Update your local repository.
3) Save all your changes, add a message, publish your changes to the main reposi
tory and update your local repository.
4) Viewing your current configuration.
5) Setting up your configuration.
6) Help!
7) Quit

Please choose an option (1-7) or press ENTER to display menu:
```

Figure 1: Previous LAMA version.

Unlike other Git clients that are aimed at broader user communities, LAMA's functionality was concentrated explicitly on what linguists need when working within INEL. This includes such basic functions as checking the status of the repository, pulling changes from the remote, staging and committing local changes and pushing them to the remote. At first, the restricted functionality of LAMA may seem rigid and and not sufficient. However, it corresponds with the established workflows and strictly follows the principles of linguistic data curation in INEL. For example, since all the linguistic data curation is done on a single Git branch, LAMA does not support branch functionality, which prevents linguists from accidental checkout of their data onto a separate branch and allows for transparent tracking of the work process. For similar reasons, various complex Git operations are not supported as well. The script is designed in such a way that one does not have to manually type any commands, but choose an option from the text-based menu. Moreover, LAMA could be easily integrated into other software used in the project:

- Messenger platforms (currently integrated within Mattermost[6] and Microsoft Teams) via API calls to webhooks in order to automatically report about errors and their origin;
- The Corpus Services framework (see section 2.3) for further quality-assurance. Earlier versions of LAMA were already able to carry out local pretty-printing of XML files before publishing the changes, moreover, there was potential to integrate other, more complex, Corpus-Services functions that was tackled at latter stages of LAMA development.

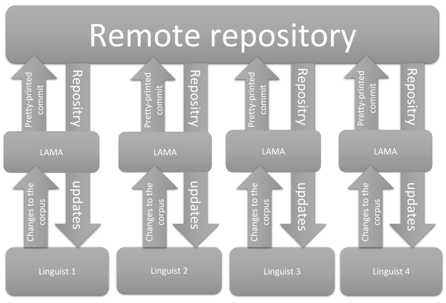


Figure 2: General LAMA workflow

Compared to the use of standardized Git clients, be it CLI or GUI, initial setup and launch of LAMA was project-specific as well. To begin with, users had to clone a corpus locally without using LAMA itself, but a default Git solution instead; afterwards they had to place the LAMA script in the respective local file structure (the script can operate only within a working directory). Another setup challenge that linguists were facing was creating a proper folder hierarchy within their working directories. This required a download (and regular updates) of Corpus Services JAR and placing it always one folder above the corpus clone to ensure the functionality of the Corpus Services pretty-printing function (see 2.3). Upon the first LAMA launch it was required to provide user credentials and afterwards the LAMA setup was ready to use, however, certain steps like repository cloning could have been repeated multiple times if work with several corpora was done simultaneously. Although initially such a setup was not considered a obstacle as long as LAMA nonetheless boosted up the performance and improved interaction with Git, later we could notice that this was a considerable shortcoming of the CLI version of LAMA. We will address this in the following sections of this paper.

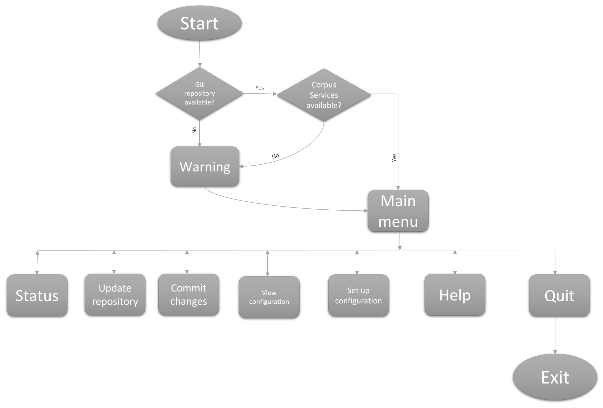


Figure 3: Algorithm of the previous LAMA version

2.3. Quality Checks

A significant part of the data curation is covered by the *Corpus Services* framework, a set of customized Java-based data validators initially developed at *Hamburg Center for Language Corpora (HZSK)*[7] as the result of longtime curation work on spoken language corpora (Ferger et al., 2020) using the EXMARaLDA System[8] (Schmidt and K., 2014) and its XML-based data formats. In the following the framework has been utilized and further developed by several projects; whereas in the infrastructure initiatives CLARIAH-DE[9], CLARIN-D[10] and the project QUEST[11] rather generic application scenarios were in the centre, development work in INEL followed project-specific tasks. The result of this effort (see https://gitlab.rrz.uni-hamburg.de/corpus-services/corpus-services) can be grouped as follows:

- Whenever a linguist working on data pushes local changes to on of the projects Git repository via LAMA, the data is automatically being pretty printed to assure unified formatting between different local copies (see Figure 2).
