Mei-friend. A viewer and last-mile editor for MEI score encodings

mei-friend, Werner Goebl, David M. Weigl (trompamusic) (ed.), 2021. https://github.com/trompamusic/mei-friend (Last Accessed: 06.02.2022). Reviewed by Oleksii Sapov (Mozarteum Foundation Salzburg), sapov@mozarteum.at.

Abstract

Mei-friend, a free and open-source package for the Atom text editor, enables convenient and fast rendering of music encoded in the format of the Music Encoding Initiative (MEI) including some important navigation and display options. Furthermore, it provides various functionalities for graphical editing of MEI code using rendered music notation. Mei-friend is primarily aimed at musicologists, librarians, musicians and music enthusiasts. Users are required to have a moderate technical expertise only owing to straightforward installation and intuitive usage of the tool.

Introduction

1   Digital music editions can be created using the format of the Music Encoding Initiative (MEI), which "share[s] many common characteristics and development practices"[1] with TEI, especially those for digital editions[2]. For MEI, rendering of XML code as symbolic music notation is essential during the edition process. Typically, projects using MEI[3] create their own (web) applications where MEI is rendered. Those are mainly designed as presentation platforms and might not be very suitable as production tools, although some project-specific functionalities can be tested in these

environments. The tool mei-friend is designed for usage *during* the process of creating an edition, more precisely for preparing basic musical text only as no editorial-specific markup is currently possible. Music notation other than Common Western Notation might be displayed if supported by the rendering engine[4]. However, no editing functionalities for working with them are available.

2   Mei-friend is distributed as a free package[5] for the Atom text editor[6]. It has been developed by Werner Goebl and David Weigl in order to meet the needs of the project TROMPA[7] and was generously shared with the MEI community[8]. The author was able to integrate the tool into his daily routine of MEI encoding very well, turning it into a more convenient and efficient process[9].

3   For this review, the following hardware/software was used: Microsoft Windows 10 Pro N, Version 21H2 (64 bit), Atom v1.58.0 x64, mei-friend v0.6.0, running on Dell Latitude 7480 (CPU Intel Core i5-6300U, RAM 8 GB). Additionally, the installation process and basic functionalities were checked on Ubuntu 21.04 with the same Atom and package versions, running on HP ProBook 440 G4 (CPU Intel Core i5-7200U, RAM 8 GB).

Methodology

4   Since MEI was introduced, a number of tools[10] have been developed which try to address the problem of MEI rendering in the first place as currently no notation software like Sibelius, Finale or MuseScore supports MEI natively. A few of the earliest initiatives include Verovio MEI viewer[11] and MEISE[12] (web-based), which can display and navigate MEI. An oXygen add-on[13] enables rendering of MEI right in the editor, but it currently seems to work for macOS only. Further on, the Verovio-Humdrum-Viewer[14] (VHV) and Verovio Editor[15] (web-based) offer functionalities for graphical editing of MEI. The former offers a relatively large amount of editing operations; however, internally VHV works with the format *kern.* This leads to some conversion inconsistencies when importing/exporting MEI files. The Verovio Editor processes MEI natively but only a few editing operations are currently implemented. In addition, some of the tools mentioned above offer other functionalities such as creating PDF and MIDI files, advanced rendering options, etc. The predecessor of mei-friend is a (legacy) Atom package – mei-tools-atom[16]. Although, this package offers hardly any functionalities beyond a basic display, it certainly provides a good base for further development.

5   None of the tools mentioned above (including mei-friend) cannot be considered as a full-fledged editor. Therefore, a typical data acquisition method in the MEI community includes conversion from data formats originally gained by manual typesetting with a notation software mentioned above or using Optical Music Recognition. Usually, manual post-processing is needed, and this is the gap that mei-friend aims to fill.

