Backtracking algorithm for lexicon generation*

Constantin Ciubotaru

Abstract

This paper is dedicated to generating process of the Romanian Cyrillic lexicon used between 1967 and 1989. The rules for transliteration of words from the modern Romanian lexicon to their equivalents written in Cyrillic were established and argued. A backtracking algorithm has been developed and implemented that generates the Cyrillic lexicon using the transliteration rules. This algorithm actually is a tool to facilitate the work of the expert. The work of the expert is reduced to checking the transliterated variants and changing the transliteration rules.

Keywords: lexicon, transliteration, backtracking algorithm, decyrillization, morpho-syntactic descriptions (MSD).

1 Introduction

The problem of digitizing and preserving the historical-linguistic heritage is a priority domain of the digital agenda for Europe. The digitization process requires solving a series of problems related to the recognition, editing, translation, and interpretation of printed texts. The solving of these problems for the Romanian historical-linguistic heritage faces difficulties and specific aspects: a large number of periods in the evolution of the language, a small volume of stored resources that are also scattered, a great diversity of alphabets.

The presence of a digitized Romanian Cyrillic lexicon will contribute to the regeneration, revitalization and preservation of the heritage related to this period. Various aspects of the problem have been exposed in [1]–[3].

*This work was supported by the project Nr. 20.80009.5007.22

The paper addresses the issues related to the digitization and transliteration of the historical-linguistic heritage printed in Cyrillic script during 1967–1989 on the territory of the Moldovan Soviet Socialist Republic (MSSR), in accordance with the linguistic norms of the modern Romanian language.

During that period the Moldovan Cyrillic alphabet (AlphaCYR) was used which actually represents the Russian alphabet without the letters "ё", "щ" and "ъ" and extended by adding the letter "ӂ" in 1967. Complete lack of resources in electronic format and presence of fragmentary grammatical descriptions that admit ambiguous interpretations represent the main difficulties specific to the period.

According to the dexonline definition, *transliteration* is the "transcription of a text from one alphabet to another, rendering the letters by their equivalents, regardless of the phonetic value of the signs" [4].

The process of transliterating Romanian words into their written equivalents with the characters of the AlphaCYR alphabet is called *cyrillization*. For instance, *"puiului"*⇒"пуюлуй", *"fiului"*⇒"фиулуй", *"cenușiu"*⇒"ченушиу", *"viermi"*⇒"вермь", *"vierii"*⇒"виерий".

The inverse procedure for cyrillization is called decyrillization, e.g. "пуюлуй"⇒ *"puiului"*, "бьет"⇒ *"biet"*, "боер"⇒ *"boier"*, "пепт"⇒ *"piept"*.

If the digitization of the text is relatively simple, the problem of recognizing the digitized text is quite complicated, especially considering the total lack of Romanian Cyrillic resources for that period. This paper extends the results presented in [5] and aims to develop a tool for generating the lexicon corresponding to that period (noted by LexCYR), starting from the lexicon of the modern Romanian language (noted by LexROM).

The general scheme of the Romanian Cyrillic lexicon generator is presented in Figure 1.

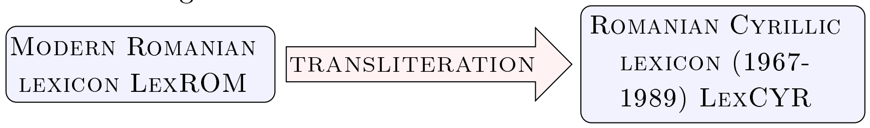
Modern Romanian lexicon LexROM
transliteration
Romanian Cyrillic lexicon (1967-1989) LexCYR

Figure 1. The general scheme of the Cyrillic lexicon generator

2 Selection of the modern Romanian lexicon

For the choice of the modern Romanian lexicon, the following three resources were examined:

1. ***Dexonline*** [4]. It contains over 900000 entries, with a convenient interface for online use. The dictionary structure is less adaptable for processing because it does not contain explicitly the inflected forms, does not contain morpho-syntactic descriptions (MSD), and includes both forms of spelling "î" from "i" and "â" from "a" ("fân"-"fîn", "pârî"-"pîrî").

2. ***The lexicon developed at the "Al.I.Cuza" University, Iași*** [6] with over 1000000 entries. The lexicon is well structured, contains MSD labels in accordance with the tagset proposed in the project MULTEXT-East [7]. But, as in dexonline, we find both spellings "î" / "â", also many proper names and words of foreign origin, to which the rules of transliteration cannot be applied.

