16th century Latin printed brevigraphs in Unicode: a computer resource

Janusz S. Bień
(retired)
jsbien@uw.edu.pl

Abstract
The content of a public git repository is presented. It contains the brevigraphs, i.e. a specific forms of scribal abbreviations 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*. The brevigraphs are encoded in Unicode following the principles of typemic transcription.

Introduction

The very first computational approach to Latin abbreviation seems to be Olaf Pluta's *Abbreviatones™: 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. In 2015 a new version, called *Abbreviationes™ Professional*, was released, which uses Unicode. To use the software a paid license is required.

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

1 Printed texts

Typecases and *typème*; typoglyphs and typochars. Cf. Fig. 9.

2 Unicode

2.1 Basics

Unicode characters are not the same as user-perceived characters. The notions of *typeme* and *textel* was proposed to remedy this.

In theory Unicode can be easily extended by interested communities by agreeing on the use of private characters. Medieval Unicode Font Initiative is a good example. However the private characters are crippled because they are missing the properties provided by the Unicode Character Database. Even if the properties are provided, it is impossible or prohibitively difficult to make programs to use them.

One of the main Unicode design principle is the distinction between the characters (some abstract objects) and glyphs (their visual images). Another important principle is that Unicode encodes characters, not glyphs. It is difficult to apply this principle in practise, because the difference is not always clear.

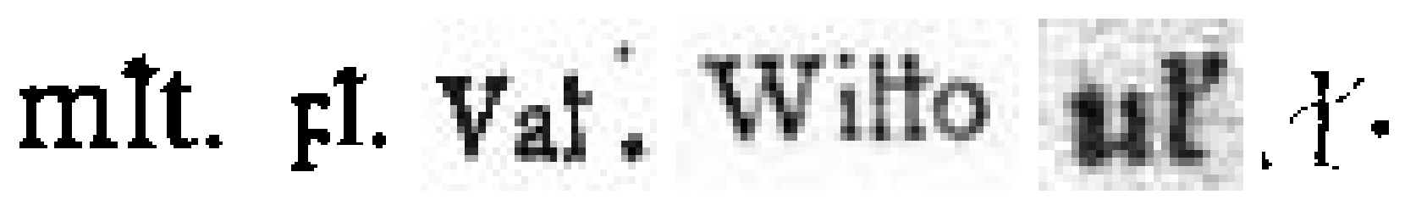

Figure 1: Different glyphs of U+A749 'LATIN SMALL LETTER L WITH HIGH STROKE'
— the representative glyph is ꝉ

Theoretically there is a method to circumvent this principles by using so called character variation sequences. The problem is that every usage of variation sequences has to be officially registered by the Unicode Consortium. Up to now no proposal concerning the Latin script has been accepted.

In March 2022 Margaret Kibi (marrus-sh) proposed to use tag characters instead of the variant sequences in the Junicode font; the tags themselves are invisible but can be be visualized for documentation purposes, see below. The proposal was supported by several font users and accepted by the font author. I hope it will become a kind of a *de facto* standard.

In consequence the important glyph variant of the character discussed above can be encoded as l󠁳󠁦 yielding ꝉ (it can be named *l with high stroke ending with flourish*).

2.2 Examples of encoding problems

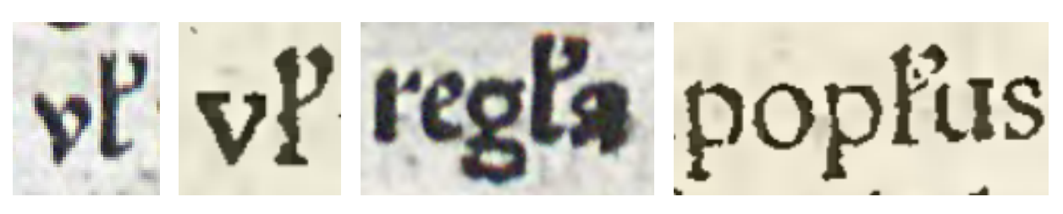

Figure 2: From the left: *vel*, *vel*, *regula*, *populus* (Zaborowski's treatise)

'LATIN SMALL LETTER L WITH HIGH STROKE' (ꝉ) not obvious, Junicode l󠁳󠁦 (ꝉ: *l with high stroke ending with flourish*).

