# 16th Century Latin Printed Brevigraphs in Unicode—a Computer Resource

Janusz S. Bień

*Abstract.* A public git repository is presented. It contains some brevigraphs, i.e., specific forms of scribal abbreviations. The brevigraphs are encoded in Unicode. They are organized into two indexes to the scans in the DjVu format: one of the abbreviated word forms and the other one inverted, i.e., of the expanded word forms. From a technical point of view the indexes are just simple CSV files. For browsing indexes, the djview4poliqarp program is recommended.

## 1. Introduction

The content of a public git repository[1] is presented, cf. also Fig. 1. The repository contains brevigraphs, i.e., a specific forms of scribal abbreviations (cf., e.g., Honkapohja 2013) found in the two editions of Stanisław Zaborowski's Latin treatise on Polish spelling entitled *Ortographia seu modus recte scribendi et legendi Polonicum idioma quam utilissimus*.

Using git and GitHub for this purpose has several advantages. The interested reader can find easily on the Internet their detailed presentations, I will mention here only the easiness of reporting mistakes and proposing corrections.

The brevigraphs are encoded in Unicode. The standard is not ideal (cf., e.g., Haralambous 2002) but we have to live with it. When needed, private characters are used, proposed by the recommendations of Medieval Unicode Font Initiative added to JunicodeTwo font courtesy of its author Peter S. Baker. The meanings and some other aspects of the brevigraphs are discussed elsewhere, namely in the paper (Janusz S. Bień, 2021).

This resource seems to be the very first computational description of brevigraphs used in printed texts. The primary reference for brevigraphs and other forms of scribal abbreviations is Capelli's *Lexicon abbreviaturarum. Dizionario di abbreviature latine ed italiane* first published in

Janusz S. Bień 0000-0001-5006-8183
retired professor, University of Warsaw, Poland E-mail: jsbien@mimuw.edu.pl

1. https://github.com/jsbien/Zaborowski-index4djview

Y. Haralambous (Ed.), *Graphonlinguistics in the 21st Century 2022. Proceedings*
Graphonlinguistics and Its Applications (ISSN: 2681-8566, e-ISSN: 2534-5192), Vol. 9.
Fluxus Editions, Brest, 2024, pp. 299–314. https://doi.org/10.36824/2022-graf-bien
ISBN: 978-2-487055-04-9, e-ISBN: 978-2-487055-05-6

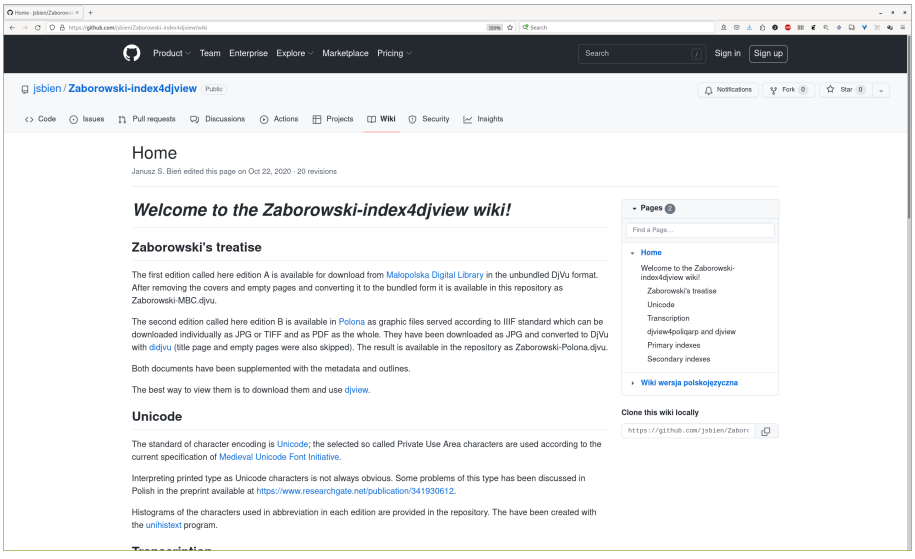


FIGURE 1. Wiki for the repository

1899 (Cappelli, 1889) and available now also as *Capelli Online*[2]. The work however describes handwritten abbreviations, represented by scan snippets, which quite often have no printed equivalents. It's worth noting that the online version was created by crowdsourcing (the call for volunteers was announced in 2015[3]) and the results are freely available also in the source form[4].

The very first computational approach to Latin abbreviations seems to be Olaf Pluta's Abbreviationes™: A Database of Medieval Latin Abbreviations awarded the 1993 German-Austrian Academic Software Prize (Deutsch-Österreichischer Hochschul-Software-Preis) for outstanding software in the humanities[5]; you can find numerous screenshots in (Pluta, 1995). In 2015 a new version, called Abbreviationes™ Professional, was released. It is said that it

> provides a standardized representation of medieval Latin abbreviations by using a Unicode-compliant font (Junicode, created by Peter S. Baker, University of Virginia) which follows the character recommendations of MUFI (Medieval Unicode Font Initiative).

