Making Historical Latvian Texts More Intelligible to Contemporary Readers

Lauma Pretkalniņa, Pēteris Paikens, Normunds Grūzītis, Laura Rituma
Institute of Mathematics and Computer Science, University of Latvia
Raiņa blvd. 29, LV-1459, Riga, Latvia
E-mail: lauma@ailab.lv, peteris@ailab.lv, normundsg@ailab.lv, laura.rituma@gmail.com

Abstract

In this paper we describe an ongoing work developing a system (a set of web-services) for transliterating the Gothic-based Fraktur script of historical Latvian to the Latin-based script of contemporary Latvian. Currently the system consists of two main components: a generic transliteration engine that can be customized with alternative sets of rules, and a wide coverage explanatory dictionary of Latvian. The transliteration service also deals with correction of typical OCR errors and uses a morphological analyzer of contemporary Latvian to acquire lemmas — potential headwords in the dictionary. The system is being developed for the National Library of Latvia in order to support advanced reading aids in the web-interfaces of their digital collections.

1. Introduction

In 2010, a mass digitalization of books and periodicals published from the 18th century to the year 2008 was started at the National Library of Latvia (Zogla and Skilters, 2010). This has created a valuable language resource that needs to be properly processed in order to achieve its full potential and accessibility to a wide audience, especially in the case of historical texts.

A fundamental issue in a massive digitalization of historical texts is the optical character recognition (OCR) accuracy that affects all the further processing steps. The experience of Tanner et al. (2009) shows that only about 70–80% of correctly recognized words can be expected in the case of the 19th century English newspapers. The OCR accuracy achieved in the digitalization project of the National Library of Latvia (NLL) is not known to the authors yet, however, in the case of historical Latvian, at least two more obstacles have to be taken into account: the Gothic-based Fraktur script (that differs from the Fraktur used in historical German) in contrast to the Latin-based script that is used nowadays, and the inconsistent use of graphemes over time.

During the first half of the 20th century, the Latvian orthography has undergone major changes and has acquired its current form only in 1957[1]. The Fraktur script used in texts printed as late as 1936 is not familiar to most readers of contemporary generation. Moreover, the same phonemes are often represented by different graphemes, even among different publishers of the same period. The Latvian lexicon, of course, has also changed over time, and many words are not widely used and known anymore.

This makes a substantial obstacle in the accessibility of Latvian cultural heritage, as almost all pre-1940 printed texts currently are not accessible to contemporary readers in an easily intelligible form.

In this paper we describe a recently developed system for transliterating and explaining tokens (on a user request) in various types of historical Latvian texts.

In the following chapters, we first give a brief introduction to the evolution of the Latvian orthography, and then we describe the design and implementation of the system that aims to eliminate the accessibility issues (to a certain extent). We also illustrate some use-cases that hopefully will facilitate the use of the Latvian cultural heritage.

[1] http://en.wikipedia.org/wiki/Latvian_language#Orthography

2. Latvian orthography

The first printed works in Latvian appeared in the 16th century. Until the 18th century the spelling was highly inconsistent, differing for each printed work. Since the 18th century a set of relatively stable principles has emerged, based on the German orthography adapted to represent the Latvian phonetic features (Ozols, 1965).

In 1870-ies, with the rise of nationalism, there were first activities to develop a new orthography that would be more appropriate to describe the sounds used in Latvian: long vowels, diphthongs, affricates, fricatives and palatalized consonants (Paegle, 2001). This goes hand in hand with the slow migration from the Fraktur script to the Latin script. The ultimate result of these efforts was an alphabet that in almost all cases has a convenient one-to-one mapping between letters and phonemes, and is almost the same as the modern Latvian alphabet that consists of 33 letters. However, the adoption of these changes was slow and inconsistent, and both scripts were used in parallel for a prolonged time (Paegle, 2008). From around 1923, Latvian books are mostly printed in the Latin script, but many newspapers still kept using the Fraktur script due to investments in the printing equipment.

There were additional changes introduced in the modern orthography in 1957, eliminating the use of graphemes ‘ch’ and ‘ŗ’, and changing the spelling of many foreign words to imitate their pronunciation in Russian. This once again resulted in decades of parallel orthographies: texts printed in USSR use the new spelling while texts published in exile resist these changes.

