Old or new, we repair, adjust and alter (texts)

UDC 811.163.41'322.2: 004.9
DOI 10.18485/infotheca.2019.19.2.3

ABSTRACT: In this paper we present how e-dictionaries and cascades of finite-state transducers, as implemented in Unitex, can be used to solve three text transformation problems: correction of texts after OCR, restoration of diacritics and switching between different language variants.

KEYWORDS: text correction, OCR errors, diacritic restoration, language variants, electronic dictionary, finite-state transducers.

PAPER SUBMITTED: 13 October 2019
PAPER ACCEPTED: 08 December 2019

Cvetana Krstev
University of Belgrade, Faculty of Mathematics
cvetana@matf.bg.ac.rs

Ranka Stanković
University of Belgrade, Faculty of Mining and Geology
ranka.stankovic@rgf.bg.ac.rs

Belgrade, Serbia

1 Text mending – introduction to problems

Text mending is one of the simplest text transformation problems, when compared to speech recognition and generation, text summarization and machine translation. It is also one of the first problems posed to computers that did not involve calculation. Miller and Friedman (1957) wrote about the reconstruction of mutilated texts from the point of view of the information theory in order to calculate the redundancy in English texts. The problem discussed at that time was how many characters can be omitted while still allowing text reconstruction by humans.

Following this line of research the first practical solutions to correcting spelling errors emerged. The idea of a program that corrects spelling errors is reported by Blaire (1960). It consists of weighting the letters to create a four- or five-letter abbreviation for each word – if abbreviations match, the words are assumed to be the same. Blaire claims that "given only a vocabulary of a properly spelled words, the computer can correct most (including unanticipated) misspellings without human assistance." He also claims that programming common orthographic rules for English can not be successful.

Soon, many new approaches were tried in spelling checking programs and most of them included a correction module. Most of them were produced for English (one of the reasons was its notorious "illogical" spelling), and dealt with errors produced by humans (typographic errors, author's ignorance) or machines during transmission and storage (Peterson, 1980).

Peterson, as many other authors, classifies typographic errors into four groups: insertions, deletions, replacements and reversals. However, texts produced by humans can also be mutilated in other ways: for many reasons, when typing, humans are sometimes compelled to use a degraded instead of a standard alphabet. This results in different types of errors. As usually diacritics are omitted, the process of transforming a text in a standard alphabet is usually called "diacritic restoration", and procedures that perform this task using various approaches were developed for many languages. (Krstev et al., 2018).

Errors produced during machine text input, for instance by Optical Character Recognition (OCR), are of a different type and different solutions were developed for detecting and correcting such errors. As early as in the late 1950s, Bledsoe and Browning (1959) wrote about solving the OCR problem by using a small dictionary with a probability assigned to each word. Despite the considerable improvement of OCR software in subsequent years the problem of OCR error detection and correction is still not considered solved (see, for example, (Kolak and Resnik, 2002)), especially for more "demanding" scripts and languages (Cyrillic, Arabic, etc.).

Transformation from one language variant to another is usually not perceived as an error/correction problem. In his US Patent (2004) Henton describes a voice system that transforms American English utterances for British listeners. The system includes spelling and lexicon normalization; the first is being solved with a set of rules, the second with a list of equivalences. Similar problems are sometimes tackled as translation problems, as for instance in the case of Arabic dialects (Salloum and Habash, 2012) – authors present a rule-based machine translation system that transforms dialectal Arabic to Modern Standard Arabic.

For Serbian, text mending problems were not often reported through scientific channels. Solutions were developed for spelling correction for various platforms (e.g. as an extension for LibreOffice) but very few scientific papers were published with the underlining procedure explained (an exception is (Ostrogonac et al., 2015)). The similar is true for problems of diacritic restoration, OCR errors correction and language variants transformation.