3. ***Reusable linguistic resources*** developed at the Institute of Mathematics and Computer Science “Vladimir Andrunachievici” [8] with over 677000 entries, including inflected forms. The formalization (packaging) of resources is quite complicated, the morpho-syntactic descriptions are incomplete.

Finally, the lexicon developed at the "Al.I.Cuza" University (LexROM) was selected with minor modifications, as follows:

1. Proper nouns and words of foreign origin were removed;

2. All words were transliterated using the spelling "â" from "a" according to the provisions of the Romanian Academy. Duplications of spellings "î" / "â" were avoided by applying an algorithm specially developed for this purpose.

The problem of spelling "î"/"â" does not affect the cyrillization process, because in both cases there is the same result at transliteration: *"â"*⇒"ы", *"î"*⇒"ы". Difficulties arise in the decyrillization process: should we apply the rule "ы"⇒ *"â"* or rule "ы" ⇒ *"î"*?

We denote by AlphaROM the Romanian language alphabet, and by LexROM(α) – all the words from LexROM that start with the letter α, $\alpha \in$ AlphaROM.

3 Used tools

To formalize the transliteration rules and program the lexicon processing algorithms there was selected the Common LISP functional programming language [9], [10].

The Notepad++ editor was used for word processing [11], which offers advanced editing capabilities, such as:

- select text both horizontally and vertically,
- store search results in separate files,
- mark lines and operations with these lines,
- allow the use of regular expressions,
- support UTF-8 encoding for Romanian letters with diacritics and Russian, for example: Ă, ț, Â, â, Ș, Э, ц, Ы, Ш, ӂ,
- rich set of plugins: exporting files in various formats (RTF, HTML), the ability to launch applications (files with the extension .exe), sorting and comparing files, etc.

4 Backtracking method

The backtracking method proposes to build the solution(s) of a problem incrementally by applying iterative and/or recursive algorithms. It is assumed that there is a finite set of candidates for solutions and some internal criteria for verifying candidates. The method can be applied to generate the lexicon, as all the necessary conditions are met:

- the modern LexROM lexicon is given,
- sets of rules for transliteration are defined,
- there is a finite set of intermediate transliterated words that represent candidates for solutions,
- there are internal criteria for verifying the variants: the order of application of the rules, context-sensitive dependencies, prefixing and suffixing, the involvement of the expert,
- the set of all solutions meets the LexCYR lexicon,
- iterative and recursive algorithms are applied.

5 Algorithm of switching to the spelling "â" from "a"

The transition to the spelling "â" from "a" also will be done by transliteration. According to the provisions of the Romanian Academy, the letter "î" will always be written at the beginning and end of the word ("început", "înger", "în", "întoarce", "a coborî", "a urî"). Inside the word, it is usually written "â" ("cuvânt", "a mârâi", "român", "fân"). There are, however, a few exceptions to this rule. Words formed by prefixing words that begin with the letter "î" will keep this "î" inside. For example, "neîmpăcat", "neîngrijit", "preîntâmpinat", "dezîntors", "reînarma". The same rule will be applied to compound words: "bineînțeles", "semiînchis", "altîncotro". There are also a few exceptions, for example, the word "altînghie" will be transliterated as "altânghie", because it is not a compound word, this is the name of a flower, also called "lady's slipper". On the other hand, the word "capîntortură" (the name of a bird) will be transliterated, together with its derivatives, as "capîntortură". It is taken into account that the word comes from "cap întors" ("turned head"). The specificity of the LexROM lexicon will also be taken into account, that includes, along with the lemma words and inflected forms, phrases and word combinations, which can be spelled with "î" from "i". These words inside the construction are separated by "˜". For example, "pe˜înserate", "de˜jur˜împrejur" etc. All words w that contain at least one letter "î" can be represented as $w = w_0 \bullet \text{"î"} \bullet w_1 \bullet \text{"î"} \bullet \ldots \bullet w_{n-1} \bullet \text{"î"} \bullet w_n$. If the word starts with "î", then $w_0 = \text{""}$. We will mark by "" the empty string. For words ending with "î" we will have $w_n = \text{""}$. Thus, for the letter "î" we get "î"$= w_0 \bullet \text{"î"} \bullet w_1$, $w_0 = w_1 = \text{""}$. For the word "coborî" we obtain: "coborî"$= w_0 \bullet \text{"î"} \bullet w_1$, with $w_0 =$"cobor", $w_1 = \text{""}$. For the combination of words $w =$"din˜cînd˜în˜cînd" we have: $w = w_0 \bullet \text{"î"} \bullet w_1 \bullet \text{"î"} \bullet w_2 \bullet \text{"î"} \bullet w_3 = \text{"din˜c"} \bullet \text{"î"} \bullet \text{"nd˜"} \bullet \text{"î"} \bullet \text{"n˜c"} \bullet \text{"î"} \bullet \text{"nd"}$. Note that $w_1 =$"nd˜" ends with "˜", which means that the next word will start with "î", analogous to the prefix situation. As a result of the conversion we get "din˜când˜în˜când".