- Being a part of the automated INEL workflows (see Figure 4), every night a battery of scripted checks is performed on each corpus via cronjob; as a result, an updated report containing errors to be fixed and warnings to be watched out for is created. About a third of the checks in the battery are relatively simple cleanup and replacement tasks, and we run those in fully automated fixing mode,

[6]https://mattermost.com/

[7]https://corpora.uni-hamburg.de/

[8]https://exmaralda.org

[9]https://www.clariah.de

[10]https://www.clarin-d.net

[11]https://www.slm.uni-hamburg.de/ifuu/forschung/forschungsprojekte/quest.html

meaning that such a check would not only spot an error in the data, but also repair it on the fly.

- Some other Corpus Services functions are meant to be run sparsely, and are used as a part of our corpus publication workflow, e.g. a converter of EXMARaLDA EXS files to the *ISO/TEI standard "Transcription of Spoken Language"* (ISO, 2016)[12].

- From time to time a need to perform some manual task arises (e.g., conversion of corpus data between different formats such as FLEX[13], ELAN(Sloetjes and Wittenburg, 2008) and EXMARaLDA, or replacement of a certain grammatical or lexical gloss in the whole corpus).

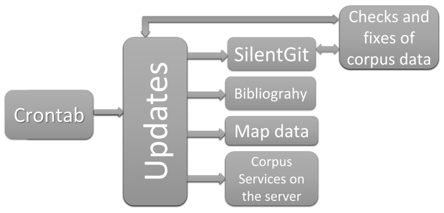

Crontab
Updates
SilentGit
Checks and fixes of corpus data
Bibliograhy
Map data
Corpus Services on the server

Figure 4: Infrastructure of the automated workflows

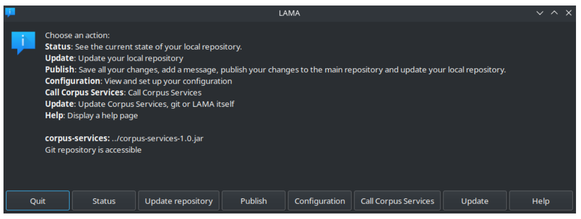
As mentioned above, although LAMA and Corpus Services until now have been contributing to the same workflows, they have not yet been integrated into one another and the coordination of the use of both components had to be coordinated separately by the technical staff of the project. Therefore, a potential of bringing the two tools closer to each other was considered beneficial for the further improvement of the project road map.

3. Putting workflows to a new level

A simple, lightweight, cross-platform Bash script allowing non-technical users to enjoy the basic functionality of Git without delving deep inside the intricacies of version control systems was a welcome addition to the project's workflow, however, after considering the feedback from the users, it became clear that some particularities of its implementation left us room for considerable improvement. First, we underestimated how cautious some people are about working with CLI as a tool, as it subjectively feels dangerous, looks impenetrable and thus falls outside their comfortable limits of control. Second, we ran into issues caused by the output produced in the command line in case something went wrong: instead of simply pointing out to the user what the problem was, the script returned what essentially was a log with the lifespan of a CLI window; unfortunately, in some cases it managed to confuse less tech-savvy users possessing pauce log-reading competence, which in turn resulted in avoidable data loss, exacerbated by the fact that the log was not actually stored anywhere. Third, the initial version was hindered by its rigid "choose a stock option" design, which effectively prohibited the user from interacting with the script in a more complex way – e.g., it was not quite up to the task of automatizing common quality control scenarios. Here is how we addressed each of the development challenges specified above:

3.1. The challenge of usability: providing a GUI

An intuitive clickable GUI alleviates many worries about the use of command line. Having in mind that we would like to keep our GUI simple to use and easy to develop while staying cross-platform, we surveyed some options as to the implementation of said interface such as the popular framework *Electron*[14] or *Swing Application Framework* in Java, but eventually settled on *Zenity*[15], a toolkit built-in in many current Linux distributions that creates GTK dialogue boxes for shell scripts. While not sophisticated enough to provide a list of features common in fully-fledged applications, it satisfies our simplicity criterion and does the job nonetheless.



Choose an action:
Status: See the current state of your local repository.
Update: Update your local repository
Publish: Save all your changes, add a message, publish your changes to the main repository and update your local repository.