6   One of the advantages that mei-friend offers is the user friendly rendering. In my other workflow, I use a local editor for code editing and a web-based application for rendering. This setup requires several steps to display the results. This is time and energy consuming when performed repetitively and might cause errors; for instance, because of file cashing in the browser. Mei-friend updates the rendered notation immediately after the code has been changed. This may happen as one types in Atom (default) or after saving the file in an external editor. An additional advantage is that the page remains the same after re-rendering (in some other web applications I used, the displayed page flips to the first page of the score).
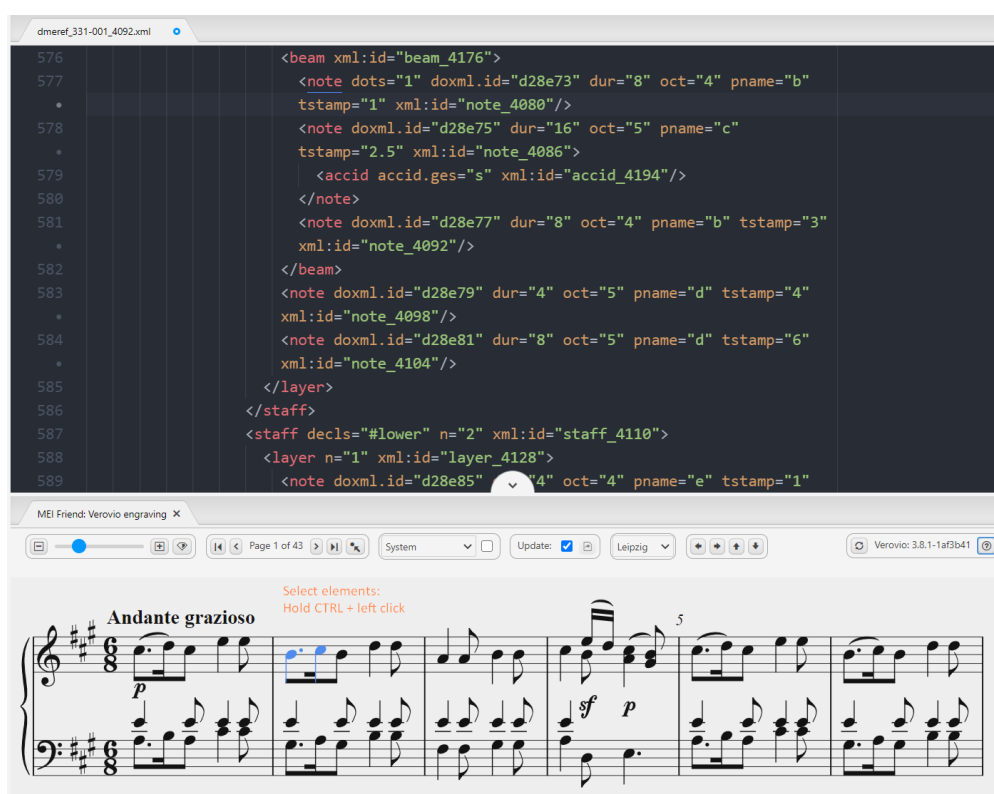

Fig. 1: Selecting multiple elements.

7   As mentioned above, the conversion to MEI from other formats is currently not lossless and a broad variety of issues can occur. Some of them can be fixed by using the editing functionalities of mei-friend. Especially useful are those for inserting the

ControlEvents[17] (e. g. `<dynam>`, `<slur>`, `<arpeggio>`) as these elements establish references to Events (`<note>`,`<chord>`) by pointing to their `@xml:ids`. Collecting these `@xml:ids` manually can be a tedious and error-prone task, especially if you work with a large score like a symphony. The approach used by mei-friend is intuitive and fast: Select the Events by clicking them in the notation window (cf. figure 1) and press a shortcut that triggers a code editing function. For instance, the shortcut 's' inserts the following MEI element into its correct position in the XML hierarchy:

```
<slur xml:id="slur-0000000025523666" startid="#some-note-id1" endid="#some-note- id2"/>
```

Code 1: An inserted <slur> after applying the shortcut 's'.

8 Currently, mei-friend enables inserting a subset of ControlEvents and the layer-level-element <beam>; respectively, deletion works for these elements only. The elements are inserted with a pre-defined (minimal) set of attributes and dynamically created `@xml:ids`. Placement attributes can be manipulated, but other attributes have to be entered manually.

9 Finally, mei-friend provides a few utilities for code refactoring operations as renumbering of measures, correction of accidentals and the utility called "regenerate encoding through Verovio". The latter adds `@xml:ids` where missing but should be applied with special care as some undesired code changes may occur (cf. the issue on GitHub).

Implementation

10 "[M]ei-friend is written in JavaScript using the Atom package framework" (Goebl, Weigl 2021, 6). It has very little dependencies which are managed using the node package manager: jQuery and Verovio. In addition, the Atom package language-mei is required. Using Verovio's JavaScript toolkit, MEI is rendered to SVG and embedded into a "repositionable pane within the Atom application window" (Goebl, Weigl 2021, 6) as it would be in a browser. Likewise, some other functionalities such as page breaks, zoom etc. are based on the respective Verovio functions.