2. https://www.adfontes.uzh.ch/en/ressourcen/abkuerzungen/cappelli-online
3. https://web.archive.org/web/20171015135838/http://www.adfontes.uzh.ch/cappelli/index.php
4. https://data.europa.eu/data/datasets/adfontes-cappelli-abbreviaturarum-openglam
5. https://olafpluta.net/software/software.html

To use the software a paid license is required, the price ranges from 99€ for a single fixed IPv4 address to 1199€ for a class B subnet. A trial access is available for free but The author has not used it as he has no intention to purchase this product. The software is mentioned in a chapter of *The Oxford Handbook of Latin Palaeography* (Pluta, 2020).

According to Honkapohja (2021, p. 27) the ORIFLAMMS project made lists of medieval manuscript abbreviations available on GitHub, but I was unable to locate it. Anyway the paper mentions several projects which encode abbreviations and their expansions in the text corpora, but all of them seem to represent manuscripts.

The repository presented here is an open resource. Everybody can use it for any pupose, modify it and distribute modified version etc.

## 2. Printed Texts

The basic notion of the traditional (letterpress) printing is type or sort (both names are confusing because of the ambiguity), i.e., a piece of metal with a face with the (reversed) image of the character to be printed with some appropriate ink on the paper, cf. Fig. 2. A more basic notion is the matrix (a mould) used to cast the types/sorts. For the purpose of encoding we can assume that all types/sorts casted from a single matrix are identical. We propose to call an image of a type/sort on paper, *typoglyph*; even in a single document they can differ because, e.g., of some paper glitch. A generalized, by ignoring such differences, typoglyphs I propose to call typographical characters, in short typochars (Janusz S. Bień, 2016–2017 [2019]).

Types/sorts has been kept in the compartments of typecases[6].

Following (André and Jimenes 2013) the types/sorts put into a single compartment are considered an abstract *typème*. For different type sizes different typecases has been used, so the size is not a property of *typème*, and the same rule holds for some other properties. On the other hand a character with accents is considered a single *typème*, because it is the image of the face of a single type/sort. The notion of typographical character mentioned above has been inspired by the notion of *typème*.

6. You can find different cases and their lays/arrangements, e.g., at `http://www.alembicpress.co.uk/Alembicprs/SELCASE.HTM`.

FIGURE 2. A ligature type and its printed image (Daniel Ullrich, Threedots, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=855947)

## 3. Unicode

### 3.1. Basics

Let's start this section with a quote from (Korpela, 2006, p. xii): *Character code problems are intrinsically difficult, and very widely misunderstood.*

To make a long story short, Unicode characters have only a loose relation to the characters we use in print or in writing (sometimes called user-perceived characters). One reason is that characters with one more diacritical signs are in principle represented as a base character and the combining characters representing the diacritics; a limited number of characters are available as precomposed ones. In consequence a printed character can be represented by several equivalent Unicode character sequences. Such a sequence is technically called an extended graphemic cluster, which in the author's opinion is a misnamer (a very limited number of these sequences are really graphemic clusters). There is no official Unicode term for the abstraction class of a character object independently of its representation. We propose to call it a *textel* (a text element). However it seems that extended graphemic clusters acquire double meaning: that of a specific sequence and that of a representation independent object. This is how the author understands its use in the SWIFT programming language, cf. also the whole thread on the Unicode mailing list[7].

### 3.2. Extending Unicode

In theory Unicode can be easily extended by interested communities by agreeing on the use of private characters. Medieval Unicode Font Initiative mentioned above is a good example. However the private characters are crippled because they are missing the properties provided by

7. https://www.unicode.org/mail-arch/unicode-ml/y2016-m09/0035.html

the Unicode Character Database[8]. Even if characteristic properties are provided[9], it may be impossible or prohibitively difficult to make programs to use them; Emacs seem to be an exception[10].

One of the main Unicode design principles is the distinction between the characters (some abstract objects) and glyphs (their visual images). Another important principle is that Unicode encodes characters, not glyphs. Unfortunately the difference is not always clear.

Let us consider an example. In (Everson et al., 2006, pp. 6, 22, 30, 34, 38) a character was proposed which was used in Latin as an abbreviation for *el*, *ul* and *vel* and in Norse as an abbreviation for *eða*, *el*, *æl* and *al*. The glyphs of this character are presented in Fig. 3. One of them was used as the so called representative glyph in the standard (5.1.0 introduced in 2008) and it served also as the inspiration for the character name.

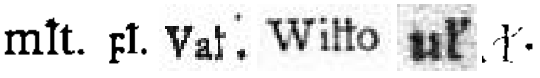

FIGURE 3. Different glyphs of U+A749 LATIN SMALL LETTER L WITH HIGH STROKE

Theoretically there is a method to circumvent these principles by using so called character variation sequences.

