A Workflow for HTR-Postprocessing, Labeling and Classifying Diachronic and Regional Variation in Pre-Modern Slavic Texts

Piroska Lendvai[1], **Maarten van Gompel**[2], **Anna Jouravel**[3], **Elena Renje**[3], **Uwe Reichel**[4,5], **Achim Rabus**[3], **Eckhart Arnold**[1]

[1]Dept. of Digital Humanities, Bavarian Academy of Sciences, Munich, Germany
[2]Digital Infrastructure, Humanities Cluster, Royal Dutch Academy of Sciences, The Netherlands
[3]Dept. of Slavic Languages and Literatures, University of Freiburg, Germany
[4]audEERING GmbH, Germany
[5]Hungarian Research Centre for Linguistics, Budapest, Hungary
{piroska.lendvai, eckhart.arnold}@badw.de
proycon@anaproy.nl
{anna.jouravel, elena.renje, achim.rabus}@slavistik.uni-freiburg.de
ureichel@audeering.com

Abstract

We describe ongoing work for developing a workflow for the applied use case of classifying diachronic and regional language variation in Pre-Modern Slavic texts. The data were obtained via handwritten text recognition (HTR) on medieval manuscripts and printings and partly by manual transcription. Our goal is to develop a workflow for such historical language data, covering HTR-postprocessing, annotating and classifying the digitized texts. We test and adapt existing language resources to fit the pipeline with low-barrier tooling, accessible for Humanists with limited experience in research data infrastructures, computational analysis or advanced methods of natural language processing (NLP). The workflow starts by addressing ground truth (GT) data creation for diagnosing and correcting HTR errors via string metrics and data-driven methods. On GT and on HTR data, we subsequently show classification results using transfer learning on sentence-level text snippets. Next, we report on our token-level data labeling efforts. Each step of the workflow is complemented with describing current limitations and our corresponding work in progress.

Keywords: historical text processing, diachronic and regional language variation, HTR-Postprocessing, data-driven error correction, transfer learning, text classification, semantic annotation, geolocalization, chronology attribution

1. Introduction

Situated in the field of historical natural language processing (NLP), our goal is to enable and scale up diachronic and variational linguistic research via facilitating the compilation of a large number of cleanly transcribed medieval texts, and running textual analyses on them. Our applied use case is the chronological and geolocational attribution of texts written in Church Slavic, predominantly in its East and South Slavic recensions. The development of our workflow includes the testing, adaptation and creation of computational language resources to this end.

We aim to generate specific resources for historical Slavic as input to the tools but the workflow can be applied to other historical languages as well, since the approaches used are generic or language-agnostic. Furthermore, the insights and classification results gained as workflow output will benefit research on cultural analytics, since our downstream classification tasks target the determination of text provenance, aiming to deliver information for tracing the emergence of individual manuscripts holding specific language data.

Historical texts are increasingly targeted by or incorporated in neural language models (Bamman and Burns, 2020; Gabay et al., 2022; Lendvai and Wick, 2022), including downstream tasks for attributing specific properties of text (Schweter et al., 2022; Liebeskind and Liebeskind, 2020), which are often designed as fine-tuning of BERT (Devlin et al., 2019), i.e., require labeled data. Nevertheless, the first major challenge for projects similar to ours is that attributing text characteristics in a supervised way would typically imply the classification of categories of language use, resp. of manuscript editing, for which there is little or no ground truth labeled data.

The second challenge originates in the morphological complexity of Church Slavic, e.g. suffixation, and that the medieval writers of these texts were often using multiple – equally correct – grapheme variants for one and the same phoneme or within one and the same morpheme. As a result, a *type* can have large number of orthographically non-standardized but philologically correct *token* variants that occur with low frequency, leading to data sparsity problems for Church Slavic NLP tools, such as Scherrer et al. (2018); Straka (2018); Qi et al. (2020); Besters-Dilger and Rabus (2021) where token variants may have been unseen in the training data, especially if those resources are

Manuscript	Century	Region	Place of Copying	Language	Main genre	Tokens	Unique tokens	Text snippets
Codex Suprasliensis	10-11	South	Eastern Old Bulgaria	Old Church Slavic; South Slavic recension	hagiographical-homiletic	65,207	18,450	4,831
Cyril of Jerusalem's Cathechetical Lectures	11-12	East	Kyivan Rus'	Old Church Slavic; South Slavic recension; Transmitted version used: East Slavic recension	dogmatic	62,011	20,936	4,282
Dionisio corpus (printed)	15-16	South	Serbia, Macedonia	Serbian Church Slavic; South Slavic recension	liturgical	142,402	42,828	10,685
Apostolos (from the Uspensky version of the Great Menaion Reader)	16	East	Muscovy	Russian Church Slavic; East Slavic recension	gospel	230,660	50,302	14,058
Sluzhabnik	18	South	Serbia	Serbian Church Slavic; South Slavic recension	liturgical	56,785	13,197	3,350
Elizabeth Bible (printed)	18	East	Muscovy	Russian Church Slavic; East Slavic recension	Bible translation	204,322	21,335	11,796
Methodius of Olympus: De lepra ad Sistelium	16	East	Kyivan Rus'	Old Church Slavic; Transmitted version: East Slavic recension	exegetic treatise	3,743	2002	259

Table 1: Overview and main characteristics of our ground truth (GT) text bodies, linked with online information.

small. This is why traditional dictionary-based methods but also neural modeling methods typically have limited performance on these data, starting from low-level NLP preprocessing tasks such as sentence splitting and tokenization, which propagates errors to the morphosyntactic level and beyond, likely impacting applied end tasks in a detrimental way.

Clearly, normalizing the texts – e.g. based on one of the Slavic orthographic norms – would undermine their character, eliminating a fair amount of orthographic variation that one would like to preserve, since such variation provides information that can be core to quantitative analysis and classification methods. Normalization would also entail using rules that cover known phenomena, but in our large dataset we likely need to take care of the unknown unknowns as well that have not yet been encountered and documented by the philological community.