11 The tool accepts files with 'xml' and 'mei' extensions and UTF-8 encoding. No validation is performed natively, but dedicated XML packages may be used. Some

encoding-related warnings and errors raised by Verovio and actions on the GUI are tracked in the console of the developer tools[18]. For erroneous data, depending on the error, rendered notation is displayed up to the point where the error occurs or is not displayed at all.

12   The performance on small[19] and middle-sized[20] files is high using the default settings. However, "[l]arge[21] MEI encodings (of about 25,000 lines and more) cause performance issues[…]"[22] including disabled XML-syntax highlighting. Two possibilities to handle this are provided: 1) Manual updates of the rendered notation and 2) usage of the speed mode. The last is an innovation which significantly improves performance on large files[23]. Currently, a few limitations[24] and downsides[25] have to be expected.

Usability, sustainability and maintainability

13   The package can be installed using GUI or from CLI[26]. The dependencies are handled by the installation process itself; a prompt appears for installation of the Atom package language-mei. The installation procedure is well described in the Atom package registry[27], which I consider to be a main documentation page.

14   The documentation provides an overview of the tool and its functionalities; an animated screenshot[28] illustrates the concepts described. Example MEI files are provided as direct download and as a link to a GitHub repository with musical works published by TROMPA. Additionally, a few MEI files can be found locally in the installation folder of the package.

15   No special tool support is provided, but issues can be posted on GitHub and are reviewed in a reasonable time. In addition, the developers are active members of the MEI community and can be reached through its common communication channels[29].

16   The learning curve based on my experience and those of our project's chief lecturer is rather moderate. No specialized programming skills are required. However, it is advantageous to have basic knowledge of how to work with text editors, in particular with Atom, and, of course with MEI itself. It may take some time to familiarize yourself with the keybindings and to memorize them.

17   Mei-friend can be used on all platforms[30] where Atom can be installed. However, a small bias towards macOS is present: The icons for the keybindings are macOS-style also on Windows and Ubuntu, and not every keybinding works on these platforms[31].

18   As mei-friend is an Atom package, the user benefits from other Atom functionalities and features such as a built-in search utility, Git and GitHub integration, other related packages, etc. In addition, Atom interoperates well with other editors. For instance, it is possible to edit the same MEI in oXygen XML editor and view the changes in the rendering pane of mei-friend immediately after the file has been saved.

19   Package development takes place on a public GitHub repository[32]. A README file contains documentation (same content as on the Atom package page) as well as acknowledgments to the project contributors and predecessors; the (MIT) license is linked here. No CITATION.cff[33] file or citation guidelines are provided.

20   The developers of mei-friend informed me that the tool is currently being redeveloped and extended as a web-browser application in the context of the project Signature Sound Vienna (development takes place on another GitHub repository which is currently private). Several general improvements are planned: automatic page flipping, a wider range of rendering options (based on Verovio), GitHub integration, etc. In addition, functionalities that are specific for digital editions (such as editorial markup) might be added.

21   Most likely the development of a web-based version of mei-friend will be priority and the Atom package will not be actively maintained. For users who still prefer to use the Atom package (due to the interoperability with other local editors, for example), it is important to keep the MEI rendering engine Verovio up to date. Currently, this can be done in several ways: 1) by running updates pushed by the developers, 2) by reinstalling the package 3) or by running an npm update command from the package installation folder.

User interaction and GUI

22   MEI code is displayed alongside a pane with rendered music notation (cf. figure 1). The music notation itself is part of the GUI as the SVG elements are clickable here. Moreover, most of them are mapped to the MEI elements in the text pane. For instance, clicking a note in the notation pane immediately shows the respective MEI element in the

editor pane. This also works vice versa; however, if the focused MEI element is out of range of the currently rendered page, the page has to be flipped manually.

Fig. 2: Toolbar.