The Unicode FAQ[11] contains the following explanation:

> Every character in Unicode can be displayed with many different glyphs: An "a" can be displayed with or without the top "hook" (a versus ɑ). A not-equals sign (≠) can be displayed with an angled or vertical slash, and so on.
>
> In some situations, however, it is important to indicate in plain text that only a subset of the possible glyphs for a character should be used, such as a vertical slash for ≠. The variation sequences are a standardized mechanism for requesting such an appearance.

The following FAQ fragment looks even as a recommendation to use variation sequences:

> Q: I'm proposing an addition to a historic script that is a variant of an existing character. Should I propose it as a new character or as a new variation sequence?
>
> A: Variation sequences provide a means to specify a certain significant glyphic variation of a character, without encoding each variation as a separate character. This is particularly useful whenever such distinction is not universally necessary.

8. Cf., e.g., https://util.unicode.org/UnicodeJsps/character.jsp?a=A749&B1=Show
9. Cf., e.g., http://www.kreativekorp.com/charset/PUADATA/
10. https://debbugs.gnu.org/cgi/bugreport.cgi?bug=32599
11. http://unicode.org/faq/vs.html

> Because the character itself is part of the variation sequence, one should be able to search and find all the instances of that particular character, independent of variation in its appearance, a task which would be more complicated if the variants were encoded as separate characters. If you can replace the variant by the existing character without significantly distorting the content of the text, then a variation sequence is the appropriate way to represent the variant, and you should propose your addition as a variation sequence.
>
> For historic scripts, the variation sequence provides a useful tool, because it can show mistaken or nonce glyphs and relate them to the base character. It can also be used to reflect the views of scholars, who may see the relation between the glyphs and base characters differently. Also, new variation sequences can be added for new variant appearances (and their relation to the base characters) as more evidence is discovered.

The problem is that every usage of variation sequences has to be officially registered by the Unicode Consortium. The only proposal related to Latin scripts the author is aware of, namely (Pentzlin, 2011), has been discussed by the Unicode Technical Committee on a meeting in February 2011 but no action was taken (we are obliged to the author of the proposal for providing this information). The fact does not encourage to submit new proposals.

Several year ago the word *Emojigeddon* was coined, which refers to the flood of emojis accepted into Unicode[12] while the original goals of the standard seem to be neglected[13]. The emojis paved the way for the intensive use of the so called tag characters, e.g., the flag of Scotland is represented as the sequence BLACK FLAG, TAG LATIN SMALL LETTER G, TAG LATIN SMALL LETTER B, TAG LATIN SMALL LETTER S, TAG LATIN SMALL LETTER C, TAG LATIN SMALL LETTER T, CANCEL TAG[14]. Tags have no glyphs, but for the editing and documentation purposes they can be visualised (see below).

In March 2022 Margaret Kibi (marrus-sh) proposed to use tags instead of the variant sequences in the Junicode font[15]. The proposal was supported by several other font users and accepted by the font author. You can find a non-trivial example of this technique in (Janusz S. Bień, 2022b). We hope it will become a kind of a *de facto* standard.

In consequence the important glyph variant of the character discussed above, namely one used in particular in (Balbi, 1460), cf. Fig. 4, which was a book probably typeset by Gutenberg himself, can be en-

---

12. Cf., e.g., https://www.buzzfeednews.com/article/charliewarzel/inside-emojigeddon-the-fight-over-the-future-of-the-unicode

13. Cf., e.g., https://www.explainxkcd.com/wiki/index.php/1953:_The_History_of_Unicode

14. Cf., e.g., https://emojipedia.org/flag-scotland/

15. https://github.com/psb1558/Junicode-font/discussions/122\#discussioncomment-2416880

FIGURE 4. Giovanni Balbi *Catholicon*, 1460 r

coded as lſf yielding ﬕ (it can be named *l with high stroke ending with flourish*).

Moreover LATIN SMALL LETTER L WITH HIGH STROKE is obviously just a variant of LATIN SMALL LETTER L. Knowledge of this fact can be useful, e.g., for searching and indexing; it is quite surprising that the standard does not provide this information, at least not explicitely. We can now express it formally by encoding ﬕ as lſh. You can find more examples in (Baker, 2022).

The author proposes to call *textons* the instances of Unicode characters which are not used in a self-contained way but are just elements of some sequences. The term was introduced with a slightly different meaning in (Janusz S. Bień, 2016).

## 3.3. Examples of Encoding Problems

Let's have now a look at a specific encoding problem described already in (Janusz S. Bień, 2021).

At first glance the character for *el* seen on Fig. 5 is not available in Unicode.

FIGURE 5. From the left: *vel*, *vel*, *regula*, *populus* (Zaborowski's treatise)

However if you search thoroughly the character proposals stored in the Unicode archive, you will find the proposal mentioned above and learn that this is just a different glyph for U+A749 LATIN SMALL LETTER L WITH HIGH STROKE. It's a pity you cannot find this information directly in the standard.

Let's now have a look at another example, namely the abbreviations of *aliter* and *similiter*, cf. Fig. 6; they come from a recently digitized Zaborowski's treatise edition which has not yet an index, cf. (Janusz S. Bień, 2022a).