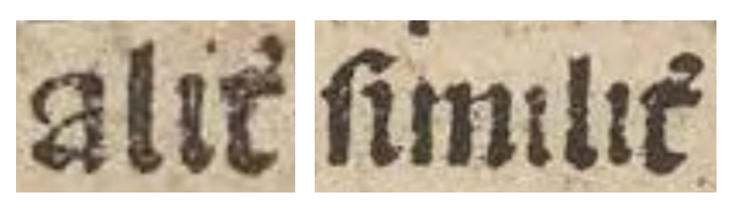

Figure 3: From the left: *aliter*, *similiter* (Zaborowski's treatise)

U+035B 'COMBINING ZIGZAG ABOVE'? U+F1C7 'MUFI COMBINING ABBREVIATION MARK ZIGZAG ABOVE ANGLE FORM' and U+F1C8 'MUFI COMBINING ABBREVIATION MARK ZIGZAG ABOVE CURLY FORM'? Junicode: ◌͛ (◌͛󠁣󠁵) and ◌͛ (◌͛󠁡󠁮)?

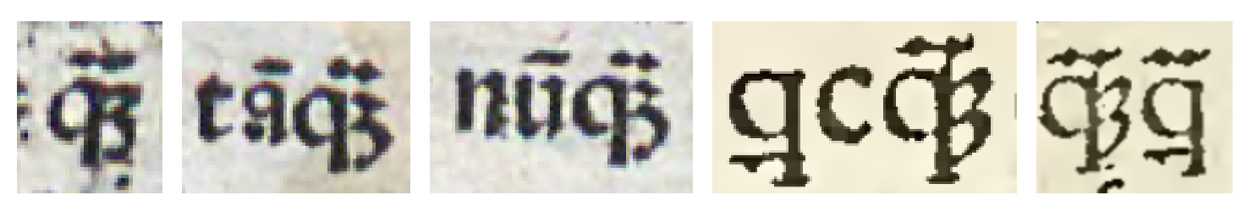

Figure 4: From the left: *quam*, *tantquam*, *nunquam*, *quicquam*, *quamquam* (Zaborowski's treatise)

U+A76B 'LATIN SMALL LETTER ET', U+E8BF 'MUFI LATIN SMALL LETTER Q LIGATED WITH FINAL ET', U+A757 'LATIN SMALL LETTER Q WITH STROKE THROUGH DESCENDER'

U+0305 'COMBINING OVERLINE'? U+0304 'COMBINING MACRON'? U+0303 'COMBINING TILDE'? U+1DD3 'COMBINING LATIN SMALL LETTER FLATTENED OPEN A ABOVE'?

3 DjVu

3.1 The format

The format was developed in 1999-2001 for serving scans and underlying text layer over Internet. The open source viewer djview4 is still actively maintained. A typical DjVu document internally contains a dictionary of glyphs (actually connected components).