Thirdly, text reuse from older sources or from sources written in a different language or recension is a regular phenomenon in Church Slavic texts. Parts of the text have been borrowed and usually integrated without any hints. It is not known if identification of the location and time where a particular text was copied might best be performed on paragraph or sentence-level, or even on the subsentential level. Linguistic research and quantification of phenomena related to diachronic and regional variation along these lines can likely only be scaled up if we are able to automatize the detection of the boundaries of such segments, for which, again, we have no labeled data.

To tackle these challenges, a global empirical question that we seek to answer in the long run is the extent to which variation in non-standardized historical texts, including irregularities originating in HTR errors, would impact NLP analysis and downstream, domain-specific classification tasks.

Related to automatic error correction, previous work includes neural approaches on OCR data, such as Lyu et al. (2021) and van Strien et al. (2020). Should we strive for automatic error correction on our medieval texts, a tool would need to take into account that a different subset of correction suggestions can be valid for texts that originate from different geographic regions – broadly differentating at least East Slavic texts as opposed to South Slavic texts –, and that correction suggestions need to be restricted to chronologically valid options. E.g. the grapheme variants in specific morphemes can differ by the region where a manuscript was copied. E.g. '-aa-' was typically used in Church Slavic recensions copied in southern parts of *Slavia Orthodoxa* vs. the so-called iotified variants of the second grapheme in this morpheme were typically used in Church Slavic recensions copied in its eastern parts.

This multi-faceted challenge is typically addressed by laborious manual processes in the Humanist community. To alleviate this, the contributions of the current submission are the following.

1. We report on the testing and adaptation of NLP infrastructure that we organize into a workflow for producing, diagnosing and classifying premodern Slavic HTR texts.

2. We aim to deliver digital awareness and skills to the philological community in terms of data representation methods such as XML and data enrichment methods such as manual entity tagging, as well as classical NLP methods such as metrics that express string similarity but also from recent advances in NLP, including deep learning.

3. We present the NLP community an applied use

case and work toward developing benchmark resources in a so far niche field of historical NLP, a.o. in terms of a dataset that encodes multiple layers of diachronic and regional language variation.

4. We describe all parts of the workflow as work in progress that yields pilot results but provides insights into the nature and complexities of such historical data, including the application of transfer learning for provenance classification tasks. So far we tackle and processed the so-called Ground Truth data, besides, we have a HTR-ed corpus of Church Slavic manuscripts that consists of 16+ million tokens (1.6 million unique). This is the data that we aim to clean up and process with the workflow that is being developed and discussed in the current paper.

2. Data Acquisition, Characteristics and Format

We selected our texts to encompass language varieties ranging from Old Church Slavic to pre-modern variants of its later versions; situated in the domain of religion and liturgy, they pertain to the written genre of non-vernacular language. The language variants exemplified below were formed under factors such as modernising tendencies that adapted to the vernacular usage at the geographic area where the texts got copied and compiled, but also archaizing trends at the turn of the 14th/15th centuries in certain Rus'ian literary schools, reintroducing specific linguistic properties characteristic of South Slavic texts. These impacts gave rise to to further individual orthographic, lexical and morphosyntactic changes.

We have worked with coarse-grained variety categorization, both in terms of time and in terms of location, as described below. Regarding region and time, the manuscripts range from medieval to early modern and can be roughly divided into three origin periods – 10th-14th, 15th-17th and 18th(-19th) centuries – in terms of linguistic development, based on codicological and paleographical aspects. The texts were copied and compiled in South-Eastern and Eastern Europe, therefore two geographic locations can be differentiated: South and East.

To transcribe those materials that had not yet been digitized, available HTR models from the Transkribus platform were used, including models for pre-modern East Slavic (Rabus, 2019) and South Slavic (Polomac, 2022). The bulk of our data had already been transcribed before the start of our project. We also experimented with eScriptorium and trained a public model for pre-modern Slavic (Rabus et al., 2023) but we found the transcription quality of the Transkribus models slightly better.

The source manuscripts and printings use Cyrillic script and non-standardized orthography, written in *scriptio continua* customary for that time, where spaces between words are used only occasionally and in an unsystematic way – some of them are assumed to mark breath pauses or to replicate syntagmatic units in the Greek source material from which they were translated. The HTR-ed texts do contain whitespace between words since the HTR model was trained to produce token segmentation. The resulting HTR material is not error free; should one want to obtain error free material, the output requires postprocessing, for which the first step is error detection.

In order to create resources for error detection, we need to determine the materials that will serve as ground truth (GT). To establish GT data, from the above two Transkribus models we obtained the GT data of their training sets and segmented these into separate bodies of texts[1]. In addition to publicly accessible resources (such as the printed Elizabeth Bible) we are compiling further texts from various sources, some of which are rarities that we will be able to publish for the first time, e.g. the patristic treatise *De Lepra* qualifies as GT data as we show below. An overview of all our GT text bodies is provided in Table 1. Our HTR body consists of texts contained in the *Great Menaion Reader*, which was compiled in Muscovy in the 16th century. These materials are rendered in two large text bodies consisting of several million tokens. They have been processed with Transkribus for the first time to create an HTR transcription and are the target of our work in progress.

2.1. Workflow Ecosystem

From the HTR process onward, our core data format was FoLiA XML (van Gompel et al., 2017; van Gompel, 2019), a data model and file format to represent digitized language resources enriched with linguistic annotation. FoLiA comes with a Python library, NLP tools and an annotation tool. Additional strong points for using FoLiA are its versatility when it comes to storing various versions of texts (such as normalized or corrected versions), label set agnostic annotation forms, and sophisticated, provenance-supporting correction mechanisms. We used the tools from the FoLiA ecosystem as docker containers or Python packages. The foliautils package provides converters for HTR export formats such as PAGE XML or plain text.

[1]Some of these texts (e.g. Codex Suprasliensis) also exist in several digital versions online, some with slight alphabet variations, not reflecting the manuscript's orthographical variation but the decision of the modern editors for a certain (at the time of the edition available?) font. Having inspected these versions, we concluded that our text seems to be closest to the manuscript we used.