The diacritic over t looks like a comma. However the character U+0313 COMBINING COMMA ABOVE present in Unicode from its beginning

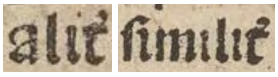

FIGURE 6. From the left: *aliter*, *similiter* (Zaborowski's treatise)

has a different purpose: principally it's the Greek *psili* (smooth breathing mark), although it has some additional applications.

Let us note that Medieval Unicode Font Initiative is not bound by the Unicode rule to encode only characters, not glyphs. So although the Unicode standard has only U+035B COMBINING ZIGZAG ABOVE, in the MUFI specification we can find also U+F1C7 COMBINING ABBREVIATION MARK ZIGZAG ABOVE ANGLE FORM and U+F1C8 COMBINING ABBREVIATION MARK ZIGZAG ABOVE CURLY FORM. So it seems the abbreviations on the Fig. 6 can be encoded as aliᷜ and ſimılıᷜ; the shapes are not identical to those in the scan, but the distinction is preserved. If you want to avoid the private characters for the reasons described earlier, with the Junicode font you can encode them also as ◌᷽ and ◌᷾.

The last example, cf. Fig. 7, comes again from (Janusz S. Bień, 2021).

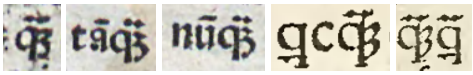

FIGURE 7. From the left: *quam*, *tanquam*, *nunquam*, *quicquam*, *quamquam* (Zaborowski's treatise)

There is no doubt about the characters U+A76B LATIN SMALL LETTER ET added to the standard (together with U+A76A LATIN CAPITAL LETTER ET) in 2008 (version 5.1.0.) following the letter *q*, and the ligature U+E8BF LATIN SMALL LETTER Q LIGATED WITH FINAL ET, a private character present in the MUFI recommendation (Haugen, 2015, p. 79) since version 2. There is also no doubt about U+A757 LATIN SMALL LETTER Q WITH STROKE THROUGH DESCENDER added to the standard (together with U+A756 LATIN CAPITAL LETTER Q WITH STROKE THROUGH DESCENDER) in 2008 (version 5.1.0).

What is problematic here is the encoding of the diacritics. In *tanquam* we have a straight line which can be encoded as U+0305 COMBINING OVERLINE or U+0304 COMBINING MACRON. In *nunquam* the line is not quite straight, is this the same diacritics as in *nunquam* or perhaps a form of tilde (U+0303 COMBINING TILDE)? The words *quam*, *tanquam*, and *nunquam* contain also a diacritic similar to diaeresis, but the dots are more or less connected, is this accidential or intentional? At present the author is not sure what the answer is.

As for *quicquam*, we can assume that LATIN SMALL LETTER ET represents *m* (it used to be written vertically to save the space), hence the meaning of the diacritic is *ua*. In the Unicode archive we can find the document (Everson et al., 2006, s. 8) stating that *ua* can be abbreviated by U+1DD3 COMBINING LATIN SMALL LETTER FLATTENED OPEN A ABOVE. Although the shape of this character in the document is not identical to our example, it seems reasonable to assume this is just a different glyph. This interpretation seem also be confirmed by Erin Blake[16], who calls the character *jagged horizontal line above letter*. The same diacritic sign occurs twice in an abbreviation, which means *quamquam*; this and other readings quoted here come from (Urbańczyk, 1983, p. 90).

Let us hope that analysing more texts in the future will allow to formulate the definite answers to the questions raised above.

## 4. DjVu

The format was developed in 1999-2001 for serving scans and underlying text layer over Internet. It's acceptance was hampered by the patents (looks like most of them expired by now), nevertheless it was quite popular in digital libraries, especially in Poland. The open source viewer djview4[17] is still actively maintained, cf., e.g., Fig. 8. The DjVu plugin for browser used the now not supported NPAPI interface and no equivalently convenient tool was created. However in my opinion DjVu is still a very good format for scanned documents used offline. One of its advantages is the simplicity of the format.

The DjVuLibre[18] library provides various tools, in particular djvused used for operations on the text layer, annotations and metadata.

A typical DjVu document internally contains a dictionary of glyphs (actually connected components), which can be viewed with the djview4shapes program[19], cf. Fig. 9. Another tool for visualisation of connected components dictionaries is Alexander Trufanov's djvudict program[20].

The dictionary of glyphs can be created in particular with minidjvu-mod[21] and minidjvu-mod-gui[22].