23   The rendering pane provides a toolbar which is divided into the following sections (cf. figure 2): 1) Zoom and "invert colors of notation", 2) page navigation, 3) system/page break behavior and "speed mode", 4) update behavior (when re-rendering should be performed), 5) font selection, 6) navigation through the MEI elements, 7) "regenerate encoding through Verovio", info for current Verovio version and toggle button for overview of keyboard shortcuts. The toolbar is not customizable. A reason for customization could be that some functions would not be used very often and can be hidden, for example the "invert colors of notation". Somewhat irritating is the "speed mode" checkbox as it is unchecked and cannot be checked by default. However, the documentation explains that the checkbox depends on the notation break option. Otherwise, the toolbar is well structured and in line with common visual patterns including tooltips.
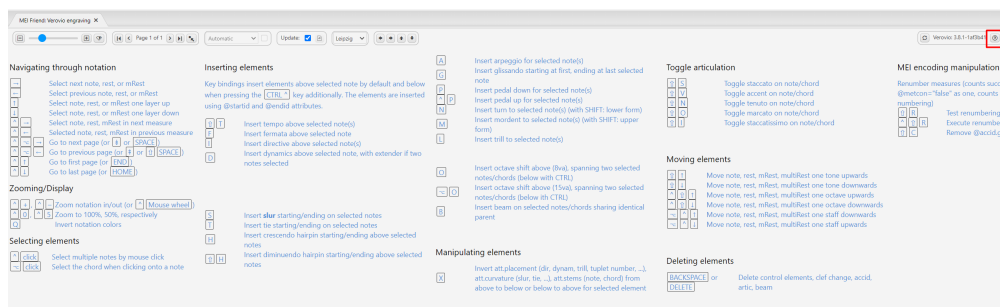

Fig. 3: Overview of the keyboard shortcuts.

24   Most user interaction happens in the music notation itself. As mentioned in the Methodology section, users can select one or more graphical elements by clicking them and applying a command. The overview of the keybindings (cf. figure 3) can be toggled with the notation pane. The list is arranged in semantically cohesive sections and provided with a verbose description. Some effort was made to find semantically appropriate shortcuts where possible. For instance, the keybinding for inserting a **s**lur is 's', for inserting an **a**rpeggio, an 'a' is used. No search function is available within this help pane. Alternatively, the keybindings can be viewed in the local package description

page[34]. The commands can also be called using the command palette[35] or from the toolbar[36].

Conclusion and outlook

25   Mei-friend can be very useful for "polishing" MEI encodings after conversion from other formats but is currently not aimed to be a full-fledged MEI editor. Display of MEI files is implemented in a user-friendly and efficient way. Speed mode – another innovation by the developers of mei-friend – significantly improves the performance of the tool. Various code editing functionalities are implemented, mainly utilizing manipulation of ControlEvents and a few attributes. The tool is currently being redeveloped and extended as a web-browser application in the context of the project Signature Sound Vienna.

Notes

1. https://web.archive.org/web/20220121074651/https://music-encoding.org/about.

2. For instance, the usage of the <app> element for alternatives (cp. https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-app.html and https://music-encoding.org/guidelines/v4/elements/app.html) or of the <supplied> element for material "supplied by the transcriber or editor for any reason" (cp. https://tei-c.org/release/doc/tei-p5-doc/en/html/ref-supplied.html and https://music-encoding.org/guidelines/v4/elements/supplied.html). Also, there are many similar concepts in the encoding of metadata (cp. <teiHeader> and <meiHead>).

3. For instance, Freischuetz Digital, Marenzio Online Digital Edition, Digital Interactive Mozart Edition, Beethovens Werkstatt.

4. The Verovio engraving library (cf. Pugin, Zitellini, and Roland, 2014).

5. https://atom.io/packages/mei-friend, currently 427 downloads [01-19-2022].

6. https://atom.io.

7. https://web.archive.org/web/20220210094706/https://iwk.mdw.ac.at/h2020-trompa.

8. The presentation at the Music Encoding Conference 2021 received the best paper award.

9. The author has been working as research assistant at the Digital Interactive Mozart Edition (Mozarteum Foundation Salzburg) since 2018 having both editorial and technical responsibilities. He has been using mei-friend since 2021 and its predecessor mei-tools-atom since 2020.

10. Only the tools for working with Common Western Notation are considered here.

11. https://www.verovio.org/mei-viewer.xhtml.

12. https://meise.de.dariah.eu.

13. https://github.com/music-encoding/oXygen-MEI-addon.

14. Sapp, 2017. https://verovio.humdrum.org.

15. https://editor.verovio.org.

16. https://web.archive.org/web/20220214130403/https://atom.io/packages/mei-tools-atom.

17. Music Encoding Initiative. 2019. *1.2.2. Events and ControlEvents* In *MEI Guidelines* (Version 4.0.1). https://music-encoding.org/guidelines/v4/content/introduction.html#eventsControlevents.