Performing a statistical analysis of the LexROM lexicon leads to selection of the set of all prefixes that can be inserted in front of words that start with the letter "î". This set is denoted by PREFIXES.

Algorithm of switching to the spelling "â" from "a"

```
0. Start
1. The lexicon of the modern Romanian language LexROM is given.
   \* We will modify this lexicon by substituting all words with their written equivalents with "â" from "a" applying the transliteration method *\
2. We modify the LexROM by applying transliteration rules for exceptional situations. For example, "altîngie" ⇒ "altângie" (in other cases "alt" will be a prefix).
3. We build the set of prefixes that can be placed in front of words which start with the letter "î". PREFIXES={"alt" "arhi" "auto" "bine" "bio" "de" "dez" "din" "ex" "ne" "nemai" "ori" "piți" "pre" "prea" "pro" "re" "semi" "sub" "subt" "super" "supra" "tele"}.
4. loop for all $w \in$ LexROM do
   4.1. if $w$ does not contain "î" then return($w$).
   4.2. We represent $w = w_0 \cdot$ "î" $\cdot w_1 \cdot$ "î" $\cdot \ldots \cdot w_{n-1} \cdot$ "î" $\cdot w_n$, where $w_0, w_1, \ldots, w_n$ are words which do not contain "î", $n \geq 1$.
   4.3. if ($w_0 =$ "") or ($w_0 \in$PREFIXES) or ($w_0 = w'_0 \cdot$"~") or ($w_1 =$ "")
        then $w_r := w_0 \cdot$"î" else $w_r := w_0 \cdot$"â".
   4.4. loop for $i$ from 1 to $(n-1)$ do
        4.4.1. $w_r := w_r \cdot w_i$
        4.4.2. if ($w_{i+1} =$ "") or ($w_i = w'_i \cdot$"~") then $w_r := w_r \cdot$"î" else
               $w_r := w_r \cdot$"â".
   4.5. end loop
   4.6. return($w_r \cdot w_n$)
5. end loop
6. Stop
```

6 The structure of the lexicons

The LexROM lexicon is represented as a list in Common LISP, each element of the list being composed of three components: (word, MSD-label, word-lemma). For each element of the LexCYR lexicon, the fourth component – the cyrillized word (Figura 2) – is included.

(ghiocei "Ncmprn" "ghiocel")	(гиочей "ghiocei" "Ncmprn" "ghiocel")
(ridic "Vmsp1s" "ridica")	(ридик "ridic" "Vmsp1s" "ridica")
(ridica "Vmn" "ridica")	(ридика "ridica" "Vmn" "ridica")
(ridicam "Vmii1p" "ridica")	(ридикам "ridicam" "Vmii1p" "ridica")
(ridicăm "Vmsp1p" "ridica")	(ридикэм "ridicăm" "Vmsp1p" "ridica")
(ridicare "Ncfsrn" "ridicare")	(ридикаре "ridicare" "Ncfsrn" "ridicare")
(ridicat "Ncmson" "ridicat")	(ридикат "ridicat" "Ncmson" "ridicat")
(ridicat "Afpmson" "ridicat")	(ридикат "ridicat" "Afpmson" "ridicat")
(ridicat "Vmp" "ridica")	(ридикат "ridicat" "Vmp" "ridica")
(ridicat "Rg" "ridicat")	(ридикат "ridicat" "Rg" "ridicat")
(ridicatele "Ncfpry" "ridicat")	(ридикателе "ridicatele" "Ncfpry" "ridicat")
(ridicatule "Ncmsvy" "ridicat")	(ридикатуле "ridicatule" "Ncmsvy" "ridicat")
(a) LexROM structure	(b) LexCYR structure

Figure 2. The lexicons structure

The MSD label is a set of characteristics of the word viewed as part of speech. The label represents a sequence of symbols, the first symbol specifying the part of speech (for example, N - noun, V - verb, A - adjective, Rg - adverb, etc). The rest of the symbols will specify the morphological characteristics of the word, such as number, gender, person, time, case, mode, etc. The scheme of the MSD label for the noun is shown in Figure 3.