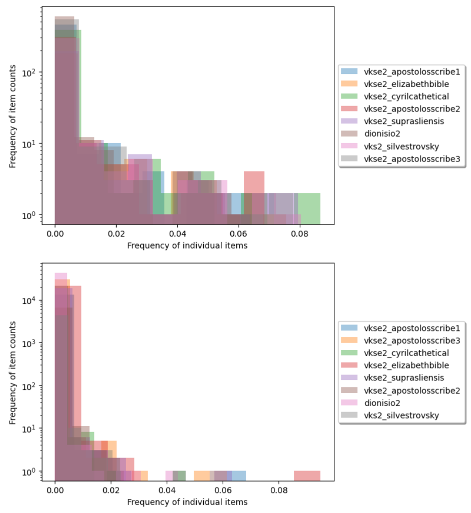


Figure 1: Histogram of character frequencies (top) and token frequencies (bottom) in some of the GT datasets. The bins on the x-axis express the proportion of occurrence of a given character resp. token.

3. Addressing HTR Errors

In order to characterize the distribution of token and unicode character occurrences across the GT datasets, we generated counts plots, cf. the histograms in Figure 1. The long tail distribution is particularly noticeable for tokens.

We are aware that some of the GT data from Transkribus reflects (perhaps idiosyncratic) transcription principles of individual editions. This introduced some more heterogeneity in terms of diacritic signs, glyph variants, superscript characters, and editorial addenda such as hyphens at the end of lines.

End-of-line hyphens are one of the several segmentation artifacts whose presence and absence is practical to be preserved along the workflow; note that these were added to the text by modern editors, e.g. in Besters-Dilger (2014), i.e. are not present in the original source text. Such hyphenation can be abundant since manuscripts often placed two columns of text on a single page making the lines short.

In the workflow we are able to address hyphenation at line endings systematically, since FoLiA keeps the provenance information regarding any changes in the document, a.o. joining token parts at line breaks, so that they are traceable throughout the ecosystem. FoLiA's converters (e.g. *FoLiA-txt*) take care of joining those words in the token layer of the XML representation which were line end hyphenated in the raw text layer.

Line end hyphenation removal may seem like a minor data normalization step that would have been achievable via a simple replacement script, but keeping track of the characteristics of various stages of text have proven important for us early on, as some of the simple solutions brought information loss that could only be manually reconstructed, making them expensive and less sustainable.

3.1. Diagnosing HTR

For data diagnostics and cleaning, we are preparing parallelized texts of GT and HTR. Currently we align texts on the line level. To this end, some steps need to be performed manually, e.g. to fix discrepancies such as superfluous lines in the Transkribus export originating in untreated layout recognition. An aligned GT-HTR dataset is being prepared on the basis of the manually corrected *De lepra* (Jouravel et al., 2024b) and another was constructed on the *Apostolos* (Besters-Dilger, 2014), based on which we sketched our anticipated error types. Upon inspecting the data, we could identify specific error types that go beyond (i) classical misrecognized words and (ii) incorrect token segmentation (i.e., falsely split or falsely joined characters) that are likely to be specific for the medieval Cyrillic data at hand. For example, (iii) superscript letter characters are very frequent in our texts, and they might be missing, misplaced (e.g. above the wrong base character), or wrongly recognized. Furthermore, (iv) character misrecognition might involve allographs.

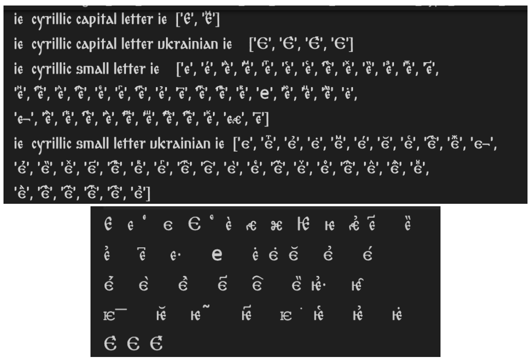
On the aligned texts, we computed the token recall (i.e., the number of shared tokens across GT and HTR, divided by the number of tokens in the GT), as well as the number of edit operations in the HTR with reference to the GT. For the *Apostolos* on 36,783 lines of text the mean token recall is 83.07 and the mean of edit operations is 1.4. Note that the score is high since this text is part of the training data for the HTR model; cf. Section 3.4.

3.2. The *sesdiff* Tool

Next, we analyzed the aligned line pairs via tools that work on the basis on string edit distance metrics. The *sesdiff* command line tool reads a two-column tab-separated input from standard input and computes the shortest edit script to go from the string in column A to the string in column B. It also computes the edit distance, aka Levenshtein distance. The output is in a simple format that is easily searchable for patterns. Besides, the tool allows to focus specifically on suffixes or prefixes, which is very practical for inflection-rich Slavic languages. We used *sesdiff* o.a. for filtering error types in the aligned data, e.g. to find false splits, i.e. words that were segmented in the HTR but not in the GT. The top area in Figure 3 illustrates such alignment in terms of Levenshtein and *sesdiff* notation.

3.3. The *analiticcl* Tool

The tool *analiticcl* is an approximate string matching or fuzzy-matching system that can be used for



ie cyrillic capital letter ie ['Є', 'Єⷨ']
ie cyrillic capital letter ukrainian ie ['Є', 'Є̆', 'Є̄', 'Є']
ie cyrillic small letter ie ['є', 'є́', 'єⷧ', 'єⷨ', 'єⷮ', 'єⷭ', 'єⷭ', 'єⷬ', 'єⷯ', 'є̏', 'єⷥ', 'єⷤ', 'є҃',
'єⷩ', 'єⷩ', 'єⷧ', 'єⷬ', 'єⷷ', 'єⷢ', 'єⷬ', 'є̉', 'є҃', 'єⷬ', 'єⷡ', 'єⷡ', 'e', 'єꙴ', 'єⷱ', 'єⷧ', 'є̇',
'є¬', 'єⷧ', 'єⷡ', 'єⷬ', 'єⷧ', 'єⷩ', 'єⷲ', 'єⷧ', 'єⷦ', 'єⷦ', 'єꙗ', 'є҃']
ie cyrillic small letter ukrainian ie ['є', 'єⷮ', 'є̉', 'є̇', 'єⷨ', 'є́', 'є̆', 'єⷭ', 'єⷬ', 'єⷤ', 'є¬',
'є̉', 'є̏', 'єⷯ', 'є҃', 'єⷡ', 'єⷡ', 'єⷢ', 'єⷬ', 'є҇', 'є̀', 'єꙴ', 'єⷬ', 'єꙋ', 'є̊', 'єⷬ', 'є̂', 'єⷧ', 'єⷹ',
'є̀', 'єⷬ', 'єⷬ', 'єⷮ', 'єⷬ', 'єⷧ']