16. https://collation.folger.edu/2021/09/brevigraphs/
17. http://djvu.sourceforge.net/djview4.html
18. http://djvu.sourceforge.net/
19. https://bitbucket.org/mrudolf/djview-shapes/
20. https://github.com/trufanov-nok/djvudict.git
21. https://github.com/trufanov-nok/minidjvu-mod
22. https://github.com/trufanov-nok/minidjvu-mod-gui

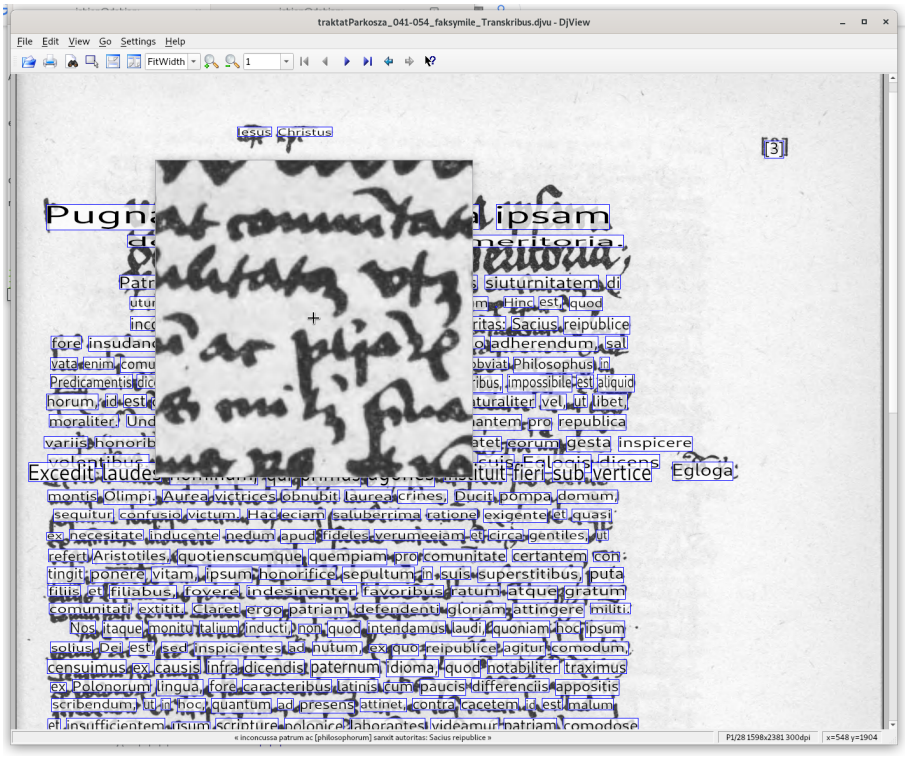


FIGURE 8. djview4: the scan and the underlying text

Looking at the glyph dictionary is the quickest way to get an overview of glyphs used in a text which can help to make the right encoding decisions.

## 5. Indexes to DjVu Documents

The indexed discussed here has been designed by Janusz S. Bień and implemented by Michał Rudolf in the djview4poliqarp[23]. From the technical point of view they are just simple CSV files (with the semicolon as the separator). They can be processed in any way a user wishes (we fully agree with Peter Robinson n.d.), but they are most conveniently browsed and edited with the djview4poliqarp program mentioned above. The program was originally designed to facilitate creating graphical concordances for the corpora search results, in particular for the

23. https://bitbucket.org/mrudolf/djview-poliqarp/

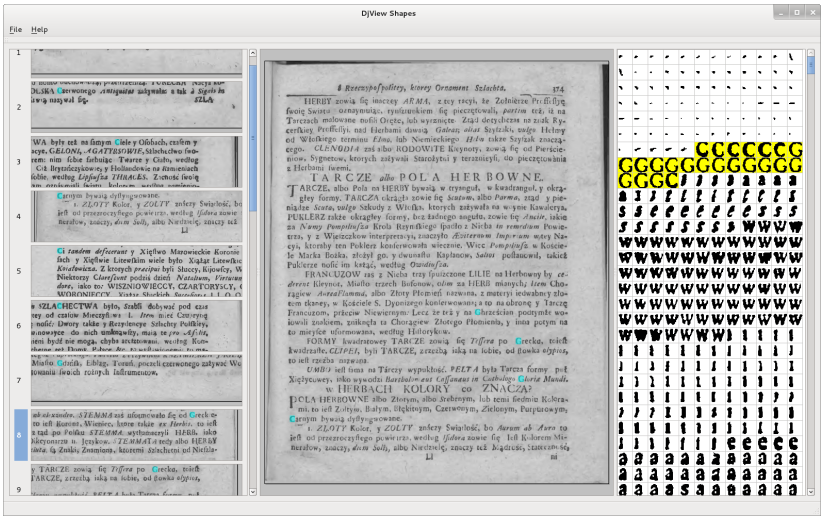


FIGURE 9. djview-shapes: similar shapes grouped together

so called IMPACT Polish Ground Truth corpus[24], cf., e.g., (Janusz S. Bień, 2014), but was later adapted to handle also offline indexes, cf., e.g., (Janusz S Bień, 2018a) and (Janusz S. Bień, 2018b), to both online and offline DjVu documents.

The brevigraph indexes discussed here are based on the two editions of Zaborowski's treatise.