$$N\begin{bmatrix}\text{c-common}\\ \text{p-proper}\end{bmatrix}\begin{bmatrix}\text{m-masculine}\\ \text{f-feminine}\end{bmatrix}\begin{bmatrix}\text{s-singular}\\ \text{p-plural}\end{bmatrix}\begin{bmatrix}\text{r}\\ \text{o}\\ \text{v}\end{bmatrix}\begin{bmatrix}\text{y-definiteness}\\ \text{n-indefiniteness}\end{bmatrix}$$

r-direct case (nominative-accusative),
o-oblique case (genitive-dative),
v-vocative

Figure 3. The MSD label structure for noun

The following algorithm based on the backtracking strategy is proposed for generating the Cyrillic lexicon LexCYR. Sets of transliteration rules are defined, cyrillization and decyrillization algorithms are constructed. The cyrillization algorithm is applied on the Romanian lexicon LexROM. A variant of the LexCYR lexicon will be obtained, which can be subjected to decyrillization, thus a new variant for the Romanian lexicon is obtained. The ideal situation would be to match these two lexicons. If inconsistencies occur, the expert intervenes, who can change the rules of cyrillization\decyrillization, can repeat the whole process or can intervene with corrections on the constructed Cyrillic lexicon.

7 Cyrillization

Unlike the problem of digitizing and recognizing printed text, which is solved relatively simply, the problem of cyrillization is more difficult. To solve this problem we will apply the transliteration method. By definition, the transliteration process consists in the consecutive application of a set of substitutions (rewriting rules). For example, *brad* ⇒ б*rad* ⇒ бр*ad* ⇒ бр*a*д ⇒ брад. Here the following rules have been applied consecutively "*b*" ⇒ "б", "*r*" ⇒ "р", "*d*" ⇒ "д", "*a*" ⇒ "а". We will call these rules general rules. For them the order of application is irrelevant.

For the letter "*i*" we have the general transliteration rule "*i*" ⇒ "и", but the following rules are also possible: "*i*" ⇒ "й" and "*i*" ⇒ "ь". Examples: *fuior* ⇒ фуйор, *fior* ⇒ фиор, *miere* ⇒ мьере.

In other cases the rules may be more complicated. For example, two rules can be applied to the letter "*g*": "*g*" ⇒ "г", "*g*" ⇒ "ж" – *gigant* ⇒ жигант. Here comes the context-sensitive rule that requires substitutions: "*gi*" ⇒ "жи", "*ge*" ⇒ "же", "*ghi*" ⇒ "ги", "*ghe*" ⇒ "ге".

Thus, it becomes obvious that these rules must be applied before applying the general rule "*g*" ⇒ "г". Moreover, substitutions "*giu*" ⇒ "жиу", "*giu*" ⇒ "жю" are also possible. For example, *giulgiu* ⇒ жюлжиу, *giugiuli* ⇒ жюжюли.

Randomly applying the transliteration rules for the word "ghiocei", the following variants are obtained: {"гхиокеи", "гхиокеь", "гхиокей", "гхиочеи", "гхиочеь", "гхиочей", "гиокеи", "гиокеь", "гиокей", "гиочеи", "гиочеь", "гиочей", "жхиокеи", "жхиокеь", "жхиокей", "ӂхиочеи", "ӂхиочеь", "ӂхиочей"}.

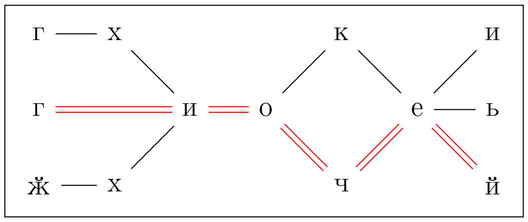

Figure 4. The transliteration scheme for "ghiocei"

In Figure 4 we show the transliteration scheme of the word "ghiocei". The scheme highlights the correct variant – "гиочей". The backtracking method allows eliminating wrong options step by step and selecting the correct one. This is done by changing the transliteration rule set, establishing some contextual dependencies, changing the order of the rules application and examining the MSD labels. In some situations the correct option can only be selected by the expert.

To fix the situations with multiple variants, we will use a list of options denoted by $[w_1][w_2]\ldots[w_n]$, finally being selected only one. For example, for the words "ghiocei" and "preaiubiți" we get:

[гх][г][ӂх] • ио • [к][ч] • е • [и][ь][й] $\Longrightarrow$"гиочей",

пр • [еа][я]• [иу][ю] • биц • [и][ь][й] $\Longrightarrow$"прябюбиць".