In this paper we present an approach to solving three text mending problems for Serbian: OCR errors, diacritics omission and language variant switching. The common characteristic of these problems is that they occur in a text systematically rather than occasionally. The approach works at the word level – “incorrect words” or “suspicious words” are recognized by dictionaries (as not belonging to them), a problem specific solution is applied to transform them, and they are corrected if the offered solution is in a dictionary. The problems are usually solved locally, which means that only “incorrect words” and “suspicious words” are considered and rarely their context or more complex structures.

2 Correction of OCR errors

In the process of digitization printed books are scanned and then optical character recognition (OCR) is applied. A text that fully corresponds to the original is rarely obtained since OCR is prone to errors. The quality of the resulting text depends on various factors: the software used, quality of the paper and print of the original text, and its language and alphabet. OCR software today is of good quality compared to its first versions, even when produced for personal rather than professional use,[1] and it is applicable to a large number of languages and scripts, including Serbian Cyrillic. However, OCR of old printed books can still be a challenge due to various reasons: the use of old and non-standard fonts, the deterioration of paper, and if the book that is digitized comes from a library, which is often the case, handwritten additions from numerous users (underlying, redactor’s marks, comments, etc.).

OCR errors that occur in scanned texts differ from typing errors, which can be divided into two groups, typographic errors that are the result of mistypes and cognitive errors caused by a misunderstanding of the correct spelling of a word (Kukich, 1992). Errors of the first type can be tackled in terms of keyboard key proximity, while errors of the second type are language dependent and are the result of user’s (mis)understanding of relevant orthographic rules. Both types of errors can result in production of either valid (but not intended) or invalid words.

Akin to typing errors, OCR errors are local: one or more characters are erroneously recognized as different characters. However, contrary to typing errors, OCR errors are rarely occasional, as the same type of errors tend to be repeated in one text, while in some other text different type of errors may frequently occur. The erroneously recognized characters can be letters,

[1] For the project presented in this section we used ABBYY FineReader 12 Professional.

punctuation marks and digits. OCR errors can also produce valid words or non-valid words.

We will present the solution for detecting and correcting OCR errors developed for the compilation of the corpus of Serbian novels written and published in the period 1840–1920.[2] The novels selected for this corpus were mainly printed in Cyrillic script (only a few of them were in Latin script). When scanning, the recognition was reduced to only one script, Cyrillic or Latin, in order to avoid confusion between theese two scripts, e.g. Latin B with Cyrillic B (corresponding to Latin V). The majority of books were obtained from the University Library “Svetozar Marković”, and the rest from other libraries and private collections.

First of all, the OCR software was set to recognize in novels printed in Cyrillic only the Cyrillic script. As a consequence, occasional phrases written in Latin script (and languages other than Serbian) could not be recognized correctly and had to be retyped manually. However, in this way confusing certain Cyrillic and Latin letters with similar graphical representation was avoided, e.g. a Cyrillic ‘a’ can be confused for a Latin ‘a’ (denoting the same letter) or a Cyrillic ‘p’ can be confused for a Latin ‘p’ (denoting different letters).

The analysis of frequently occurring OCR errors in Cyrillic texts showed that the following error types predominate:

– Some individual letters are mutually confused, for instance letters ‘п’, ‘и’, ‘н’, letters ‘с’ and ‘е’, letters ‘с’ and ‘о’ (but not ‘е’ and ‘о’);
– A group of letters can be confused for one letter or another group of letters, e.g. two letters ‘га’ and one letter ‘ш’; two letters ‘шп’ and two letters ‘иш’;
– Letters can be confused for digits or punctuation or special marks. For instance, a digit ‘0’ and a letter ‘О’; a letter ‘И’ can be recognized as 11 (but not vice versa) or a letter ‘љ’ can be recognized as ‘л>’ (but not vice versa).

Our solution addresses only falsely recognized words that result in non-valid words. It follows four steps:

1. The text obtained by OCR is processed using Serbian morphological electronic dictionaries (SMD) (Krstev, 2008);

[2] This corpus is a part of the *European Literary Text Collection corpus* (ElTEC) developed in the scope of the COST action 16204 *Distant Reading for European Literary History* (d-reading).