This presents a great challenge, as the major orthographic changes have occurred relatively late and, thus, a huge proportion of Latvian printed texts have been published in obsolete orthographies. Furthermore, the available linguistic resources and tools, such as

dictionaries and morphological analyzers, do not support the historical Latvian orthography.
Figure 1 illustrates some of the issues that have to be faced in the processing pipeline if one would semi-automatically convert a text in Fraktur into the modern Latvian orthography. It should be mentioned that, in the scope of this project, OCR is provided by a custom edition of ABBYY FineReader (Zogla and Skilters, 2010).

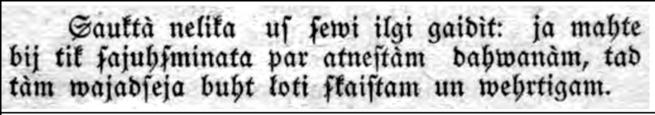
The original facsimile (the old Fraktur orthography):
 Sauktà nelika uſ ſewi ilgi gaidit: ja mahte bij tik ſajuhſminata par atneſtàm dahwanàm, tad tàm wajadſeja buht loti ſkaiſtam un wehrtigam.
The actual result of OCR:
Sauktà nelika us sewi ilgi gaidît: ja mahte bij tik sajuhsminata par atnestām dahroanàm, tad tàm roajadseja buht ļoti skaistam un wehrtigam.
The expected result of OCR (the old Latin orthography):
Sauktā nelika uz sewi ilgi gaidīt: ja mahte bij tik sajuhsminata par atnestām dahwanām, tad tām wajadzeja buht ļoti skaistam un wehrtigam.
Transliteration into the modern orthography:
Sauktā nelika uz sevi ilgi gaidīt: ja māte bija tik sajūsmināta par atnestām dāvanām, tad tām vajadzēja būt ļoti skaistām un vērtīgām.

Figure 1: A sample sentence in the historical Latvian orthography and its counterpart in the modern orthography along with intermediate representations.

3. Transliteration engine

We have developed a rule-based engine for performing transliterations and correcting common OCR errors. In this chapter we describe the engine assuming that rules defining the transliteration and error correction are already provided. Some guidelines on how to acquire a set of period- and/or publisher-specific rules are briefly discussed in the last section of this chapter.
To satisfy the user interface requirements[2], the engine is designed to process a single token at a time. The workflow can be described as follows:

- The input data is a single word (in general, an inflected form).
- Find all transliteration rules that might be applied to the given word and apply them in all the possible combinations (thus acquiring potentially exponential amount of variants).
- Find the potential lemmas for the transliteration variants using a morphological analyzer of the contemporary language (Paikens, 2007).
- Verify the obtained lemmas against large, authoritative wordlists containing valid Latvian words (in the modern orthography) of various domains and styles, as well as of regional and historical lexicons.
- Assign a credibility level to each of the proposed variants according to the transliteration and validation results. In an optional step, the transliteration variants (both word forms and lemmas) can be ranked according to their frequency in a text corpus.

Note that the contextual disambiguation of the final variants (if more than one) is left to the reader.
Below we shall describe most significant parts of the workflow in more depth.

3.1 Types of transliteration rules

Our transliteration engine uses two types of rules: obligatory and optional. The obligatory rules describe reliable patterns (usually for the standard transliteration, but also for common OCR error correction) that are always applied to the given word, assuming that in practice they will produce mistakes only in rare cases. When this set of rules is applied to a target string, only one replacement string is returned (except cases when a target string is a substring[3] of another target string; see Figure 2: 'tsch' vs. 'sch').
The optional rules describe less reliable patterns (usually for OCR correction, but also for transliteration) that should be applied often, but not always. I.e., the optional rules produce additional variants apart from the imposed ones (by the obligatory rules). When a set of optional rules is applied, it is allowed to return more than one replacement string for a given target string.
All rules are applied "simultaneously", and the same target string can be matched by both types of rules (e.g. a standard transliteration rule is that the letter 'w' is replaced by 'v', however, the Fraktur letter 'm' is often mistakenly recognized as 'w').
Figure 2 illustrates various rules of both types (some of them are applied to acquire the final transliteration in Figure 1). Note that OCR errors are corrected directly into the modern orthography (e.g. 'ro' is transformed into 'v' instead of 'w').

```
<rules>
  <obligatory>
    <str find="à" replace="ā"/>
    <str find="ah" replace="ā"/>
    <str find="w" replace="v"/>
    <str find="tsch" replace="č"/>
    <str find="sch" replace="š"/>
    <str find="ees" replace="ies" match="end"/>
  </obligatory>
  <optional>
    <str find="ro" replace="v"/>
    <str find="a" replace="ā"/>
    <str find="l" sensitive="yes">
      <replace>I</replace>
      <replace>J</replace>
    </str>
  </optional>
</rules>
```

Figure 2: A set of sample transliteration rules.