18. View > Developer > Toggle Developer Tools.

19. Cf. https://dme.mozarteum.at/movi/file/561/001.

20. Cf. https://dme.mozarteum.at/movi/file/332/002.

21. Cf. https://dme.mozarteum.at/movi/file/550/001.

22. Goebl, Weigl 2021, 8. See also the detailed explanation there.

23. For a detailed description of the implementation, see Goebl, Weigl 2021, 8-10.

24. "Works currently only with the `breaks` option set to `System(line)` and `Page and System(encoded)`." Cf. the documentation page.

25. For instance, the "flip-page" button shifts the score to the next page after the searched one.

26. https://flight-manual.atom.io/using-atom/sections/atom-packages.

27. https://web.archive.org/web/20220209081823/https://atom.io/packages/mei-friend.

28. Currently visible on GitHub only.

29. https://music-encoding.org/community/community-contacts.html.

30. Mac, Windows, Linux. Cf. macOS, Windows, Linux. Cf. https://web.archive.org/web/202221026021350/https://flight-manual.atom.io/getting-started/sections/installing-atom/.

31. For instance, the keybinding for "Show/Hide Notation" (ctrl-shift-m). As a workaround in Atom, it is possible to create custom keybindigs, cf. https://web.archive.org/web/20221122152245/https://flight-manual.atom.io/using-atom/sections/basic-customization/.

32. https://github.com/trompamusic/mei-friend.

33. https://citation-file-format.github.io.

34. File > Settings > Packages > `<mei-friend>`.

35. https://web.archive.org/web/20220209101922/https://atom.io/packages/command-palette.

36. Packages > MEI friend.

References

Goebl, W.; and M. Weigl, D. Alleviating the Last Mile of Encoding: The mei-friend Package for the Atom Text Editor. In Münnich, S.; and Rizo, D., editor(s), *Music Encoding Conference Proceedings* 2021, pages 31–39, 2022. Humanities Commons,
https://music-encoding.org/conference/proceedings.html.

Music Encoding Initiative. 2019. *MEI Guidelines* (Version 4.0.1).
https://music-encoding.org/guidelines/v4/content.

mei-friend package for Atom:
https://atom.io/packages/mei-friend.

Music Encoding Initiative. 2022.

https://web.archive.org/web/20220121074651/https://music-encoding.org/about.

Pugin, Laurent, Rodolfo Zitellini, and Perry Roland. 2014. "Verovio: A Library for Engraving MEI Music Notation into SVG.". In Proceedings of the 15th International Society for Music Information Retrieval Conference, 107– 12. Taipei, Taiwan: ISMIR. https://doi.org/10.5281/zenodo.1417589.

Sapp, Craig. 2017. "Verovio Humdrum Viewer". Presented at *Music Encoding Conference 2017* (MEC 2017), Tours, France. Slides available from http://bit.ly/mec2017-vhv. Tool available from http://Verovio.humdrum.org.

Trompamusic, https://web.archive.org/web/20220107224134/https:/iwk.mdw.ac.at/h2020-trompa.

Factsheet

Resource reviewed	
Title	mei-friend
Editors	Werner Goebl, David M. Weigl (trompamusic)
URI	https://github.com/trompamusic/mei-friend
Publication Date	2021
Date of last access	06.02.2022

Reviewer	
Name	GND Sapov, Oleksii
Affiliation	Mozarteum Foundation Salzburg
Place	Salzburg, Austria
Email	sapov (at) mozarteum.at

General information		
Software type	What type of software is it? (cf. Catalogue 0.1.1)	Software tool
Identification of the environment	On which platform runs the tool? (cf. Catalogue 1.4)	Another application
Purpose	For what purpose was the tool developed? (cf. Catalogue 1.5)	developed for a specific project or materials
Funding	Which is the financial model of the tool? (cf. Catalogue 1.6)	Free/open
Maturity	What is the development stage of the tool? (cf. Catalogue 1.5)	Release
Methods and implementation		
Programming Language	Which programming languages and technologies are used? (cf. Catalogue 2.3)	Other: JavaScript
Reuse	Does the tool reuse portions of other existing software? (cf. Catalogue 2.3)	yes
Input format	Which input formats are supported? (cf. Catalogue 2.4)	.xml, Other: .xml/mei