2. All words not detected by dictionaries are marked as potential results of OCR errors;
3. In these words one or two letters (in general, characters) that were identified as prone to false recognition are replaced with other letter(s) thus generating candidate words. This process is repeated for all letters that were identified as frequent sources of confusion. Moreover, the process is applied to already generated candidates but avoiding the circular replacements (e.g. 'п' → 'и' → 'н' → 'п'). Example: if ппво occurs in a text as an unknown word, then 'п' → 'и' is applied twice which results in a string *ппво*_*ипво*_*пиво*_иивo*, after that 'и' → 'н' is applied resulting in *ппво*_*ипво*_*пиво*_*иивo* _*нпво*_*пнво*_*ниво*_*инво*_*иивo*. After that rule 'н' → 'п' is not applied because it would generate same candidates.
4. Candidates are accepted if they represent words in SMD; others are rejected. For the above example, candidates пиво 'beer' and ниво 'level' would be accepted.

A few examples of the application of this procedure are given in Table 1.

Special attention is paid to hyphenated words. A hyphen in a Serbian OCR text can signify a word hyphenated at the end of the line or a hyphen in a multi-word. Our procedure first eliminates the hyphen and generates candidates by replacements, as illustrated by Table 1. For instance, in the case of Љу-бомнр, the hyphen is eliminated, and several candidates are produced: *_*Љубомпр*_***Љубомир***_*Љубомнр, one of which is accepted (the masculine first name *Ljubomir*). However, if this does not produce any valid word then the hyphen is retained and various replacements are applied to the component of the multi-word that is not a valid simple word (or to both of them). For instance, there are two incorrect letters in Нбрахпм-Хасан. With the elimination of a hyphen no valid candidates can be produced: *_*НбрахнмХаеапа*_*НбрахнмХасапа*_ *НбрахимХаеапа*_*НбрахимХасапа*_*НбрахнмХаеана*... If the hyphen remains, then corrections are attempted only for Нбрахпм since Хасан is a valid word (*Hasan*, a masculine first name): *_*Нбрахнм*_*Нбрахим*_*Нбрахпм*_... *Ибрахнм*_*Ибрахпм*_***Ибрахим***... One of the candidates is accepted – *Ibrahim*, a masculine first name.

The actual replacements are performed by finite-state transducers (FST) implemented in Unitex.[3] A separate FST is written for each replacement

[3] UnitexGramLab, a lexicon-based corpus processing suite.

a non-valid word	**candidates**	**the replacement in text**
One error – one candidate		
кнша	*_*киша*_*кшиа*_*кпша*_*кнша	кнша ⇒ +киша+
Two errors – one candidate		
прекннутп	*_*прскпиуги*_*прскпиути*_*прекпиуги*_...*прекинуги*_*прекинути*_*прскниуги*..._*ирекииути*_*ирскннугн*_*ирскннутн*...	прекннутп ⇒ +прекинути+
One error – two candidates		
погрешпо	*_*нотрснио*_*нотрсшно*_*ногрснио*_*погренио*_*погрешно*_*потрсшио*_*иотрешио*_*потрешио*_*погрешио*_*потрснио*_*потрсшно*...	погрешпо ⇒ +погрешно+погрешио+
One error – no candidates		
срдски	*_*срдскп*_*срдскн*_*ердски *_*ердеки*_*ердскп*_*срлски	***срдски...

Table 1. All candidates are separated with an underscore and delimited with asterisks. The valid replacements in a corrected text are surrounded with plus signs, words that could not be corrected are marked with asterisks. The incorrect letters are given in red, the accepted candidates are in blue.

pair. These FSTs are, however, very simple since they only give characters to be incorrect and character(s) that can replace them (see the top FST in Table 2). This simple FST invokes another generic FST which corresponds to the type of replacement (see the bottom FST in Table 2), and there are just a few such FSTs. The FST presented at the bottom in Table 2) works for the case when two letters should be replaced with one or more letters. The upper path in this FST works like this: the beginning of the word that needs to be corrected (words preceded with `*_*`) is assigned to variable `$1$`, while the last two letters are assigned to variable `$3$`. If the value of variable `$3$` is equal to letters that should be replaced (`$3.EQUAL=ul$`) then the output is produced that indicates the path that produced it (`*%1*`), followed by