[2] The system will provide back-end services for reading aids (in a form of pop-up menus) in the web-interfaces of the NLL digital collections.

[3] The longest substring not necessarily is the preferable one.

For any rule it is possible to add additional requirements that it is applied only if the target string matches the beginning or the end of a word, or an entire word, and/or that the rule is case-sensitive.

Transliteration rules are provided to the engine via an external XML file. The current implementation of the engine allows providing several alternative rule sets (e.g. one for the 17th century books and another for the 19th century magazines). An appropriate set of rules is chosen automatically, based on the document's metadata: publication year and typeface.

3.2 Applying transliteration rules

When the transliteration engine is started, each set of rules is loaded into the memory and is stored in a hash map using the target strings as keys. This gives us the ability to access all the possible replacements for a given target string in effectively constant time[4].

Transformations are performed with the help of dynamic programming and memorization. Each token is processed by moving the cursor character by character from the beginning to the end. In each position we check if characters to the left from the cursor correspond to some target string. In an additional data structure we keep all transformation variants for the first character, for the first two characters, for the first three characters etc. The transformation variants for the first *i* characters are formed as follows (consult Figure 3 for an example):

- For every rule whose target string matches the characters from the *k*-th position till the *i*-th position, a transformation variant (for the *i*-th step) is formed by concatenating each transformation variant from the *k*-th step with the rule's replacement string.
- From each transformation variant in length *i*-1 form a transformation variant in length *i* by adding the *i*-th character from the original token if there is no obligatory rule with a target string matching the last character(s) to the left from the cursor.

When the cursor reaches the end of the string, the obtained transformation variants are sorted in two categories: "more trusted" variants that are produced by the obligatory rules only, and "less trusted" variants that are produced also by the optional rules.

Input: dahroanàm			
Step 1:	d	Step 5:	dāro, dāv
Step 2:	da, dā	Step 6:	dāroa, dāva, dāroā, dāvā
Step 3:	dā, dɑ̄	Step 7:	dāroan, dāvan, dāroān, dāvān
Step 4:	dār	Step 8:	dāroanā, dāvanā, dāroānā, dāvānā
Output (Step 9): dāroanām, dāvanām, dāroānām, dāvānām			

Figure 3: Sample application of transliteration rules. The input comes from Fig. 1 (line 2, token 6). Consult Fig. 2 for the rules applied (producing the underlined strings).

In Figure 3, it turns out that "dāroanām" is a more trusted variant than "dāvanām", although actually it is vice versa. The false positive variant is eliminated in the next processing step, while the other one is kept (see Section 3.3).

To speed up the transliteration, it is possible for user to instruct the engine not to use the optional rules for the current token.

3.3 Verifying transliteration variants

If transliteration is performed in the way it is described in the previous section, it produces plenty of nonsense alternatives. Thus we need a technique to estimate which of the provided results is more credible. One such estimate is implicitly given by the differentiation between obligatory and optional rules.

Another way to deal with this problem is to obtain a large list of known valid words and check the transliteration variants against it. Typically these would be lists of headwords from various dictionaries, however, due to the rich morphological complexity of Latvian, word lists, in general, are not very usable in a straightforward manner, but we can use a morphological analyzer to obtain the potential lemmas for the acquired transformation variants.

The exploited analyzer (Paikens, 2007) is based on a modern and rather modest lexicon (~60 000 lexemes) — although a lot of frequently used words are the same in both modern and historical Latvian, there is still a large portion of words out of vocabulary. Therefore we use a suffix-based guessing feature of the analyzer to extend its coverage when the lexicon-based analysis fails.

Transliteration variants whose lemmas are found in a list of known words are considered more credible. Currently we use wordlists from two large Latvian on-line dictionaries: one that primarily covers the modern lexicon (~190 000 words, including regional words and proper names), and one that covers the historical lexicon (>100 000 words, manually transliterated in the modern orthography). To extend the support for proper names (surnames and toponyms), we also use the Onomastica-Copernicus lexicon[5].

In the whole transliteration process we end up with four general credibility groups for the transliteration variants:

1. only the obligatory rules have been applied; at least one lemma is found in a dictionary;
2. at least one optional rule has been applied; at least one lemma is found in a dictionary;
3. only the obligatory rules have been applied; no lemmas could be verified by a dictionary;
4. at least one optional rule has been applied; no lemmas could be verified by a dictionary.