The first edition called here edition A is available for download from Małopolska Digital Library[25] in the so called unbundled DjVu format. After removing the covers and empty pages and converting it to the bundled (single file) form it is available in the repository as Zaborowski-MBC.djvu.

The second edition called here edition B is available in Polona digital library[26] as graphic files served according to IIIF standard which can be downloaded individually as JPG or TIFF and as PDF as the whole. They have been downloaded as JPG and converted to DjVu with didjvu[27]; title page and empty pages were also skipped. The result is available in the repository as Zaborowski-Polona.djvu.

Both documents have been supplemented with the metadata describing their origin and the outlines containing traditional page identifiers

24. https://szukajwslownikach.uw.edu.pl
25. http://mbc.malopolska.pl/publication/89609
26. https://polona.pl/item/73794330
27. http://jwilk.net/software/didjvu

(the identification of the sheet, the numer of the leaf in the sheet, the side *recto* or *verso*).

The best way to view them is to download them and use djview4 program mentioned earlier.

The indexes are named respectively ZaborowskiA.cvs and ZaborowskiB.csv. We called them the primary indexes.

Every line of an index file consists of three or four fields:

1. The entry used for sorting and incremental search. The entries in the primary indexes consist of the abbreviations, e.g., `mãib⁹'.
2. The reference to the relevant image fragment in the form used by djview4 viewer mentioned earlier, namely an Universal Resource Locator. In the indexes discussed here the scheme and authority parts are absent, and the path limited to the file names, this means in practise that djview4poliqarp has to be called with the index directory as the default one. The fragment part is also missing, and the query part contains the dimensions and the coordinates of the image fragment in the djview4 specific form; it can contain also the specification of a color used for highlighting. This field is created with an appropriate tool. In particular djview4 and djview4poliqarp can be used for this purpose. Here is an example:

   ```
   Zaborowski_MBC.djvu?djvuopts=&highlight=561,954,133,58&page=1
   ```

3. A description: a text displayed for the current entry in a small window under the index.
4. An optional comment displayed after the entry. In the primary index this is the abbreviated word, proceeded by REFERENCE MARK for a more distinctive display, e.g., ※ manibus.

The entries can be displayed in several orders:

- File order, in practise it means the order of the brevigraphs occurences in the treatise.
- Alphabetic order word by word, i.e., spaces and hyphens are relevant.
- Alphabetic order letter by letter, i.e., spaces and hyphens are ignored.
- *a tergo* (the reverse alphabetical order).

The indexes contain also some additional auxiliary entries.

First of all there are entries describing words which are not abbreviations but are interesting for other reasons; in particular, they document the usage of LATIN SMALL LETTER ET as a final 'm'.

Secondly, they are entries allowing, when displaying an index in the file order, to move quickly to a specific page or, when both A and B indexes are diplayed together, to the beginning of an edition.

As we have seen already, interpreting printed type as Unicode characters is not always obvious. For verification purposes the histograms

of the Unicode characters used in the abbreviations in each edition are also provided in the repository. They have been created with the unihistext program[28].

There are also the secondary indexes named respectively ZaborowskiAi.cvs and ZaborowskiBi.csv ('i' meaning 'inverted'). In the secondary indexes the fields 1 and 4 are exchanged, so the abbreviated words are now the entries. They are generated from the primary ones with a sed one-liner program.

Loading both indexes and sorting the joined index in an alphabetic order allows to compare how the words were abbreviated in the A and B editions.

The transcription, based on the one provided by Urbańczyk (1983), has been synchronized with the scans with the use of Transkribus[29]. The results has been used as the basis for indexes, which were later verified and extensively modified. These changes has not been applied yet to the texts stored in the Transkribus system.

In djview4poliqarp program you can use the left panel for displaying the entries you find interesting, cf. Fig. 10 and Fig. 11.

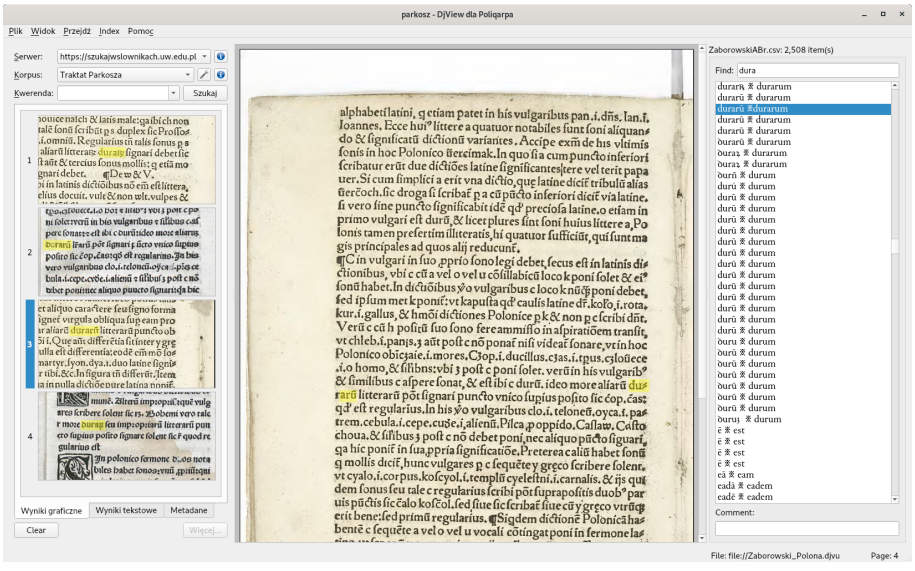


FIGURE 10. A primary index

28. https://bitbucket.org/jsbien/unihistext/
29. https://readcoop.eu/transkribus/