We denoted by "•" the concatenation operation.

8 Classification of transliteration rules

8.1 General rules

The general transliteration rules are presented in Table 1.

Table 1. General rules of transliteration

latin	a	ă	â	b	c	d	e	f	g	h	i	î	j	k
cyrillic	а	э	ы	б	к	д	е	ф	г	х	и	ы	ж	к
latin	l	m	n	o	p	r	s	ș	t	ț	u	v	x	z
cyrillic	л	м	н	о	п	р	с	ш	т	ц	у	в	кс	з

To formalize (program) these substitutions, we will introduce the function *replace-all (w lat cyr)*, which will modify the word *w* substituting all occurrences *lat* with *cyr*. This is possible because the order of application of these substitutions is not relevant. E.g, *replace-all* ("dividend" "d" "д") = "дiviденд".

Depending on the filtering stage, it is possible to enter some new general transliteration rules, for example, *replace-all* (w "gh" "г"), *replace-all* (w "ch" "к").

Usually, these substitutions are the last filter in the process of cyrillization.

8.2 Rules for prefixes

Because the words are interpreted as strings, we have to use the notions of prefix and suffix defined to process strings, as opposed to the grammatical notions of suffix and prefix. Thus, by prefix (suffix) of the string w we will define any substring w_1 (w_2) for which $w = w_1 \bullet w_2$. Substrings w_1 and w_2 can also be empty, i.e. "". Often the transliteration rules for prefixes differ from general rules. Thus, the prefixes "ia" and "iu", with small exceptions, will be transliterated as "я" and "ю", as opposed to their appearance inside the word when in most cases they will be transliterated as options "[я][иа]" and "[ю][иу]". Another example: in the LexROM there are about 650 words that start with the prefix "crea". Only for 6 situations it will be transliterated by "кря". All other occurrences of the prefix will be transliterated by креа". These 6 situations can be easily highlighted and formalized. This observation suggests the need to introduce a special set of transliteration rules for prefixes.

We note these rules by *replace-prefix (w prefixlat prefixcyr)*. For example, *replace-prefix (w "creang"* "крянг"), *replace-prefix (w "crea"* "креа"). The order of application is important for this type of substitution, which is simple: prefixes that are prefixes of other prefixes will be transliterated last. Thus, we first will try the transliteration "creang" ⇒ "крянг", then the transliteration "crea" ⇒ креа". Prefix rules are defined separately for all words in the LexROM that begin with the same letter. Thus, for all letters there will be defined sets of rules for prefixes that will be applied first in the transliteration process.

8.3 Rules for suffixes

Analogously to the situation with the transliteration of prefixes, also there are defined transliteration rules for suffixes involving some specific conditions. For example, for the termination "ci" transliterations are possible: "ci" ⇒ "ч", "ci" ⇒ "чь", "ci" ⇒ "чи". To make the correct decision, MSD labels are checked. The rule "ci" ⇒ "ч" is applied, for example, for masculine nouns to the singular, nominative-accusative case (MSD = "Ncmsrn", "arici" ⇒ "арич", "cinci"⇒" чинч").

The "ci" ⇒ "чи" rule is applied to infinitive verbs and 3rd person verbs (MSD = "Vmis3s" and MSD = "Vmn", for example, "a munci" ⇒ "а мунчи"), and the rules "ci" ⇒ "чь", "ti" ⇒ "ть", "ţi" ⇒ "ць", "şi"⇒" шь", etc. – for nouns and adjectives in the plural dative-genitive case, and also for second-person present and past tense verbs (MSD ∈ {"Vmii2p", "Vmis2s", "Vmis2p", "Vmil2s", "Vmil2p", "Vmip2s", "Vmip2p", "Vmsp2s", "Vmsp2p", "Vmmp2p" }). Some examples are presented in Table 2.

Table 2. Transliteration of verb terminations

MSD	lat	cyr	MSD	lat	cyr
Vmii2p	citeaţi	читяць	Vmip2s	citeşti	читешть
Vmis2s	citişi	читишь	Vmip2p	citiţi	читиць
Vmis2p	citirăţi	читирэць	Vmsp2s	citeşti	читешть
Vmil2s	citiseşi	читисешь	Vmsp2p	citiţi	читиць
Vmil2p	citirăţi	читирэць	Vmmp2p	citiţi	читиць

Based on the above, unconditioned and conditioned rules are defined for the transliteration of suffixes. Respectively, the functions are defined *replace-suffix (w lat* cyr) and *replace-suffix-if (w label lat* cyr *msd)*. The "label" argument of the *replace-suffix-if* function represents the label MSD of the processed word *w*, and the argument "msd" – a set of valid

MSD labels for this rule. Unlike the rules for prefixes that are defined separately for each letter, the rules for suffixes are universal and can be applied to all words.