For instance, if we take the variants from Figure 3, "dāroanām" is not found in the morphological lexicon and by guessing it might be lemmatized as "dāroana" (noun) or "dāroant" (verb) — none of these nonsense words can be found in a dictionary. However, "dāvanām"

[4] This is important for the future use-cases where the service will provide probabilistic full-text transliteration.

[5] http://catalog.elra.info/product_info.php?products_id=437

is both recognized by the morphological lexicon as “dāvana” (‘gift’) and is found in a dictionary. A sample of full output data that is returned by the transliteration and lemmatization service is given in Figure 4.

```
<translit input="dahroanàm">
  <group opt_rules="no" dict="yes"/>
  <group opt_rules="yes" dict="yes">
    <variant wordform="dāvanām">
      <lemma form="dāvana">
        <dict id="MEV"/>
        <dict id="SV"/>
      </lemma>
    </variant>
    <variant wordform="dāvānām">
      <lemma form="dāvāna">
        <dict id="MEV"/>
      </lemma>
    </variant>
  </group>
  <group opt_rules="no" dict="no">
    <variant wordform="dāroanām"/>
  </group>
  <group opt_rules="yes" dict="no">
    <variant wordform="dāroānām"/>
  </group>
</translit>
```

Figure 4: Sample output data returned by the transliteration and lemmatization service.

Usually each of these groups contain more than one variant, thus it would be convenient to sort them in a more relevant order, e.g. by exploiting word form frequency information from a text corpus. For instance, “dāvāna” (in Figure 4) is a specific orthographic form of “dāvana”; it is not used in modern Latvian and is rarely used even in historical texts. A reasonable solution (at the front-end) could be that variants that are found in an on-line dictionary (e.g. the one tagged by ‘SV’ in Figure 4) are given before other variants (in the same credibility group), as they can be further passed to the dictionary service to get an explanation (see Section 4). Also note that the current trade-off is that lemmas that are not found in any dictionary are not included in the final output to avoid overloading end-users with too many irrelevant options.

3.4 Alternative sets of transliteration rules

Linguists distinguish several general groups in which Latvian historical texts can be arranged according to the orthography used.

In the current architecture, the transliteration service receives a single word form per request along with two metadata parameters: publication year and typeface (Fraktur or Latin).

Taking into account the general groups and the provided metadata, for each case there is a specific, handcrafted set of transliteration and OCR correction rules. The metadata is used for automatic selection of a rule set. However, we cannot guarantee that the selection is the most appropriate, if all the parameters overlap between two groups (due to the fact that several historical orthography variants were used in parallel for some time, and changes were rather gradual). There is also an issue caused by the uniform OCR configuration that has been used for all texts despite the orthographic and typeface differences: in the result, part of the rule sets extensively deal with OCR errors lowering precision to improve recall (i.e., overgenerating transliteration variants).

There are at least two (parallel) scenarios how this issue could be addressed in future. First, a specific OCR configuration (a FineReader training file) could be adjusted for each text group, running the OCR process again and enclosing configuration IDs in the metadata. To a large extent, this could be done automatically, involving manual confirmation in the borderline cases. Second, a larger text fragment could be passed along with the target word form, so that it would be possible to detect specific orthographic features by frequency analysis of letter-level n-grams and by analyzing the spelling of common function words. This would allow choosing an optimal set of transformation rules to ensure an optimal error correction and transliteration[6]. It would also decrease the amount of nonsense transliteration variants.

3.5 Disambiguation and evaluation

The transliteration system, as described above, results in multiple options for possible modern spellings of a given word form. While this is useful (or at least acceptable) in interactive use-cases, other uses (e.g. full-text transliteration) may require automatic disambiguation, choosing the most probable variants. The probability ranking could be obtained by comparing the variants against a word frequency table obtained from a modern text corpus of a matching genre (i.e., newspapers, fiction, etc.), according to the metadata of the analyzed text. In addition, a POS tagger of modern Latvian[7] can be used to eliminate part-of-speech categories that are contextually unlikely possible.

Furthermore, the likelihood of each transformation rule can be estimated by comparing an automatic transliteration of a historical text with a manual transliteration of the same text. While it would be too labor-intensive to manually create such a parallel corpus, we plan to obtain a parallel corpus of historical texts that have been reprinted (as is) in the modern orthography, and use this data to evaluate and improve the final system.