Configuration: View and set up your configuration
Call Corpus Services: Call Corpus Services
Update: Update Corpus Services, git or LAMA itself
Help: Display a help page

corpus-services: ../corpus-services-1.0.jar
Git repository is accessible

Figure 5: The main dialogue box.

Another usability goal was to eliminate the needlessly abstruse installation procedure one had to go through before launching the script for the first time. Among the requirements were the need to have a working clone of one of our corpora and an up-to-date JAR package file containing Corpus Services; then on the first launch the user had to remember to provide their credentials (in the INEL case - GitLab) without a script prompt to do so. Seeing how the users - quite rightfully - struggled to complete the installation on their own, we decided to automatize the process where possible, striving to provide a solution that would work "out of the box". Since Zenity, not available natively on Windows and MacOS, is now required to run the script, a step where Zenity is downloaded and installed was added as well. The result is that the matured script walks the user through its installation, and is able to clone

[12]http://www.iso.org/iso/cataloguedetail.htm?csnumber=37338
[13]https://software.sil.org/fieldworks/
[14]https://www.electronjs.org/
[15]https://help.gnome.org/users/zenity/stable/

a repository or fetch the JAR file on its own, thus making this once-esoteric procedure, which often required guidance, as smooth as possible. In addition to that, LAMA acquired the ability to update its core dependencies, Git and Corpus Services, as well as LAMA itself, within the script. Our users were happy to see that they gained an ability perform the operations described above by themselves, without having to ask a member of the technical team.

3.2. The challenge of output: logging

Since LAMA logging was not ideal in the previous version, our users ran into issues, e.g. they would mishandle a Git merge conflict, snowballing it into something pernicious and data-damaging. Having that in mind, we modified the way LAMA keeps track of its past activities: in case of an error the user will be presented with a concise message saying what the error was about, while the complete log is now being collected in the background. That simple feature allowed us to deal with problems more efficiently.

3.3. The challenge of quality control: Corpus Services implementation

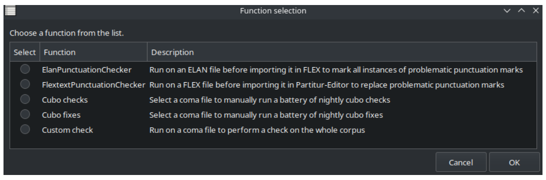
In older versions of LAMA, only the first part of our workflow delineated in 2.3 was implemented in the script; anyone wishing to run Corpus Services for other tasks had no options but to call it from the command line using specific syntax which is not immediately clear to a linguist. That in turn severely hindered wider-scale propagation of Corpus Services as a quality control tool in non-INEL environments. In other words, we needed a user-friendly interface for it. A web interface for Corpus Services is in the works, but it has some downsides rendering it unfit to use with our corpora, namely a) it is not possible to run checks in fixing mode via the web interface; and b) there is an upper limit of 2 GB on the amount of data one can upload to be checked there, and the INEL corpora sit well outside that limit (e.g., the INEL Selkup Corpus has nearly 12 GB of files). However, Zenity-powered LAMA proved to be the solution we were looking for in the first place, as the script has already had basic Corpus Services support with none of the web interface downsides, and simply needed a GUI to let a user navigate quality control tasks gracefully.

In order to facilitate the use of the interface, we identified three large groups of tasks one might want to perform there:

- Common tasks, which include pretty printing and the ability to manually run automated checks and fixes specified in our workflow above;

- Simple tasks, or Corpus Services functions which do not require any parameter;

- Complex tasks, on the other hand, require at least one parameter such as a tier name.



Choose a function from the list.

Select	Function	Description
○	ElanPunctuationChecker	Run on an ELAN file before importing it in FLEX to mark all instances of problematic punctuation marks
○	FlextextPunctuationChecker	Run on a FLEX file before importing it in Partitur-Editor to replace problematic punctuation marks
○	Cubo checks	Select a coma file to manually run a battery of nightly cubo checks
○	Cubo fixes	Select a coma file to manually run a battery of nightly cubo fixes
○	Custom check	Run on a coma file to perform a check on the whole corpus

Figure 6: Quality control functions.

To run simple and complex tasks the user is asked to select a desired Corpus Services function and parameters if necessary. Common tasks are meant to be run much more often than the other tasks - hence the name; each such task comes with a description of what it does, and has some of the parameters filled in by default. The list of common tasks is curated by our technical team based on feedback received from linguists.

All in all, a GUI solution, however simple, helped us reduce overhead efforts incurred by substantial impregnability of the previous version of our toolset, thus bridging the gap between the tools we develop and the people actively using them: hence the positive feedback received from our users, who are now able to perform quality control operations on their own. Automatizing existing quality control and data manipulation procedures in the GUI allows us to focus our future development on new, as-yet-untrodden ways to improve and enrich the quality of linguistic data produced in the project.