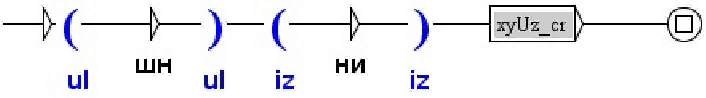
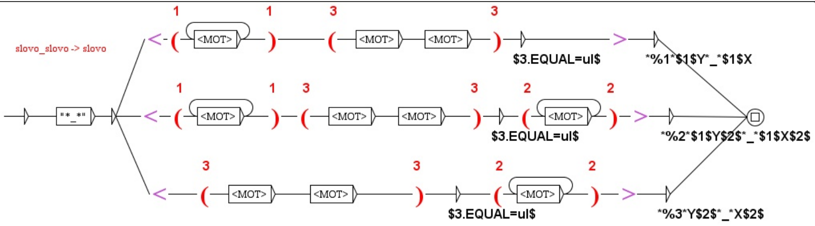
Letters *шн* to *ни*

Two letters (value of variable $ul$) are replaced by one or two letters (value of variable $iz$) – replacement int he opposite direction is not performed.
 slovo_slovo -> slovo "*_*" <MOT> $3.EQUAL=ul$ *%1*$1$Y*_*$1$X *%2*$1$Y2*_*$1$X2 *%3*Y2*_*X2

Table 2. A FST that initiates a correction for a specific replacement pair (top). A generic FST that performs a certain type of replacement (bottom).

two words: the original, input word (`$1$Y`) and the corrected, output word (`$1$X`).[4]

The application of these FSTs always leaves in a processed texts the original word to which a candidate is added. Also, FSTs for each replacement pair are iterated which enables the same corrections in one word: for instance, in сиежии three letters ‘и’ occur, first and second are incorrect and should both be replaced by ‘н’ yielding a valid word снежни – *snežni* ‘snowy’.

Just a few general FSTs (bottom of Figure 2) were developed for a few types of corrections – one or two character replacement, in one or both directions. Specific FSTs (top of same figure) were developed many – for various confusion pairs that can occur in an OCR text. Moreover, as each texts can bring its own problems, new specific FSTs can be easily produced and added to processing.

[4] Both of these words are modified in order to suppress the infinite modifications of same characters – letters not used in Serbian Latin alphabet Y and X replace original letters and their replacement. After all modifications of the same pair are done, the original characters are restored.

Table 3 illustrates the whole process, from a rough OCR output to a clean text.[5]

A text after OCR
- Е. *нпје него броћ! Тебе ће неко *еад *пптатн шта ти хоћеш, а *пгга нећеш! Него. кажи ти мени. је ли теби *бнла позната моја наредба, којом се забрањује тумарање по турским кућама? — *Нпје. — Ја где си ти *бно за ово месец дана — У *болннци.
A text after automatic correction
— Е. +++пије+++није+++ него броћ! Тебе ће неко +++сад+++ +++ништи+++пишти+++питати+++ шта ти хоћеш, а ***пгга нећеш! Него. кажи ти мени. је ли теби +++била+++ позната моја наредба, којом се забрањује тумарање по турским кућама? — +++Није+++. — Ја где си ти +++био+++ за ово месец дана — У +++болници+++.
A text after reading and correcting
— Е, није него броћ! Тебе ће неко сад питати шта ти хоћеш, а шта нећеш! Него, кажи ти мени, је ли теби била позната моја наредба, којом се забрањује тумарање по турским кућама? — Није. — Ја где си ти био за ово месец дана — У болници.

Table 3. Three phases of a text correction: 1) Marking of incorrect/unrecognized words by e-dictionaries; (2) Automatic correction by offering possible candidates; (3) Elimination of incorrect candidates and correction of remaining errors by a reader.