Output format	Which output formats are supported? ([cf. Catalogue 2.4](#))	.xml, Other: .xml/mei
Encoding	Which character encoding formats are supported? ([cf. Catalogue 2.4](#))	utf-8
Encoding preprocessing	Is a pre-processing conversion included?	yes
Dependencies	Does the documentation list dependencies on other software, libraries or hardware? ([cf. Catalogue 3.2](#))	yes
Dependencies installation	If yes, is the software handling the installation of dependencies during the general installation process (you don't have to install them manually before the installation)?	yes
Documentation and support		
Documentation	Is documentation and/or a manual available? (tool website, wiki, blog, documentation, or tutorial) ([cf. Catalogue 3.4](#))	yes
Documentation format	Which format has the documentation? ([cf. Catalogue 3.3](#))	readme
Documentation parts	Which of the following sections does the documentation contain? ([cf. Catalogue 3.3](#))	'Getting Started' section (installation and configuration), Examples
Documentation language	In what languages is the documentation available? ([cf. Catalogue 3.3](#))	English
Support	Is there a method to get active support from the developer(s) or from the community? ([cf. Catalogue 3.4](#))	yes
From of support	Which form of support is offered? ([cf. Catalogue 3.4](#))	Mailing-list
Issue tracker	Is it possible to post bugs or issue using issue tracker mechanisms? ([cf. Catalogue 3.4](#))	yes
Usability and sustainability		
Build and install	Grade how straightforward it is to build or install the tool on a supported platform: ([cf. Catalogue 3.6](#))	straightforward
Tests	Is there a test suite, covering the core functionality in order to check that the tool has been correctly built or installed?	yes

	([cf. Catalogue 3.7](#))	
Portability and interoperability	On which platforms can the tool/software be deployed? ([cf. Catalogue 3.8](#))	Linux/BSD/Unix, Mac OS X, Windows
Devices	On which devices can the tool/software be deployed? ([cf. Catalogue 3.8](#))	Desktop, Laptop
Browsers	If the tool is web-based: On which browsers can the tool/software be deployed? ([cf. Catalogue 3.8](#))	Not applicable (if not web-based for example)
Plugins	If the tool is web-based: Does the tool rely on browser plugins? ([cf. Catalogue 3.8](#))	yes
API	Is there an API for the tool? ([cf. Catalogue 3.8](#))	yes
Code	Is the source code open? ([cf. Catalogue 3.9](#))	yes
License	Under what license is the tool released? ([cf. Catalogue 3.9](#))	MIT
Credits	Does the software make adequate acknowledgement and credit to the project contributors? ([cf. Catalogue 3.9](#))	yes
Registered	Is the tool/software registered in a software repository? ([cf. Catalogue 3.9](#))	yes
Possible contribution	If yes, can you contribute to the software development via the repository/development platform?	yes
Analysability, extensibility, reusability of the code		
Analysability	Can the code be analyzed easily (is it structured, commented, following standards)? ([cf. Catalogue 3.10](#))	yes
Extensibility	Can the code be extended easily (because there are contribution mechanisms, attribution for changes and backward compatibility)? ([cf. Catalogue 3.10](#))	yes
Reusability	Can the code be reused easily in other contexts (because there are appropriate interfaces and/or a modular architecture)? ([cf. Catalogue 3.10](#))	yes
		yes

Security and privacy	Does the software provide sufficient information about the treatment of the data entered by the users? ([cf. Catalogue 3.11](#))	
Supportability and maintenance	Is there information available whether the tool will be supported currently and in the future? ([cf. Catalogue 3.12](#))	yes
Citability	Does the tool supply citation guidelines (e.g. using the Citation File Format)? ([cf. Catalogue 3.13](#))	yes
User interaction, GUI and visualization		
User profile	What kind of users are expected? ([cf. Catalogue 4.1](#))	Humanities researcher, Digital humanist
User interaction	What kind of user interactions are expected? ([cf. Catalogue 4.1](#))	Text editing, Visualization
User Interface	What kind of interface does the tool provide? ([cf. Catalogue 4.2](#) and [0.1.1](#))	Graphical User Interface (GUI)
Visualization	Does the tool provide a particular visualizations (in terms of analysis) of the input and/or the output data? ([cf. Catalogue 4.3](#))	yes
User empowerment	Is the user allowed to customize the functioning of the tool and the output configuration? ([cf. Catalogue 4.4](#))	yes
Accessibility	Does the tool provide particular features for improving accessibility, allowing „people with the widest range of characteristics and capabilities" to use it? ([cf. Catalogue 4.5](#))	yes