8.4 Context sensitive rules (for diphthongs and triphthongs)

As in the case of prefixes (suffixes), along with the usual grammatical notions diphthong/triphthong, we examine other combinations consisting of two, three or more letters. We mentioned above the behavior of the diphthongs "ia" and "iu" as prefixes, but also as occurrences within the word. Other examples are presented in Table 3.

Table 3. Transliteration of diphthongs/triphthongs

Diphthong/ triphthong	Transli- teration	Examples
"ioa"	[оа]	cioară⇒ "чоарэ"
	[ьоа]	chioară⇒ "кьоарэ"
	[иоа]	mioară⇒ "миоарэ"
"ii"	[ий]	fiicele⇒ "фийчеле"
	[ии]	viile⇒ "вииле"
"eie"	[ее]	creier⇒ "креер"
	[ей]	conveier⇒ "конвейер"

Diphthong/ triphthong	Transli- teration	Examples
"ea"	[а]	ceață⇒ "чацэ"
	[я]	rea⇒ "ря"
	[еа]	ocean⇒ "очеан"
"ch"	[к]	ochi⇒ "окь"
"gh"	[г]	ghid⇒ "гид"
"ge"	[же]	ger⇒ "жер"
"ci"	[чи]	circ⇒ "чирк"

Namely the transliteration of these constructions generates the most ambiguities. More information on this topic can be found in [12]. To make right decisions, sometimes contextual rules can be supplemented with morpho-syntactic information (MSD labels). The order of application of the rules is very important. Contextual dependencies always have priority over general rules.

9 Cyrillization algorithm

The cyrillization algorithm applies consecutively the transliteration rules, previously defined, to all the words in the LexROM. It is important to follow the order of application of the rules. Of course, optional combinations will be generated, which correspond to the ambiguities. This means that later it is necessary to modify the transliteration rules or to request the intervention of the linguistic expert.

Below we present the formalized algorithm.

CYRILLIZATION ALGORITHM

```
0. Start
1. The lexicon of the modern Romanian language LexROM and transliteration rules are given.
   \* We will build the Romanian Cyrillic lexicon LexCYR for the period 1967-1989 *\
2. Initial LexCYR = $\varnothing$, LexROM$_1$ = LexROM.
3. loop for all letters $\alpha \in$ AlphaROM do
  3.1. loop for all words $w \in$ LexROM$_1(\alpha)$ do
    3.1.1. Transliteration rules for prefixes are applied.
    3.1.2. Transliteration rules for suffixes are applied.
    3.1.3. Context-sensitive rules for transliteration are applied.
    3.1.4. General rules for transliteration are applied. The obtained result is denoted by $wcyr$.
    3.1.5. $wcyr$ is included in LexCYR.
  3.2. end loop
4. end loop
5. Stop
```

10 Decyrillization

Decyirillization faces the same problems as cyrillization. General and contextual rules are also defined. The general rules are relatively simple, for example, а $\Rightarrow$ *a*, р $\Rightarrow$ *r*, ю $\Rightarrow$ *iu*, ь $\Rightarrow$ *i*. If only the general rules are applied to transliteration, we obtain, for example, пуюлуй $\Rightarrow$ *puiului*, бьет ⇒ *biet*, боер ⇒ *boer*, пепт ⇒ *pept*. The last two transliterations are incorrect. Correct would be боер ⇒ *boier*, пепт ⇒ *piept*. In this case, as for cyrillization, contextual rules are required (for prefixes/suffixes, diphthongs/triphthongs). E.g, г•β ⇒ *gh*•β, if β∈{е,и,я,ю,ь} and г•β ⇒ *g*•β, if β∉{е,и,я,ю,ь} (георгинэ ⇒ *gheorghină*, гогоашэ ⇒ *gogoașă*).

Rules for the letter я: я ⇒ *ia* (usually at the beginning of the word), ия ⇒ *ia* (usually at the end of the word). If it is difficult to make the right decision to transliterate the letter я inside the word, then the algorithm will use the rule я ⇒ [*ia*][*ea*], leaving the right decision to the expert. More information on this topic can be found in [13].