3 Diacritic restoration

The problem of diacritic restoration in Serbian occurs only for texts that are using Latin script. Diacritic omission happens when the Serbian Latin alphabet is reduced to the 26 letter Latin alphabet that can be accommodated by ASCII character set, and affects five letters: *š*, *đ*, *č*, *ć* and *ž*. As a result, some letters or letter groups become ambiguous:

– *c* – can stand for *č* and *ć* (and *c*);
– *z* – can stand for *ž* (and *z*);

[5] The example text is from the novel by Lazar Komarčić "One devastated mind" (Један разорен ум).

- *s* – can stand for *š* (and *s*);
- *dj* – can stand for *đ* (and *dj*);
- *dz* – can stand for *dž* (and *dz*).

When compared with the problem of OCR errors, it can be observed that the repertoire of these "errors"[6] as well as their possible corrections is limited and is known in advance. Namely, candidates for correction are only those that are represented in dictionaries. Similar to the OCR errors – a word that contains letters *c*, *s*, *z* or a digraph *dj* can be both correct and incorrect because a diacritic is missing, e.g. *kuca* can be correct (small dog) or a diacritic can be missing *kuća* (house). However, contrary to the OCR solution presented in the previous section, such words are treated as potentially incorrect and candidates for correction are offered, if they exist. Words that contain neither letters *c*, *z*, *s* nor digraphs *dj* and *dz* will be treated as correct and will not be subject to any correction, e.g. *voda* (water). The preposition of our procedure is that text contains no diacritics.

For each potentially incorrect word in the text the procedure will offer a list of all possible candidates:

- This list may contain the original word because it exists in dictionaries: *kupaca* ⇒ *kupača* (*kupač* 'bather'), *kupaća* (kupaći 'bathing'), *kupaca* (kupac 'buyer');
- It need not contain the original word (because it is not in dictionaries): *jezice* ⇒ *ježiće* (*ježiti se* 'to bristle' and *ježić* 'diminutive of hedgehog'), *jeziče* (*jezik* 'tongue'), *ježice* (*ježica* 'female hedgehog');
- If the list contains only the original word, and no other candidates, it will be accepted as correct.

For all potentially incorrect words procedure ranks its candidates according to the frequency of their occurrence in the Corpus of Contemporary Serbian (SrpKor).[7] For instance, the candidates for *kupaca* occur with following frequencies: *kupača* → 207, *kupaća* → 6, *kupaca* → 2202.

In order to be able to offer ranked candidates in diacritic restoration a special dictionary was produced from the standard Serbian Morphological Dictionaries, in which the entries for the forms *kupaca*, *kupača* and *kupaća* are as follows:

```
kupaca,kupac.N+Hum:mp2v
```

[6] We will call them "errors" although they are done intentionally.
[7] Corpus of Contemporary Serbian

```
kupača,kupač.N+Hum:mp2v:ms2v:ms4v
kupaća,kupaći.A:aefs1g:aefs5g:aenp1g:aenp4g:akms2g...
```

The construction of the dictionary to be used in diacritic restoration (SMD_DR) from SMD is performed by the following steps:

- All word forms containing at least one diacritic and all word forms containing at least one letter *c*, *s*, or *z* or a digraph *dj* or *dz* are extracted from SMD;
- Diacritics are removed from each word form that contains them, while the original word form is saved as the value of a new marker `+CR=`;
- All information that is not needed for this procedure is deleted (lemma, POS, syntactic and semantic markers, etc.);
- Information about the frequency of the original word form in SrpKor is added;
- Information for same word forms is merged.

This process for the previously examples is illustrated in Table 4.

`kupaca,kupac.N+Hum:mp2v`	`kupaca,.X+CR=kupaca(21)`	⇒ SMD_DR
`kupača,kupač.N+Hum:mp2v`	`kupaca,.X+CR=kupača(2)`	
`kupaća,kupaći.A:aefs1g`	`kupaca,.X+CR=kupaća(1)`	
`kupaca,.X+CR=kupaca(21)_kupača(2)_kupaća(1)`		

Table 4. Production of an entry in SMD_DR

Relative frequencies in SrpKor were calculated and assigned to each candidate for correction in order to facilitate calculation and usage. For instance, relative frequency 1 was assigned to tokens that had in the corpus absolute frequency $[1, 10]$.More about this can be read in (Krstev et al., 2018).