Є	є ͤ	є	Є ͤ	є̀	ꙁє	ѥ	Ѥ	ѥ	ꙁє̉	є҃		є̏
є̉	є҃	є·	e	є̇	є̇	є̆	є̉	є́				
є̉	є̀	є̉	є҃	є҇	є̏	ѥ̉·	ѥ҇					
є¯	ѥ̆	ѥ̃	ѥ҃	є˙	ѥ̉	ѥ̉	ѥ̇					
Є	Є	Є										

Figure 2: Partial view of character diagnostics showing base letters in all seen variations (top); the corresponding part of the alphabet table for *analiticcl*, where we listed those characters that can be seen as possible variants of each other (bottom).

spelling correction or text normalization. Texts can be checked against a validated or corpus-derived lexicon (with or without frequency information) and the tool will return spelling variants for queries. Therefore, from each GT text we separately inferred a lexicon and fed it to *analiticcl*. Keeping the lexicon sources is practical as the tool will always return in which lexicon it finds a variant for a query.

analiticcl performs similarity computation to find variants based on a number of components: edit/Levenshtein distance, longest common substring, prefix match, suffix match and casing, making a log linear combination of its various components. The user needs to set the priority of these components in terms of weights. After manual trial-and-error observations of various parameter combinations on selected examples from our data, the setting that worked best was to prioritize longest common substring (0.6) over Levenshtein distance (0.4) and to disable the rest of the similarity components.

The distinguishing feature of *analiticcl* is the usage of the technique of anagram hashing that drastically reduces the variant search space and makes quick lookups possible even over larger edit distances. The underlying idea of this technique is largely derived from prior work of Reynaert (2011). *analiticcl* prunes token variants according to parameters that the user can set, such as score thresholds and maximum candidates to list, the maximum anagram distance and maximum edit distance between the query and the search candidate, which we were manually adjusting and set to 2 and 5, respectively.

Importantly, the system requires a table that lists character variant correspondences of the alphabet, i.e. grouped characters that can be seen as possible variants of each other. Such a grouping is needed for computing similarity distance metrics, in which the grouped characters are going to be considered as identical, cf. the bottom screenshot in Figure 2. The grapheme variant table that is fed to *analiticcl* was manually compiled based on automatically generated Unicode character combinations in the GT files (as illustrated by the top part of the figure). The grapheme variant table is in an evolving state, e.g. diacritical marks are still likely to be added. Despite the GT files designating texts from different geographical regions, for a first HTR diagnosis approach we worked with a single character correspondence table. Since character encoding and composition might have implications for similarity computation, input to the tool should be in Unicode Normal Form C ('Composed') to ensure that the combining diacritical marks and their targets constitute one codepoint whenever possible.

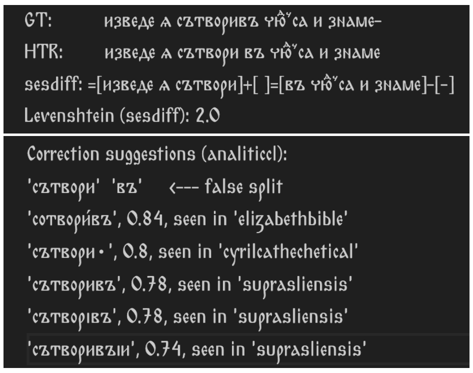
In the *analiticcl* system, all character substitutions based on the alphabet carry the same weight for the distance algorithm in the matching process. However, in practice certain characters might be more often confused by HTR than others, therefore, *analiticcl* allows for ingesting a list of known confusable patterns, based on which an extra rescoring can be performed, to give slight bonuses or penalties to the scores for specific confusables.

3.4. Analysing False Splits

The HTR material we diagnosed consisted of the *Apostolos* text and some parts of *De lepra* (cf. Table 1). Despite the GT of *Apostolos* – available from Besters-Dilger (2014) – being part of the Transkribus model that we employed for HTR, we run HTR on the scanned *Apostolos* manuscript again, in effect generating diagnostic data for iterative HTR improvement. Indeed, the resulting HTR of the *Apostolos* proved not to be error free, thus we could use this material for error diagnosis and correction experiments.

The script operating *analiticcl* generated a spreadsheet table for our domain experts so that they could inspect GT-HTR aligned lines in which the HTR had a falsely split word. The table contained the automatic correction suggestions by *analiticcl* for each of the false splits, so that domain specialists could observe and validate them. Out of 36,783 aligned text lines, 1,767 contained at least one false split based on the sesdiff pattern (4.8%). Out of these, for 962 analiticcl returned variants above the similarity threshold of 0.85 with the above parameter settings (54%); note that the falsely split words are often at the beginning or at the end of the line and are themselves fragments, i.e. not valid tokens but merely onsets or offsets of hyphenated words, for which the tool cannot find valid variants in the lexicon.