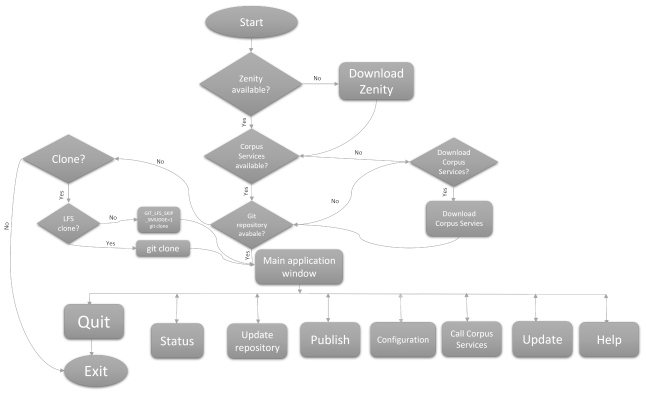


Figure 7: Algorithm of the current LAMA version

4. Conclusion and future work

In our paper we outlined technical workflows and presented internal tools used in the INEL project to process transcribed spoken language data, described the challenges we have met, and the solutions we have implemented to achieve and maintain continuous quality control of language resources being created in the project.

When devising best practices for language data-oriented tools, intended both for in-house use and the wider scientific community, it is prudent to know what the users need and build the tools according to their needs and technical competences. An unnecessarily difficult to use tool, no matter how good, is unfortunately doomed to be rejected by some prospective users simply because the learning curve was too steep. Implementing GUI is a surefire step to flatten the curve, thus widely improving usability of the tool for many a user. We stand however by the “easy to use, easy to develop” motto for simple tools such as LAMA, where allocating extra resources to make it into a full-fledged application would not increase its usability and functionality proportionally to the efforts required.

Corpus Services finally getting a GUI, along with a web-based interface underway, is a welcome development that we hope would further increase its applicability beyond the INEL context. That being said, wider rates of community uptake require Corpus Services to evolve from a project-oriented quality control tool to something wider as well; absorbing both the availability to conform to commonly used “generic” standards of data quality as well as looking for implementations for other kinds of language data, i.e. audiovisual corpora. The development is continuously ongoing and both INEL and outside researchers are always welcome to test new tools functionality as soon as it becomes available. However, an extensive UX study on how newer LAMA versions improve the workflows is still required, after which we will be able to support our results with quantitative data.

5. Bibliographical References

Arkhipov, A. V. and Däbritz, C. L. (2018). Hamburg corpora for indigenous northern eurasian languages. *Tomsk Journal of Linguistics and Anthropology*, 21(3):9–18.

Ferger, A. and Jettka, D. (2021a). Fun with VCS - more with less: A Tool for Facilitating the Use of Git in Linguistic Research Data Management. Zenodo, September.

Ferger, A. and Jettka, D. (2021b). LAMA - your friendly and easy git script.

Ferger, A., Hedeland, H., Jettka, D., and Pirinen, T. (2020). Corpus services.

Hedeland, H. and Ferger, A. (2020). Towards continuous quality control for spoken language corpora. *Quality Control for Spoken Language Corpora.*, 15(1).

ISO. (2016). Language resource management — transcription of spoken language. Standard, International Organization for Standardization, Geneva, CH, August.

Schmidt, T. and K., W. (2014). Exmaralda. In Ulrike Gut Jacques Durand et al., editors, *Handbook on Corpus Phonology*, pages 402–419. Oxford University Press.

Sloetjes, H. and Wittenburg, P. (2008). Annotation by category-elan and iso dcr. In *6th international Conference on Language Resources and Evaluation (LREC 2008)*. Max Planck Institute for Psycholinguistics, The Language Archive, Nijmegen, The Netherlands.

6. Language Resource References

Brykina, Maria and Orlova, Svetlana and Wagner-Nagy, Beáta. (2021). *INEL Selkup Corpus (Version 2.0).*

Däbritz, Chris Lasse and Gusev, Valentin. (2021). *INEL Evenki Corpus (Version 1.0).*

Däbritz, Chris Lasse and Kudryakova, Nina and Stapert, Eugénie. (2019). *INEL Dolgan Corpus (Version 1.0).*

Gusev, Valentin and Klooster, Tiina and Wagner-Nagy, Beáta. (2019). *INEL Kamas Corpus (Version 1.0).*