Another difficult problem is the transliteration of the letter ы, which can be replaced by either *î* or *â*. The algorithm follows exactly the recommendations of the Romanian Academy regarding this spelling.

DECYRILLIZATION ALGORITHM

```
0. Start
1. The Romanian Cyrillic lexicon for the period 1967–1989 LexCYR and
   the decyrillization rules are given.
   \* We will build the lexicon of the modern Romanian language (noted by
   LexROM$_2$) applying the transliteration method *\
2. Initial LexROM$_2$ = $\varnothing$
3. loop for all letters $\beta \in$ AlphaCYR do
  3.1. loop for all words $w \in$ LexCYR($\beta$) do
    3.1.1. Transliteration rules for prefixes are applied.
    3.1.2. Transliteration rules for suffixes are applied.
    3.1.3. Context-sensitive rules for transliteration are applied.
    3.1.4. Transliteration rules for the letter  kyr ы are applied.
    3.1.5. General rules for transliteration are applied. The obtained result
           is denoted by $wrom$.
    3.1.6. $wrom$ is included in LexROM$_2$.
  3.2. end loop
4. end loop
5. Stop
```

11 Lexicon generation technology

As it was mentioned above, there is a total lack of electronic resources for the period 1967-1989, a complete exposition of the grammar used is missing, and many of interpretations of the transliterated words are ambiguous. Therefore, a major role in the process of generating the lexicon belongs to expert. The proposed technology aims to automate this process. Having the cyrillization and decyrillization algorithms and the formalized sets of transliteration rules, the lexicon generation process can be realized as an backtracking algorithm. The process runs in several iterations, at each iteration the expert intervenes to modify the set of rules and, possibly, directly the built Cyrillic lexicon. This scheme is described in detail in Figure 5.

12 Conclusion

The paper proposes a backtracking technology for the generation of the Romanian Cyrillic lexicon for the period 1967–1989 applying the transliteration method. Starting from the lexicon of the modern Romanian language [6] the cyrillization and decyrillization algorithms are applied consecutively.

The intermediate results are made available to the experts, who can modify\extend the set of rules applied to transliteration, and to directly correct the built Cyrillic lexicon. The final lexicon is obtained as a result of performing several such iterations. The main problems to be solved by the experts are the ambiguities that appear as a result of cyrillization\decyrillization.

For all words in the LexROM(c) (171846 words), 6381 ambiguities were detected at the first iteration, which represents 3.7%. To overcome these ambiguities there were required two iterations. Of course, the degree of accuracy depends considerably on the qualification of the expert. The proposed technology allows the return to the previous intermediate variants, thus revising the lexicon.