The bottom area in Figure 3 illustrates correction



```
GT:       изведе а сътворивъ чюⷣꙿса и знаме-
HTR:      изведе а сътвори въ чюⷣꙿса и знаме
sesdiff: =[изведе а сътвори]+[ ]=[въ чюⷣꙿса и знаме]-[-]
Levenshtein (sesdiff): 2.0
```

```
Correction suggestions (analiticcl):
'сътвори'  'въ'    <--- false split
'сотвори́въ', 0.84, seen in 'elizabethbible'
'сътвори • ', 0.8, seen in 'cyrilcathechetical'
'сътворивъ', 0.78, seen in 'suprasliensis'
'сътворiвъ', 0.78, seen in 'suprasliensis'
'сътворивъıи', 0.74, seen in 'suprasliensis'
```

Figure 3: One line of aligned GT and HTR. Top: *sesdiff* notation showing a false split, string similarity in terms of edit operations (Levenshtein). The HTR token recall rate for this string is 4/6. Bottom: correction suggestions from the *analiticcl* tool with internal similarity scores and evidence references to various lexicons.

suggestions from *analiticcl* for the exemplified false split in the top area. In the figure, we see from the aligned GT (first line) and HTR (second line) that the HTR engine correctly recognized all but one character in the manuscript line: the one following the superscript д in the 4th token. The correct character should be the combining diacritical mark of double grave accent, also known to Slavists as the *kendema*, but it was misrecognized as a *Cyrillic payerok* character (which indeed has a similar shape to the kendema in the handwritten text). In the example, the kendema that phonetically corresponds to the sound [iː], would form a ligature with the superscript д, so that the fourth token would be (when spelled out on the baseline, without superscript notation) чюдиса 'miracles'. But as the misread payerok (corresponding to the sound [ə]) is a supescript variant of the baseline grapheme ъ, this HTR error has led to a non-existing lexeme *чюдъса (*'mirucles', an imaginary corruptly spelled word).

Surprisingly, the character error went undetected in terms of string similarity since the GT itself exhibits the same error, i.e. wrongly uses the *payerok* character at the same position. Furthermore, as we pointed out above, hyphenation at the end of the GT string comes from modern editors, i.e. is not there in the medieval manuscript, so its absence from the HTR string may need to be evaluated in a way that is tailored to project-specific use cases. These observations call attention to the importance and difficulty of maintaining clean GT data that should be not only human readable but should additionally encode provenance information for textual amendments etc. in a way that allows to generate unbiased machine readable resources from it.

Regarding token segmentation, there is a false split in the HTR string сътвори въ, which chops off the past participle ending -въ from the verb. Note that both the string that resulted from the first part of the split – сътвори ('he created') –, as well as the second resulting part -въ (the preposition 'in') would be valid lexemes and frequently attested tokens.

For the false split, *analiticcl* provided plausible suggestions in three out of five cases. Suggestions 1 and 4 illustrate that the tool is able to operate with diacritical marks even though they are not (yet) part of the alphabet variant file. The correction suggested at rank 1 involves the character *Cyrillic small letter i + combining acute accent*, while the correction suggestion at rank 4 involves the character *Cyrillic small letter Byelorussian-Ukrainian i* that is indeed listed in the alphabet file as a variant. Interestingly, the top ranked correction suggestion corresponds to the East Slavic vocalized form, where ъ transforms into о, although the alphabet file does not explicitly list ъ and о as orthographic alternatives. The suggestions at rank 2 and rank 5 are not plausible.

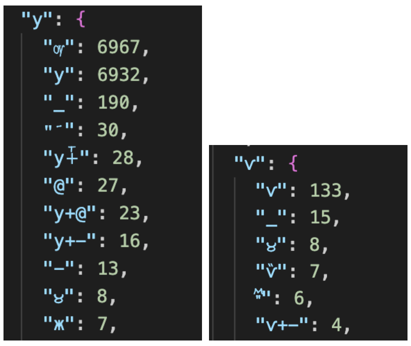
The current approach is, as detailed above, partly lexicon-based, and partly grapheme based and the aim of the diagnostic table is to identify a method primarily for the detection of HTR errors. The feedback for split reconstruction was that results look promising and many of the suggestions are plausible. We are going to expand the implementation of the approach and will quantify the results in follow-up reports.

3.5. The *TextAlign* tool

Error diagnostics can take place in a way that an edit cost function is learned from data. This is the working principle of the BAS TextAlign tool (Reichel, 2012) that is available as a web service (Kisler et al., 2017). By setting the cost parameter to 'intrinsic', costs are derived from smoothed conditional character co-occurrence probabilities which are subtracted from 1. The aligner allows for a uniform modeling of the three supported edit operations substitution, deletion and insertion. Furthermore, it can robustly align text sequences, regardless whether they were generated from same or different character inventories. Since the cost function relies on probabilities, its reliability increases with larger input text sizes.

Our goal by using the *TextAlign* webservice was manifold. First and foremost, we wanted to generate further diagnostic material. By postprocessing the tool's output we could extract a character co-occurrence statistics, which is important to assess grapheme-level HTR misrecognitions. Figure 4 displays grapheme co-occurrence statistics for two unicode characters.

For both, their dictionary shows the absolute co-occurrence counts of the focus character in the



```
"y": {
  "ꙋ": 6967,
  "y": 6932,
  "_": 190,
  "ˉ": 30,
  "y‡": 28,
  "@": 27,
  "y+@": 23,
  "y+—": 16,
  "—": 13,
  "ȣ": 8,
  "ж": 7,
```

```
"ѵ": {
  "ѵ": 133,
  "_": 15,
  "ȣ": 8,
  "ѷ": 7,
  ѿ": 6,
  "ѵ+—": 4,
```

Figure 4: Illustration of postprocessed *TextAlign* output: grapheme co-occurrence statistics. The dictionary shows the absolute co-occurrence counts for each character of the source text (GT) with the corresponding character (sequence) in the aligned text (HTR). For explanation see Section 3.5.

source text (GT) and the corresponding character (sequence) in the aligned text (HTR). Since *deletions* in the aligned text are marked by '_', we see e.g. that 'y' is replaced by '_', i.e. is deleted in the aligned text in 190 cases. Likewise, *insertions* are marked by '+' concatenations. E.g. 'y' aligns to 'y+@' in 23 cases; since '@' is a placeholder for whitespace in the notation, this means whitespace insertion in the HTR text. For both target characters we can observe co-occurrence with their (phonetically motivated) alternative graphemes in the HTR, but also what seems like character replacements based grapheme shape similarity.

The output of the tool can be used to make the alphabet variant table for *analiticcl* as complete as possible, and can also be assistive in preparing a character confusables list to serve as future input to the *analiticcl* tool.