The produced SMD_DR has almost one million entries (word forms that are candidates for correction), of which 95.1% have only one candidate for correction, 4.4% have two candidates, and remaining the 0.5% have 3 or more candidates. However, in a number of cases when more than one candidate exists, all candidates occur with the comparable frequency which makes it difficult to choose the right one despite the ranking. Some of difficult cases are:

- `čašu(10)\_času(28)\_ćasu(1)` (candidates are forms for *čaša* ‘glass’, *čas* ‘hour/moment’, *ćasa* ‘bowl’);
- `reci(24)\_reči(261)\_reći(145)` (candidates are forms for *reka* ‘river’, *reč* ‘word’, *redak* ‘line’ and *reći* ‘to say’);
- `veće(155)\_veče(34)\_vece(1)` (candidates are forms of *veće* ‘council’, *veći* ‘bigger’, *veče* ‘evening’ and *vece* ‘WC’).

In order to reduce the number of multiple candidates, procedure uses some additional resources.

- A list of 30 most frequent trigrams obtained from SrpKor in which at least one word would contain letters *c*, *s*, *z* or a digraphs *dj* if diacritics were removed; for instance, *zbog toga sto* ‘because of that’ ⇒ *zbog toga što*, thus avoiding multiple candidates for *sto* (*sto* ‘table/hundred’ and *što*, a functional word that can be an adverb, a pronoun or a conjunction);
- A list of 50 most frequent bigrams obtained from the SrpKor in which at least one word would contain letters *c*, *s*, *z* or a digraph *dj* if diacritics were removed; for instance, *znaci da* ‘meaning that’ ⇒ *znači da*, thus avoiding multiple candidates for *znaci* (a form of *znak* ‘sign’ and *značiti* ‘to mean’);
- A dictionary of multi-word units (MWU) (nouns, adjectives, adverbs, pronouns, conjunctions and interjections) obtained from a dictionary of more than 18,000 MWU lemmas; for instance, *Dobro vece* ⇒ *Dobro veče* ‘Good evening’ (avoiding multiple candidates for *vece*) or *ukrstene reci* ⇒ *ukrštene reči* ‘cross-words’ (avoiding multiple candidates for *reci*, but also resolving the preceding adjective *ukršten*). These dictionary contains all inflective forms,so, for instance, *ukrstenih reci* (the ginitive form) would be corrected as well – *ukrštenih rećі-.*

After the application of the diacritic restoration procedure, a new version of the text is obtained which contains, for each word form in the original text that contains letters *c*, *s*, *z* or digraphs *dj*, *dz*, a list of zero[8] or more candidates obtained from the dictionary SMD_DR, or one candidate obtained from lists of trigrams or bigrams, or a dictionary of MWUs for a sequence of words. The result of the application of the procedure to a sample text is given in Table 5.[9]

[8] This list is empty if a word from neither with diacritics/digraphs nor without them exists in SMD.

[9] The excerpt is taken from the novel “Komo” by Srđan Veljarević.

the source text	the text with suggested corrections
KGB mu je ponudio da saradjuje s njima, i da ce mu onda knjige biti objavljivane. Brodski je odbio. I nije mogao da objavljuje. Posle nekog vremena predlozili su mu da napusti zemlju, i da ce tako biti najbolje, za njega i za drzavu. Brodski je seo u avion za Bec. Poneo je pisacu masinu, nesto odece, zbirku poezije Dzona Dona, i flasu votke, poklon za pesnika Vinstana Odna, koji ga je docekao na beckom aerodromu.	KGB mu je ponudio da *5a(sarađuje(25)) *2(i da će) mu onda knjige biti objavljivane. V_*5b(Brodski(2)_brodski(3)) je odbio. I nije mogao da objavljuje. V_*5b(pošle(1)_posle(1422)) nekog vremena *5a(predložili(12)) su mu da napusti *5b(zemlju(137)_žemlju(1)), *2(i da će) tako biti najbolje, za njega i za *5a(državu(81)). V_*5b(Brodski(2)_brodski(3)) je seo u avion za V_*5a(Beč(22)). Poneo je *4(pisaću mašinu(0)), *5a(nešto(515)) *5a(odeće(13)), zbirku poezije *5a(Džona(17)) Dona, i *5a(flašu(4)) votke, poklon za pesnika Vinstana Odna, koji ga je *5a(dočekao(12)) na *5a(bečkom(5)) aerodromu.