In order to become aware of the role of the expert and that of con-

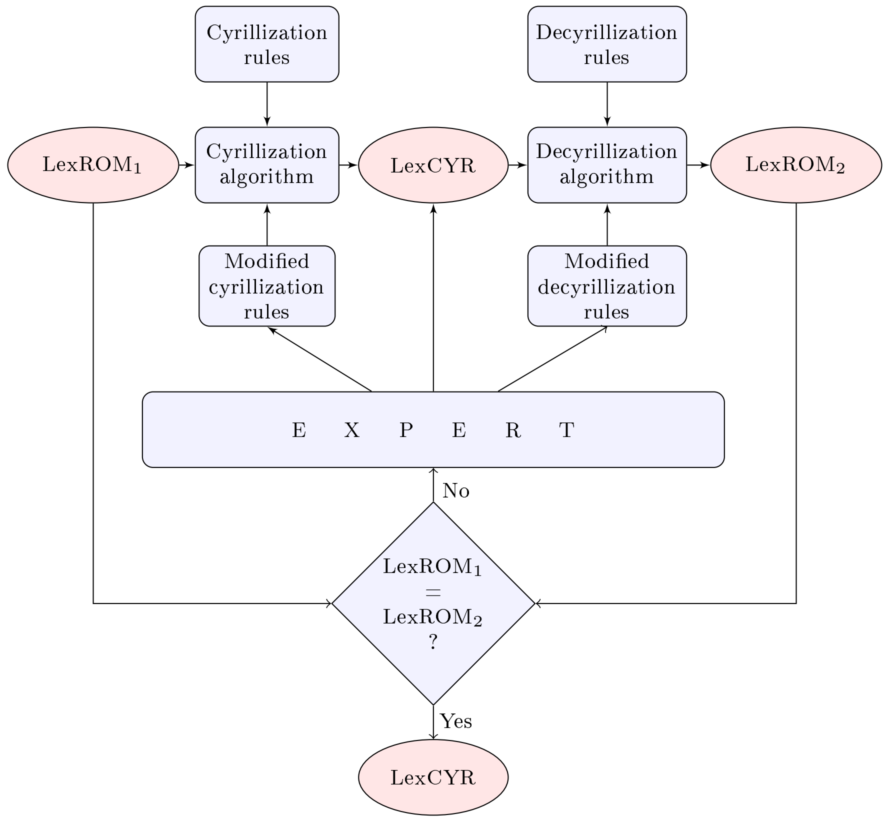

Figure 5. The scheme for generating the Romanian Cyrillic lexicon

textual dependencies, a test was performed applying to the LexROM(c) only the general rules of transliteration (paragraph 8.1). As a result, 42.2% of the correct words are obtained.

References

[1] S. Cojocaru, E. Boian, C. Ciubotaru, A. Colesnicov, V. Demidova, and L. Malahov, "Regeneration of printed cultural heritage : challenges ang technolologies," in *The Third Conference of Matematical*

Society of the republic of Moldova, (19-23 August, Chişinău), 2014, pp. 481–489.

[2] C. Ciubotaru, A. Colesnicov, and L. Malahov, "Vitalization of Moldavian Printings (1967–1989)", in *Proceedings of the 4th Conference of Mathematical Society of Moldova, CMSM4'2017"*, (June 28-July 2, 2017, Chisinau, Rep. of Moldova), pp. 491–494, ISBN 978-9975-71-915-5, Available: `http://cmsm4.math.md/Proceedings_CMSM4.pdf`.

[3] C. Ciubotaru, S. Cojocaru, A. Colesnicov, V. Demidova, and L. Malahov, "Regeneration of cultural heritage: problems related to moldavian cyrillic alphabet", in *Proceedings of the 11th International Conference "Linguistics Resources and Tools for Processing the Romanian Language (ConsILR-2015)"*, (Alexandru Ioan Cuza University, Iași, Romania, 26-27 November 2015), pp.177–184, ISSN: 1843-911X, Available: `http://consilr.info.uaic.ro/2015/Consilr_2015.pdf`.

[4] *Dexonline. Online Dex. Romanian language dictionaries.* [Online]. Available: `https://dexonline.ro/definitie/translitera\%C8\%9Bie`.

[5] C. Ciubotaru, V. Demidova, and T. Bumbu, "Generation of the Romanian Cyrillic lexicon for the period 1967 – 1989," in *Proceedings of the Fifth Conference of Mathematical Society of Moldova IMCS-55*, (September 28 - October 1, Chisinau, Republic of Moldova), 2019, pp. 309–316.

[6] The UAIC Natural Language Processing Group, *WebPosRo/resources/*, Alexandru Ioan Cuza University, Faculty of Computer Science. [Online]. Available: `http://nlptools.info.uaic.ro/WebPosRo/resources/posDictRoDiacr.txt`.

[7] Tomaz̧ Erjavec, ed., *MULTEXT-East Morphosyntactic Specifications, Version 3.0*, May 10th, 2004. [Online]. Available: `http://nl.ijs.si/ME/Vault/V3/msd/html`.

[8] *Reusable Resources for Romanian Language Technology*, Institute of Mathematics and Computer Sciences, Moldova. [Online]. Available: `http://www.math.md/elrr/res_main.php`.

[9] Guy L. Steele, *Common Lisp the Language*, 2nd ed., USA: Thinking Machines, Inc. Digital Press, 1990, 1029 p. ISBN:1-55558-041-6.

[10] *CLISP – an ANSI Common Lisp*, Slashdot Media. [Online]. Available: `http://sourceforge.net/projects/clisp/files/clisp/2.49/`.

[11] *Notepad++. Downloads.* [Online]. Available: `https://notepad-plus-plus.org/download/v7.7.1.html`.

[12] V. Demidova, "Particular Aspects of the Cyrillization Problem," in *The Third Conference of Matematical Society of the Republic of Moldova*, (Chişinău, 19-23 August), 2014, pp. 493–498.

[13] V. Demidova, "Peculiarities of decyrillization of the Romanian language," *Studia universitatis Moldaviae. Seria "Științe exacte și economice"*, no. 2(82), pp. 16–20, 2015. (in Romanian).

Constantin Ciubotaru

Received March 2, 2021
Revised March 20, 2021

Vladimir Andrunachievici Institute of
Mathematics and Computer Science
Republic of Moldova
E–mail: `chebotar@gmail.com`