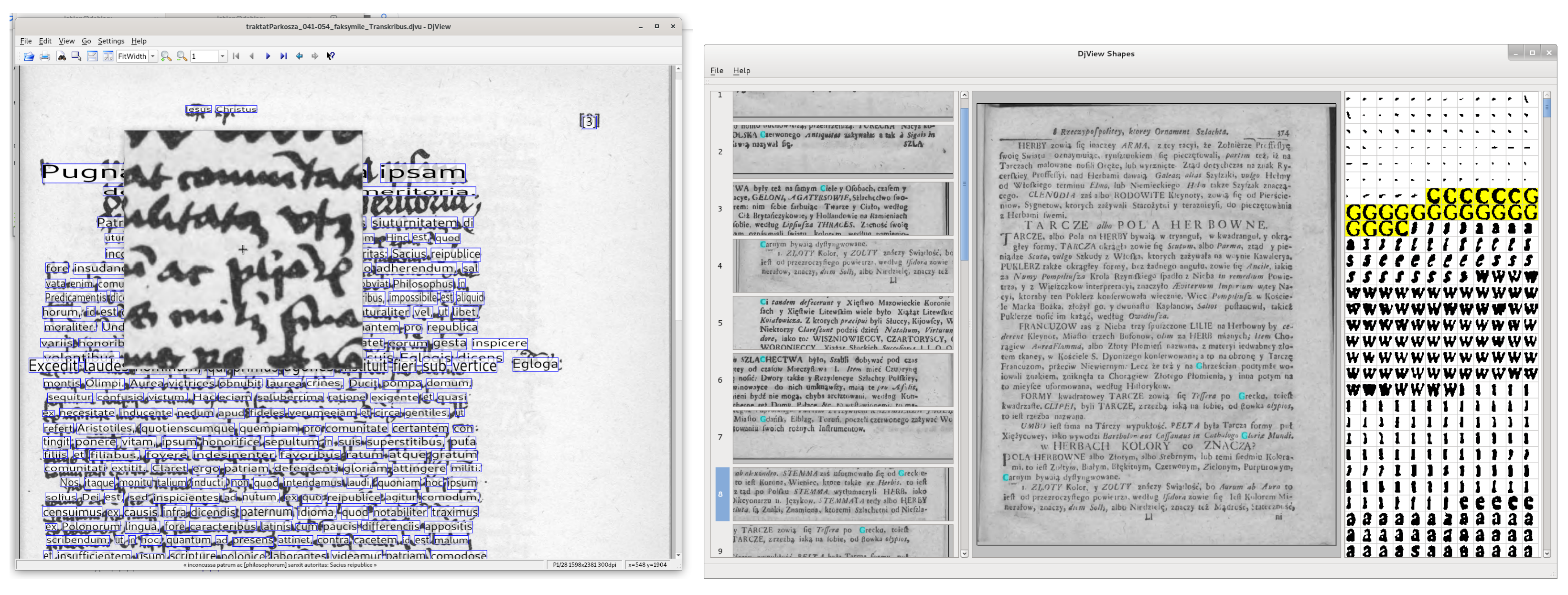

Figure 5: djview4: the scan and the underlying text, djview-shapes: shapes similar to the letter *C*

3.2 Indexes for DjVu documents

From the technical point of view the indexes are just simple CSV files (with the semicolon as the separator). The indexes can have other uses beside being browsed with djview4poliqarp.