Table 5. The output of the procedure for diacritic restoration

The output presented in Table 5 is only an intermediate result which has to be further transformed into a corrected text. Information in this intermediate output – a list of candidates, their relative frequency in the reference corpus, an indication of the type of correction (e.g. *4 indicates the dictionary of MWUs, *2 the list of trigrams, *5 the dictionary SMD_DR, etc.) – can be used in the cleaning phase. The cleaning can be very simple by accepting all suggestions – when there is more than one candidate choose one with the higher frequency – or it can be more sophisticated. In the later case, when there is more than one candidate the procedure could choose one if it has a significantly higher relative frequency, e.g. at least 10 or 100 times. In the example given, for the two cases of multiple candidates: *5b(pošle(1)_posle(1422)) and *5b(zemlju(137)_žemlju(1)),[10] both criteria would chose *posle* and *zemlju*, respectively, which would be the right choice in both cases. The procedure can also look in the broader context and apply some decision rules. For instance, for a word form *celu* there are three candidates: čelu(95), ćelu(1), celu(44) (forms of *čelo* ‘forehead/cello’, *ćela* ‘bold

[10] Actually, there is one more case of multiple candidates, V_*5b(Brodski(2)_brodski(3)), stemming from two different entries in SMD: *Brodski*, a surname, and *brodski* ‘like a boat’; however, the correction result is the same – no correction, a word form *Brodski* remains.

head', *ceo* 'whole', respectively). If this word form appears after the preposition *za* that demands the genitive, the accusative, or the instrumental case, then *čelu* would be discarded (being the dative or the locative form of *čelo*) despite having the highest relative frequency. If some multiple candidates cannot be resolved by rules nor by frequencies, they remain in the text for the user to chose the right one.

4 Switching between two Serbian pronunciation variants

In Serbian, two standard variants of pronunciation are in use, Ekavian and Ijekavian. They differ in the reflection of the old Proto-Slavic phoneme (*jat*): in the Ekavian variant it is replaced predominantly by *e*, while in the Ijekavian variant it is replaced by syllables *ije/je*, and sometimes *i*. A Serbian text is usually written in one of these variants.

Sometimes it is desirable to transform text written in one pronunciation into another. Although this is not an issue related to errors, the problem is similar. Namely:

– When transforming an Ekavian text to Ijekavian, for each word containing the letter *e* it must be determined whether it is a reflection of the phoneme *jat* and that it should therefore be replaced by an Ijekavian variant containing *ije/je/i*;
– When transforming an Ijekavian text to Ekavian, for each word containing *ije/je/i* it must be determined whether it is a reflection of the phoneme *jat* and that it should therefore be replaced by an Ekavian variant containing *e*.[11]

The problem, though different in nature, has similarities with problems of OCR error correction and diacritic restoration:

– Like in the case of diacritic omission "errors" are limited to a small number of letters and/or syllables, which implies that a dictionary solution might be appropriate;

[11] The problem is further complicated by morphological alternations, e.g. *nežan* vs. *nježan* 'tender' and *leto* vs. *ljeto* 'summer'. In the case of Ijekavian *nj* and *lj* are not consonant groups but dighraphs whose corresponding Cyrillic letters are *њ* and *љ*. Although our procedure also deals with such cases they are not explained here for reasons of simplicity.