3.6. HTR Diagnostics: Strengths and Limitations

It is important to appreciate the reuse of shared tools in the community, since this mitigates duplicated work and at the same time can grow the resilience of such tools since they are getting tested on unseen data and new use cases, which is indeed one of the goals of several past and ongoing national and international language research infrastructure projects. Nevertheless it is also important to see that one must not underestimate the effort needed to spend on interfacing these resources with one's own use case. For instance, tools require understanding about their installation and access methods, requirements for input and output data formatting, structuring, interpretation, and so on.

We note that our aligned dataset is currently small and not yet representative, making it suboptimal for the tools. We focused on false splits (i.e., erroneous segmentation), since these are more difficult to treat than HTR misrecognitions where token boundary segmentation is intact. False joins are likely even more difficult to address. Our domain experts found that for the obvious false splits, automatic correction would take less time than manual. For the remaining cases, in order to obtain gold standard corrections, a human would likely still need to check the larger context of the given manuscript to approve an automatic correction.

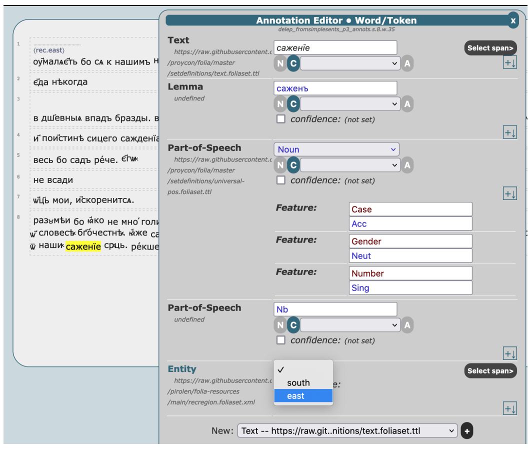
Line-level error detection is clearly suboptimal for a partly lexicon-based approach, since the line might begin as well as end with a non-word. We are going to use another way unitizing the text, i.e. segmentation on the text snippet level. We query *analiticcl*'s variant model with an exact input string and set it to correct it as a single unit. So far we did not make use of the tool's detection aspect that automatically determines which part of the input needs correction, which we will address in future work.

4. Classification of Diachronic and Regional Variation

In a subsequent component of the workflow, we set out to investigate the extent to which BERT (Devlin et al., 2019) can be used for text classification tasks that characterise temporal-spatial dimensions of our data. The first six lines in Table 1 designate the GT data that we used for these deep learning experiments; for details cf. Lendvai et al. (2023).

Recasting this task for machine learning is not trivial; we took over the document-level annotation for creating the ground truth labeling on the text snippet level, that determining the manuscript copying time (granularity: century, binned into three labels: '10-12', '15-16' and '18') and geographic region (granularity: language region, currently binary). We segmented the texts into sentence-like units (text snippets) using the Stanza sentence segmenter, with the language model set to 'Old Church Slavonic', which was a crude first approach, as we illustrate in Jouravel et al. (2024a).

In a matrix of experiments, we compared direct finetuning of the base models from the Hugging Face repository on the downstream tasks vs. domain adapting a model and its subsequent finetuning. Domain adaptation was realized by means of vocabulary extension of the tokenizers of the pretrained BERT models, and subsequent adaptation on the training partitions of the documents in a masked language modeling task. For each of the downstream tasks we finetuned the off-the-shelf as well as the domain-adapted variants of the three base BERT models in the same way: adding a classification head to the base model consisting of one feed-forward hidden layer with a *tanh* activation function, and a final output projection layer to the


1 ⟨rec.east⟩
оу҆малѧє҆ть бо сѧ к нашимъ
2 є҆̂да нѣкогда
3
в дш҃евныѧ впадъ бразды.
4 и҆ поѝстинѣ сицего саждені
5 весь бо садъ рéче. є҆г҃лѧ
6 не всади
7 ѿц҃ь мои, и҆скоренитсѧ.
8 разꙋмѣи бо ꙗ҆ко не мно́голи
ѿ словесѣ бг҃о́честнѣ. ꙗ҆же
ѿ нашиⷯ саженїе срцⷭ҇ь. рéкше

Annotation Editor • Word/Token
delep_fromsimplesents_p3_annots.s.8.w.35

Text https://raw.githubusercontent.c /proycon/folia/master /setdefinitions/text.foliaset.ttl	саженїе
Lemma undefined	саженъ confidence: (not set)
Part-of-Speech https://raw.githubusercontent.c /proycon/folia/master /setdefinitions/universal-pos.foliaset.ttl	Noun confidence: (not set) Feature: Case Acc Feature: Gender Neut Feature: Number Sing
Part-of-Speech undefined	Nb confidence: (not set)
Entity https://raw.githubusercontent.c /pirolen/folia-resources /main/recregion.foliaset.xml	✓ south east

New: Text -- https://raw.git..nitions/text.foliaset.ttl

Figure 5: Adding a token-level entity annotation for *region* in FLAT to the text of *De lepra*. The part-of-speech and lemma annotations shown in the Annotation Editor box were assigned to the token by FLAT by ingesting the corresponding values from the (suboptimal) UDPipe analysis output.

respective number of classes.
Input to this head was the mean pooling over the hidden states of the last encoder layer to which we applied a dropout of 0.1. Model finetuning was done in 4 epochs with a learning rate of $3e-5$, the AdamW optimizer with a weighted Cross Entropy loss, and a batch size of 16. We did not freeze the encoder layers and kept the best model in terms of Unweighted Average Recall (UAR) on the development set and evaluated it on the held-out test set. All pretrained models were outperformed by their domain-adapted variants and reached overall high performance on the downstream classification tasks.