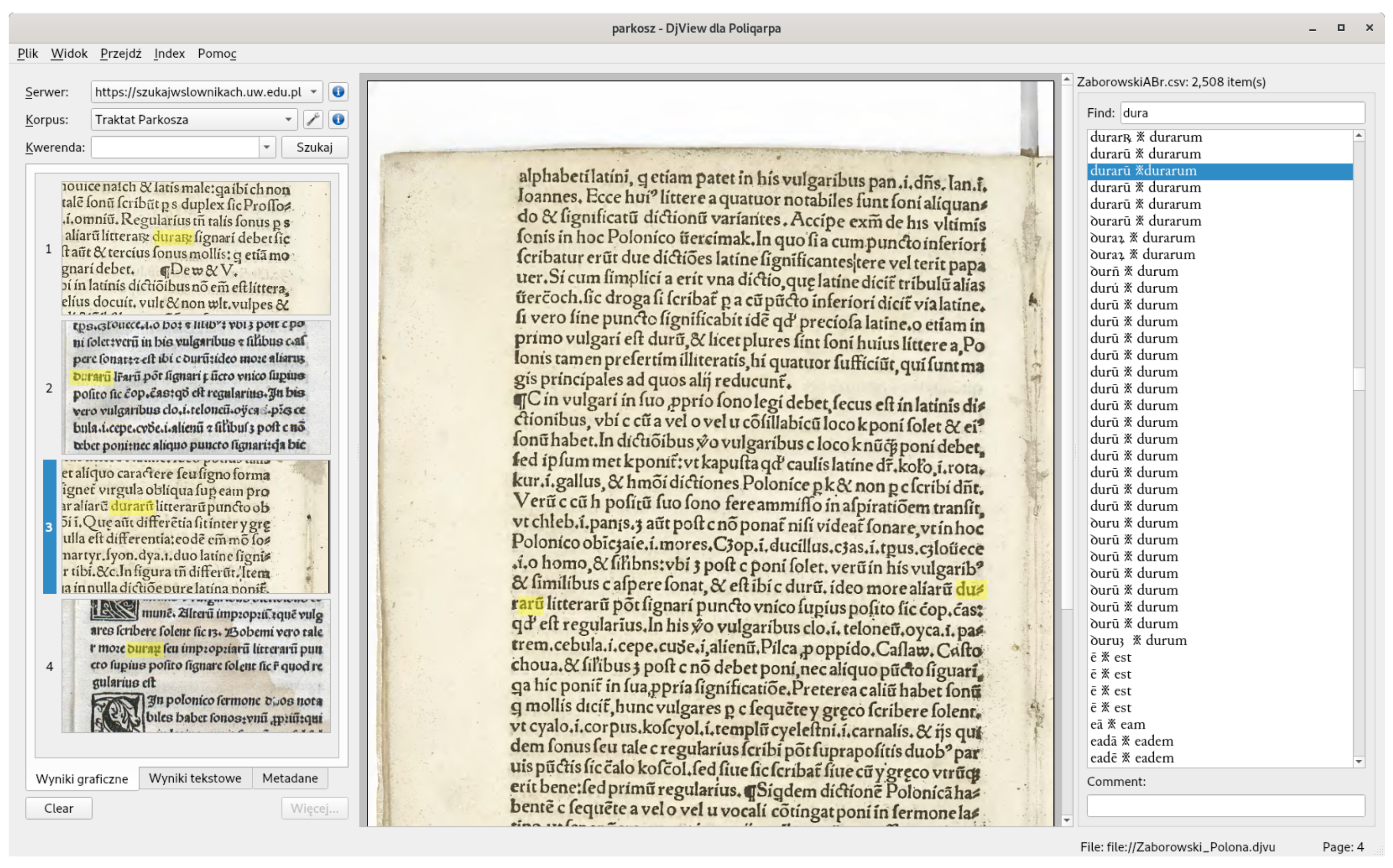

Figure 6: A primary index: the entry is an abbreviation

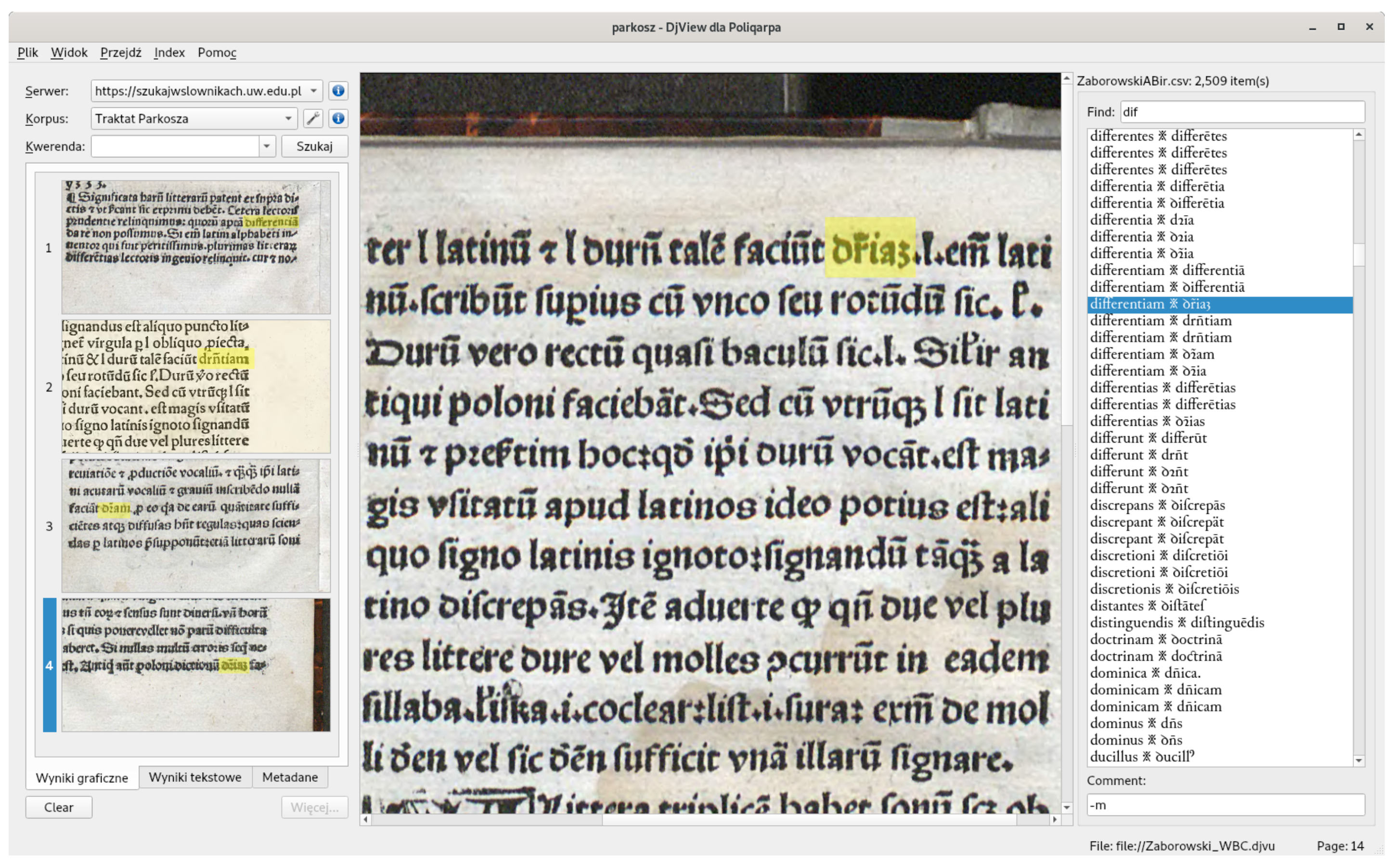

Figure 7: A secondary index: the entry is the abbreviated word

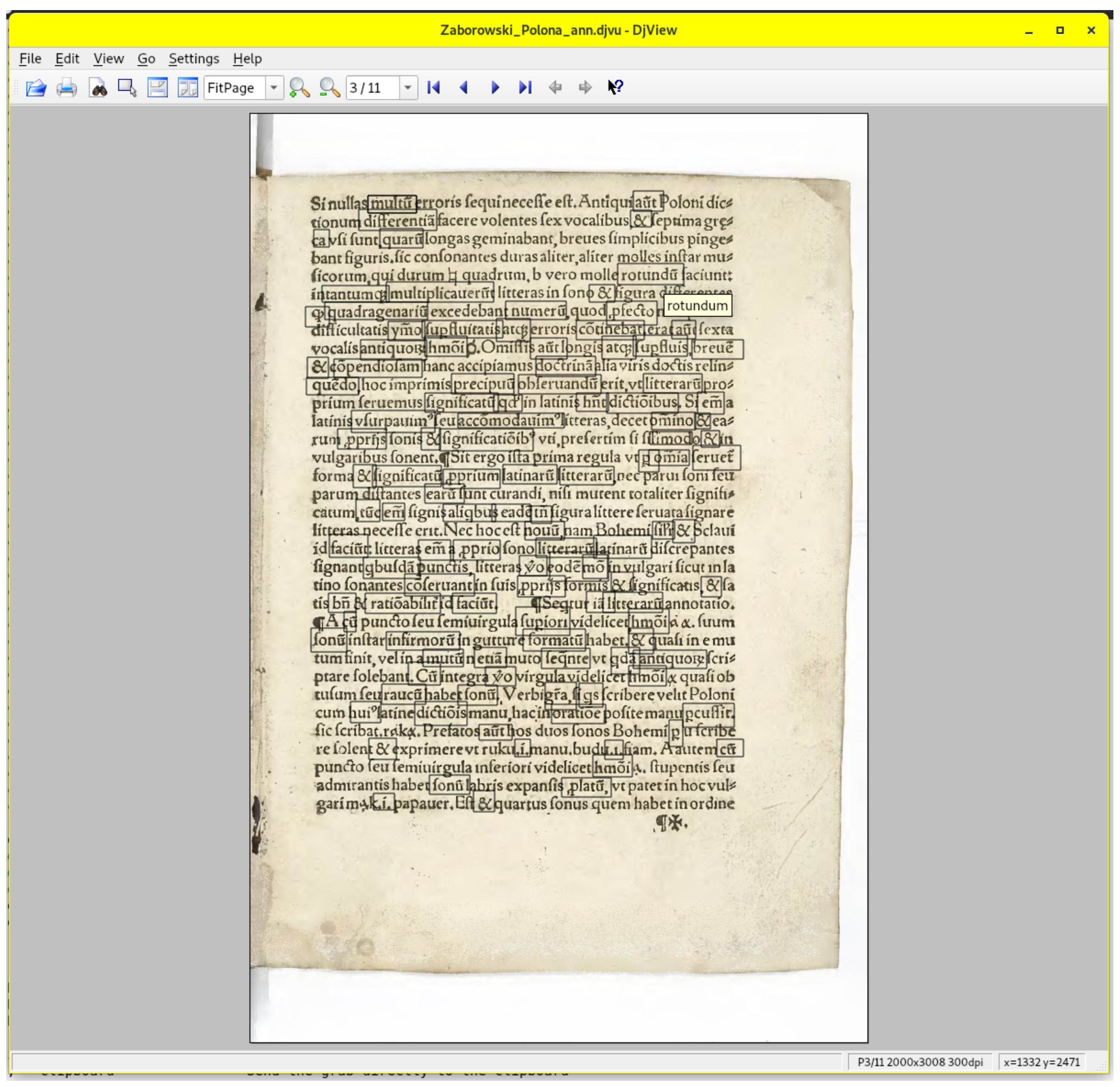

Figure 8: An index converted to the tooltips

4 Final remarks

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

Figure 9: A ligature type and its printed image (Daniel Ullrich, Threedots, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=855947)