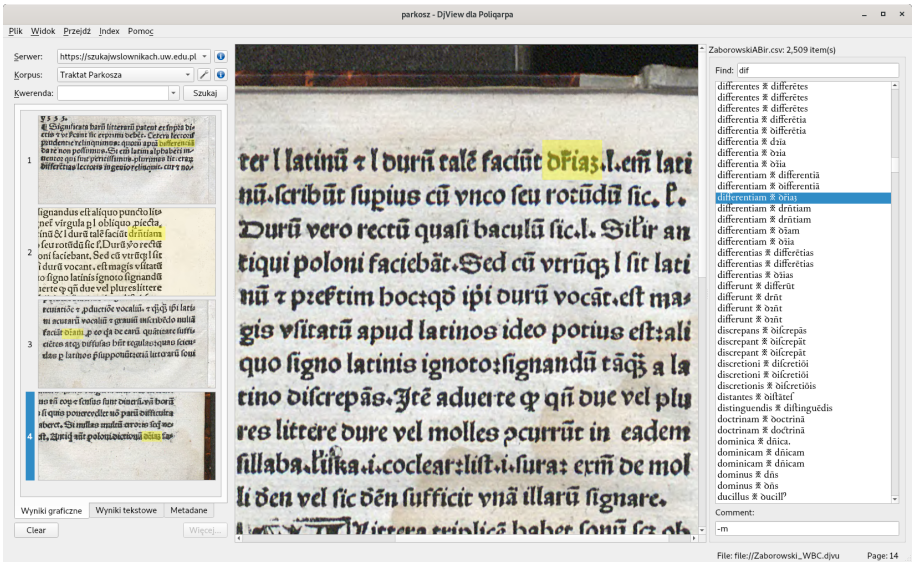


FIGURE 11. A secondary index

As it was mentioned before, the indexes can have other uses beside being browsed. For example, it is quite easy to convert, with just some regular expressions, an index into the djvused input to create a DjVu document where abbreviations are somehow marked and the expansions provided as tooltips, cf. Fig. 12.

## 6. Final Remarks

The paper (Honkapohja, 2021) entitled *Digital Approaches to Manuscript Abbreviations: Where Are We at the Beginning of the 2020s?* was already mentioned earlier. It's main focus is the place of abbreviations in the theory of writing systems, but it contains also a section concerning computer encoding of abbreviations and/or their expansions. With the exception of some early corpora, all the projects mentioned encode the texts in XML, most of them following the recommendations of Text Encoding Initiative[30], which discusses abbreviations in section 3.6.5[31].

Perhaps in some future Zaborowski's treatise will be also encoded in TEI XML, but for the present purpose the tools and resources described in the paper are fully adequate.

30. `https://tei-c.org/`
31. `https://tei-c.org/release/doc/tei-p5-doc/en/html/CO.html#CONAAB`

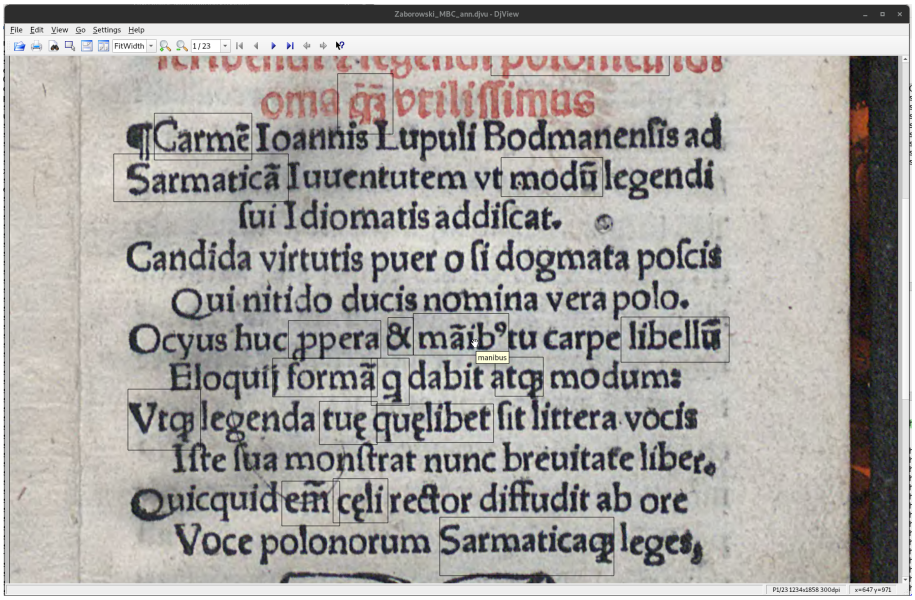


FIGURE 12. A tooltip with the expansion of mãib⁹ abbreviation

## References

André, Jacques and Rémi Jimenes (2013). "Transcription et codage des imprimés de la Renaissance." In: *Revue des Sciences et Technologies de l'Information—Série Document Numérique* 16.3, pp. 113–139.