4. Dictionary service

On a user request, an unknown word (lemmatized in the modern orthography by the transliteration service) is passed to a dictionary service that is based on a large on-line dictionary of Latvian[8]. The dictionary contains more than 185 000 entries that are compiled from the

[6] This would even allow distinguishing more specific rule sets than it is possible by relying only on the (extended) metadata.
[7] http://valoda.ailab.lv/ws/tagger/
[8] http://www.tezaurs.lv/sv/

Dictionary of the Standard Latvian Language[9] and more than 150 other sources. It covers common-sense words, foreign words, regional and dialect words, and toponyms (contemporary and historical names of regions, towns and villages in Latvia). Explanations include synonyms, collocations, phraseologies and historical senses.

```
<dict id="SV">
  <entry src="KV">
    <word>
      <lemma>dāterēt</lemma>
      <gram>apv.</gram>
    </word>
    <sense>
      <def>Ātri un neskaidri runāt.</def>
    </sense>
  </entry>
</dict>
```

Figure 5: An entry returned by the dictionary service.

A simple entry returned by the dictionary service is given in Figure 5. It gives a meaning of a rarely used historical regional word for which even Google returns no hits (as of 2012-03-03).

5. Use-cases

The initial and primary goal is to integrate these services in the interactive user interface of an on-line digital library of historical periodicals[10], allowing users to get hints on what a selected utterance of a (historical) word means.

Another major goal is to facilitate extraction and cataloguing of named entities in historical corpora. For this purpose, the transliteration engine will be integrated in a named entity recognition system that is currently being developed. It will be used while indexing person names and other named entities mentioned in texts by mapping these names to their modern spelling. This will allow searching for proper names regardless of how they might be spelled in the historical documents.

6. Conclusion

We have designed and implemented a set of services (currently, a demo version) that facilitates the accessibility of historical Latvian texts to contemporary readers. These services will be used to improve the accessibility of historical documents in the digital archives of the National Library of Latvia — a sizeable corpus containing approx. 4 million pages[11].

The evaluation of transliteration accuracy for various text groups is pending.

We have observed that the quite erroneous OCR output of Latvian Fraktur script can be tackled by the same technical means as orthography changes. Although they seem different problems, the correlation between font-face changes and orthography developments, as well as the possibility to match the transformation results against a large lexicon allows tackling both problems simultaneously.

It has to be noted that the system is designed to be generic and extensible for other transliteration needs by specifying appropriate sets of lexical transformation rules. While currently it is aimed to be used for analysis of historical texts, future work could address the transliteration of modern texts in cases where different spelling is systematically used. For instance, in the case of user generated web content where various transliteration approaches for non-ASCII characters have often been used in Latvian due to the technical incompatibilities and inconvenience of various systems.

References

Ozols, A. (1965). *Veclatviešu rakstu valoda*. Riga: Liesma

Paegle, Dz. (2001). Latviešu valodas mācībgrāmatu paaudzes. Otrā paaudze 1907–1922. In *Teorija un prakse*. Riga: Zvaigzne ABC, pp. 39–47.

Paegle, Dz. (2008). Pareizrakstības jautājumu kārtošana Latvijas brīvvalsts pirmajos gados (1918–1922). In *Baltu filoloģija XVII, Acta Universitatis Latviensis*, pp. 89–102

Paikens, P. (2007). Lexicon-Based Morphological Analysis of Latvian Language. In *Proceedings of the 3rd Baltic Conference on Human Language Technologies (Baltic HLT 2007)*, Kaunas, pp. 235–240

Tanner, S., Muñoz, T., Ros, P.H. (2009). Measuring Mass Text Digitization Quality and Usefulness: Lessons Learned from Assessing the OCR Accuracy of the British Library's 19th Century Online Newspaper Archive. *D-Lib Magazine*, 15(7/8)

Zogla, A., Skilters, J. (2010). Digitalization of Historical Texts at the National Library of Latvia. In I. Skadiņa, A. Vasiļjevs (Eds.), *Human Language Technologies — The Baltic Perspective (Baltic HLT 2010)*, Frontiers in Artificial Intelligence and Applications, Vol. 219, IOS Press, pp. 177–184

[9] Latviešu literārās valodas vārdnīca. Vol. 1–8. Riga: Zinātne, 1972–1996 (>64 000 entries).

[10] http://www.periodika.lv/

[11] It is expected that a running demo of these reading aids will be available in May 2012.