4.1. *BERT* Experiments: Strengths and Limitations

Our downstream tasks had coarse-grained classes that we could generate from the manuscript level. Since we lack ground truth provenance labels attributed on sub-manuscript level, we seek possibilities that would enable us to point out phonological, morphological, etc. characteristics that correlate with the task to classify diachronic and regional variation. One such possibility would be to make our task setup more granular so that classes can relate to the token and to the character levels.
Our current goal was to investigate the extent to which a generic BERT approach on the level of text snippets, i.e. semantically-oriented, would make use of the heterogeneous data characteristics. It is left for future work to thoroughly examine to what extent the textual content was contributing to the performance scores and whether tokenization within the language model can be geared toward grammatically meaningful word pieces that would be characteristic for surface textual properties.
Besides experiments on the GT data, we plan to examine the impact of errors and that of error correction by measuring BERT's performance directly on the downstream tasks; preliminary results obtained on our uncorrected HTR texts point out a drop in performance, but the data points are orders of magnitude larger (ca. 900k text snippets, vs. ca. 5k in the current experiments) thus such a bird's-eye-view comparison is statistically questionable.

5. Domain-Specific Labeling

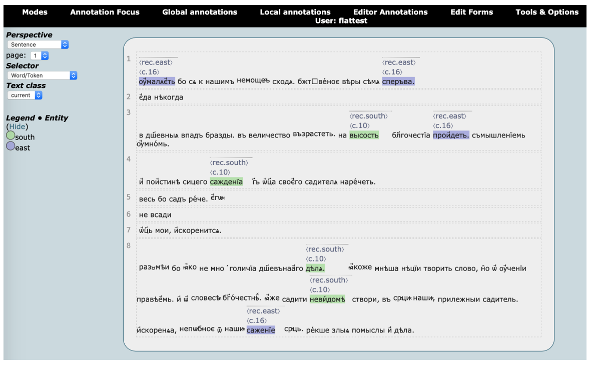
In this section, we introduce our efforts for the creation of token-level labeling.

5.1. The *FLAT* Annotation Tool

The *FLAT* Annotation Tool is a web-based linguistic annotation environment for data markup in a flexibly configurable way. The annotated data is stored in FoLiA XML format. *FLAT* also allows to a lesser extent for direct data analysis, using its built-in corpus search tool. We installed *FLAT* as a docker container service on a server so that multiple domain specialists are able to access the annotation interface and collaboratively add markup to the texts.
In order to allow token-based annotations in *FLAT*, the text needs to be token segmented. We segmented the text into sentences and tokens by using the morphosyntactic parser *UDPipe* (Straka, 2018) via its REST API, setting its analysis model to *Old Church Slavonic PROIEL*. Conveniently, *FLAT* can ingest data in several formats besides FoLiA XML, a.o. the CoNLL-U format that *UDPipe* produces.
In Figure 5 and Figure 6 one can observe that UDPipe makes several errors, both in morphosyntactic analysis and in sentence segmentation. We are investigating the latter in our recent study (Jouravel et al., 2024a) and are currently working on improving the sentence segmentation automatically.

5.2. Annotation Scheme Development

FLAT lets end users configure annotation schemes. We started with a scheme that corresponds to the downstream tasks: binary *region* (or: recension) annotations and multi-value *century* annotations. Figure 5 shows the process of adding a token annotation in the editor in the browser. The part-of-speech and lemma annotations were assigned by FLAT from the corresponding values from the UDPipe analysis, and can hold errors. The editor allows for hand-correcting these. Sentence segmentation for the displayed data is performed by UDPipe too and is not fully correct, e.g. snippets can have syntactically unmotivated boundaries and length.
As a result, Figure 6 shows how token-level annotations are rendered in FLAT for the *century* and



🟩 south
🟪 east

1 ⟨rec.east⟩ ⟨c.16⟩ оу҆малѧ́єть бо сѧ к нашимъ немощемъ схода. бжт□ве́ноє вѣры сьмѧ ⟨rec.east⟩ ⟨c.16⟩ сперьва.

2 є҆́да нѣкогда

3 в дш҃евныѧ впадъ бразды. въ величество възрастеть. на ⟨rec.south⟩ ⟨c.10⟩ высость бл҃гочестїа ⟨rec.east⟩ ⟨c.16⟩ пройдеть. съмышленїемь оу҆мно́мь.

4 и҆ поистинѣ сицего ⟨rec.south⟩ ⟨c.10⟩ сажденїа г҃ь ѿц҃а своє҆го садителѧ наре́четь.

5 весь бо садъ ре́че. є҆гѡ

6 не всади

7 ѿц҃ь мои, и҆скоренитсѧ.

8 разѹмѣи бо ꙗ҆ко не мно ́голичїа дш҃евънаа҆го ⟨rec.south⟩ ⟨c.10⟩ дѣлѧ. ꙗ҆коже мнѣша нѣцїи творить слово, ꙗ҆о ѿ оу҆ченїи правѣє́мь. и҆ ѿ словесъ бл҃го́честнѣ. ꙗ҆же садити ⟨rec.south⟩ ⟨c.10⟩ неви́домѣ створи, въ срц҃ихъ нашихъ прилежныи садитель. и҆скоренаа, непѡбноє ѿ нашихъ ⟨rec.east⟩ ⟨c.16⟩ саженїє срц҃ь. ре́кше злыѧ помыслы и҆ дѣла.

Figure 6: Token-level annotations in FLAT for the entity types *region* and *century*. Labels are rendered above the corresponding token(s) (e.g. 'c.16' for 16th century), as well as shown on the side in terms of color codes for *region*.

region entities. The labels appear above the corresponding tokens and are also color coded. Note that the two entity annotation layers are technically independent of each other in the XML source, thus we can set values for *century* independent of *region*, and *century* labels do not need to be binned, allowing for finer-grained classification. We will continue developing the annotation scheme based on upcoming pilot experiments.

6. Summary and Conclusions

We aimed to inform the NLP and Digital Humanities communities about our effort for tackling the complex problem of provenance attribution for historical language data.

We presented a prototype workflow with the goal of scaling up the potentials of historical Slavic Studies. Since even advanced HTR models are producing errors, we described how NLP tools can be used (a) to diagnose HTR output as a feedback for recognition engines and (b) to utilize data-driven resources for correcting HTR errors.