Baker, Peter S. (2022). *Junicode—the font for medievalists. Specimens and user manual for version 2*. https://github.com/psb1558/Junicode-font.

Balbi, Giovanni (1460). *Catholicon*. https://www.loc.gov/item/47043559/. Mainz.

Bień, Janusz S (2018a). "Elektroniczny indeks do słownika Lindego [An electronic index to Linde's dictionary]." In: *Kwartalnik Językoznawczy* 2015.3, pp. 1–19.

———— (2018b). "Elektroniczne indeksy fiszek słownikowych [Electronic indexes for dictionary fiches]." In: *Kwartalnik Językoznawczy* 16.2, pp. 16–27.

———— (2022a). "Polskie litery w traktacie Stanisława Zaborowskiego. Litera A i pochodne [Polish letters in Stanisław Zaborowski's treatise. Letter A and derivates]." In: *Poznański Półrocznik Językoznawczy* 1, pp. 1–20.

———— (2016). "Problemy kodowania znaków w korpusach historycznych [Character encoding problems in historical corpora]." In: *Semantyka a konfrontacja językowa*. Ed. by Danuta Roszko and

Joanna Satoła-Staśkowiak. Vol. 5. Warszawa: Instytut Slawistyki PAN, pp. 67–76.

Bień, Janusz S. (2016–2017 [2019]). "Repertuar znaków piśmiennych—problemy i perspektywy [Towards an electronic repertoire of basic text elements]." In: *Kwartalnik Językoznawczy* 2016.2016/4-2017/1, pp. 1–18.

———— (2022b). "Representating Parkosz's alphabet in the Junicode font." In: *TUGboat* 43.3, pp. 247–251.

———— (2014). "The IMPACT project Polish Ground-Truth texts as a DjVu corpus." In: *Cognitive Studies | Études Cognitives* 14, pp. 75–84.

———— (2021). "Traktat Stanisława Zaborowskiego i skróty brachygraficzne [Scribal abbreviations in Zaborowski's treatise]." In: *Poznański Półrocznik Językoznawczy* 1 (30), pp. 1–42.

Cappelli, Adriano (1889). *Lexicon abbreviaturarum. Dizionario di abbreviature latine ed italiane*. Milan: Ulrico Hoepli.

Everson, Michael et al. (2006). *Proposal to add medievalist characters to the UCS*. Tech. rep. N3027. ISO/IEC JTC1/SC2/WG2.

Haralambous, Yannis (2002). "Unicode et typographie: un amour impossible." In: *Document numérique* 6.3, pp. 105–137.

Haugen, Odd Einar, ed. (2015). *MUFI character recommendation version 4.0*. http://hdl.handle.net/1956/10699. Medieval Unicode Font Initiative.

Honkapohja, Alpo (2013). "Manuscript abbreviations in Latin and English: History, typologies and how to tackle them in encoding." In: *Studies in Variation, Contacts and Change in English. Principles and Practices for the Digital Editing and Annotation of Diachronic Data*. Vol. 14. https://varieng.helsinki.fi/series/volumes/14/honkapohja/.

———— (July 2021). "Digital Approaches to Manuscript Abbreviations: Where Are We at the Beginning of the 2020s?" In: *Digital Medievalist* 14.

Korpela, Jukka K. (Jan. 2006). *Unicode Explained*. Sebastopol, CA: O'Reilly.

Pentzlin, Karl (2011). *Proposal to add Variation Sequences for Latin and Cyrillic letters*. Tech. rep. L2/11-059. ISO/IEC JTC1/SC2/WG2 and UTC.

Pluta, Olaf (1995). *Abbreviationes, the first electronic dictionary of medieval Latin abbreviations*.

———— (2020). "Abbreviations." In: *The Oxford Handbook of Latin Palaeography*. Ed. by Frank T. Coulson and Robert G. Babcock. Oxford: Oxford University Press, pp. 9–24.

Robinson, Peter (n.d.). "Why Interfaces Do Not and Should Not Matter for Scholarly Digital Editions." https://www.slideshare.net/PeterRobinson10/why-interfaces-do-not-and-should-not-matter-for-scholarly-digital-editions.

Urbańczyk, Stanisław (1983). *Die altpolnischen Orthographien des 16. Jahrhunderts*. Ed. by Stanisław Urbańczyk and Reinhold Olesch. Vol. 37. Slavistische Forschungen. Köln-Wien: Böhlau.