Our workflow prototype includes several approaches and an applied end task. The pipeline starts with presenting its ecosystem and discusses data acquisition. Next, it explores HTR error correction approaches. After this outlook it takes three concrete tools and applies them on our data. The pipeline then leads to classification experiments with BERT for the applied end task. Afterwards, we discuss an annotation tool that would benefit the shortcomings of the previous components of the pipeline.

Our paper's scope goes beyond the presentation of HTR error correction and its evaluation. The latter is currently small scale, since our goal was to describe the pipeline with several components of a workflow that we have been testing for Church Slavic. The primary goal of our HTR correction effort is, for the time being, not to diagnose HTR engines but to get started on material to improve the HTR output created earlier. It is often an issue in cultural heritage projects that the source materials cannot be retro-digitalized but need to be cleaned. The development of these tools enables reproducibility, which is completely missing when only manual corrections are applied: Tools can be rerun at any time, or can be adjusted and rerun, so that correction cycles are repeatable and thus reproducible. Tools are also parameter-configurable, so that one can run them with turning certain knowledge components on and off, leading to studies that can observe the impact of different knowledge components on the tools' results.

These tools also allow human domain specialists to transform their expert knowledge into reusable resources that the tools ingest (e.g. grapheme correspondence tables), aggregating and representing, i.e. formalizing such knowledge in a sustainable, machine-readable way. Especially since the tools used are generic and freely available, the approaches of the described workflow can be applicable to other languages.

References

David Bamman and Patrick J. Burns. 2020. Latin BERT: A contextual language model for classical philology. *arXiv*, 2009.10053.

Juliane Besters-Dilger, editor. 2014. *Kommentierter Apostolos. Volume 1.* Otto Sagner, Freiburg i. Br.

Juliane Besters-Dilger and Achim Rabus. 2021. Neural morphological tagging for Slavic: strengths and weaknesses. *Scripta & e-Scripta, The Journal of Interdisciplinary Mediaeval Studies*, 21:79–92.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Simon Gabay, Pedro Ortiz Suarez, Alexandre Bartz, Alix Chagué, Rachel Bawden, Philippe Gambette, and Benoit Sagot. 2022. From FreEM to D'AlemBERT: a large corpus and a language model for early modern French.

Anna Jouravel, Elena Renje, Piroska Lendvai, and Achim Rabus. 2024a. Assessing Automatic Sentence Segmentation in Medieval Slavic Texts. In *Proc. of the Digital Humanities 2024 Conference, 6-9 August, 2024, Washington, DC, USA*.

Anna Jouravel, Janina Sieber, and Katharina Bracht. 2024b. *Methodius von Olympus: De lepra. Griechischer und slavischer Text, mit Einleitung und deutscher Übersetzung.*, volume NF, 31 of *Die griechischen christlichen Schriftsteller der ersten drei Jahrhunderte*. De Gruyter, Berlin.

Thomas. Kisler, Uwe Reichel, and Florian Schiel. 2017. Multilingual processing of speech via web services. *Computer, Speech, and Language*, 45(C):326–347.

Piroska Lendvai, Uwe Reichel, Anna Jouravel, Achim Rabus, and Elena Renje. 2023. Domain-Adapting BERT for Attributing Manuscript, Century and Region in Pre-Modern Slavic Texts. In *Proceedings of the 4th International Workshop on Computational Approaches to Historical Language Change 2023 (LChange'23) co-located with EMNLP2023, Singapore*.

Piroska Lendvai and Claudia Wick. 2022. Finetuning Latin BERT for Word Sense Disambiguation on the Thesaurus Linguae Latinae. In *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, pages 37–41, Taipei, Taiwan. Association for Computational Linguistics.

Chaya Liebeskind and Shmuel Liebeskind. 2020. Deep learning for period classification of historical hebrew texts. *Journal of Data Mining & Digital Humanities*.

Lijun Lyu, Maria Koutraki, Martin Krickl, and Besnik Fetahu. 2021. Neural OCR Post-Hoc Correction of Historical Corpora. *Transactions of the Association for Computational Linguistics*, 9:479–493.

Vladimir Polomac. 2022. Serbian Early Printed Books: Towards Generic Model for Automatic Text Recognition using Transkribus. In *Proceedings of the Conference on Language Technologies & Digital Humanities*, Ljubljana, Slovenia.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Achim Rabus. 2019. Recognizing Handwritten Text in Slavic Manuscripts: a Neural-Network Approach Using Transkribus. *Scripta & e-Scripta, The Journal of Interdisciplinary Mediaeval Studies*, 19:9–32.

Achim Rabus, Walker Riggs Thompson, and Daniel Stökl Ben Ezra. 2023. Generic HTR model for Old Cyrillic uncial and semi-uncial script styles (11th-16th c.). DOI 10.5281/zenodo.7755483.

Uwe D. Reichel. 2012. PermA and Balloon: Tools for string alignment and text processing. In *Proc. Interspeech*, Portland, Oregon, USA.

Martin Reynaert. 2011. Character confusion versus focus word-based correction of spelling and OCR variants in corpora. *International Journal on Document Analysis and Recognition*, 14.

Yves Scherrer, Susanne Mocken, and Achim Rabus. 2018. New Developments in Tagging Pre-modern Orthodox Slavic Texts. *Scripta & e-Scripta, The Journal of Interdisciplinary Mediaeval Studies*, 18:9–34.

Stefan Schweter, Luisa März, Katharina Schmid, and Erion Cano. 2022. hmBERT: Historical multilingual language models for named entity recognition. *arXiv*, 2205.15575.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Maarten van Gompel. 2019. FoLiA: Format for Linguistic Annotation. Documentation.

Maarten van Gompel, Ko van der Sloot, Martin Reynaert, and Antal van den Bosch. 2017. FoLiA in Practice: The Infrastructure of a Linguistic Annotation Format. *J. Odijk & A. van Hessen (Eds.), CLARIN in the Low Countries*, pages 71–82.

Daniel van Strien, Kaspar Beelen, Mariona Coll Ardanuy, Kasra Hosseini, Barbara McGillivray, and Giovanni Colavizza. 2020. Assessing the impact of OCR quality on downstream NLP tasks. In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, volume 1.