– Similar to problems of OCR error correction and diacritic restoration, an *e* in an Ekavian text word can be a reflection of *jat* (*reka* $\Longleftrightarrow$ *rijeka* ‘river’) or not (*zeka* ‘bunny’); a syllables *ije/je* in an Ijekavian text word can be a reflection of *jat* (*snijeg* $\Longleftrightarrow$ *sneg ‘snow’* and *mjesec* $\Longleftrightarrow$ *mesec* ‘month/moon’) or not (*sujeta* ‘vanity’ and *prijem* ‘reception’).

For the problem of switching between two pronunciations two systems were developed: the first one transforms an Ekavian text to its Ijekavian version, the other transforms an Ijekavian text to its Ekavian version. These systems work in the similar way as the system for diacritic restoration, and each of them uses its own dictionary for transformation.

Ekavian	Ijekavian	translation
`reka,N612+Ek`	`rijeka,N612+Ijk`	‘river’
`zeka,N741+Zool`		‘rabbit’
`sneg,N291+Ek`	`snijeg,N291+Ijk`	‘snow’
`prijem,N1`		‘reception’
`mesec,N9+Ek`	`mjesec,N9+Ijk`	‘moon/month’
`sujeta,N600`		‘vanity’

Table 6. Pronunciation markers in SMD

Dictionaries for variant transformation, as in the case of the dictionary for diacritic restoration SRP_DR were obtained from the standard SMD. Pronunciation variants in the standard SMD are marked but not connected (this is explained in more details in (Lazić and Škorić, 2019) in the same issue). The marker `+Ek` denotes Ekavian specific lemmas, and the marker `+Ijk` Ijekavian specific lemmas, while entries for lemmas that do not contain a reflection of *jat* do not have a variant marker, as illustrated in Table 6.

As explained in (Lazić and Škorić, 2019), specific rules were developed for linking corresponding Ekavian and Ijekavian entries. This enabled the production of two specific dictionaries: Ijk2Ek for transforming an Ekavian text to Ijekavian, and Ek2Ijk for the transformation in the other direction. Lemmas that are same in both pronunciations are not represented in these dictionaries because for them no transformation is needed - as in the case of the dictionary for diacritic restoration in which entries that do not contain *c*, *s*, *z* and *đ* are omitted. Examples of entries in these two dictionaries are represented in Table 7.

Ijk2Ek	Ek2Ijk
`rijeka,.X+EK=reka(865)`	`reka,.X+IJK=rijeka(235)`
`rijekama,.X+EK=rekama(121)`	`rekama,.X+IJK=rijekama(34)`
`snijeg,.X+EK=sneg(473)`	`sneg,.X+IJK=snijeg(129)`
`snijega,.X+EK=snega(298)`	`snega,.X+IJK=snijega(97)`
`mjesec,.X+EK=mesec(2282)`	`mesec,.X+IJK=mjesec(644)`
`mjeseca,.X+EK=meseca(3922)`	`meseca,.X+IJK=mjeseca(3955)`

Table 7. Dictionary entries in dictionaries for transformation form Ijekavian to Ekavian and vice versa.

Frequencies incorporated in them for the selection of candidates are obtained from two different corpora – one containing only texts written in Ekavian pronunciation and the other containing only texts written in Ijekavian pronunciation. In the case of multiple corrections, they are merged in one entry, as in the SRP_DR dictionary. Specific problems may arise with multiple corrections when transforming texts in either direction:

– An Ijekavian form is a homograph of a form that does not contain Ijekavian *ije/je*:
  - `njega,.X+Ijk+EK=nega(56)_njega(5459)` (*njega/nega* 'nursing' vs. *njega* 'him');
  - `bolje,.X+Ijk+EK=bole(42)_bolje(4279)` (*bolje/bole* a form of *boljeti/boleti* '(they) hurt (aorist)' vs. *bolje* 'better').
– An Ekavian form is a homograph of a form that does not contain Ekavian *ije/je*:
  - `beg,.X+IJK=bijeg(78)_beg(15)` (*beg/bijeg* 'runaway' vs. *beg* 'bey');
– An Ekavian form has two or more different Ijekavian forms:
  - `cedila,.X+IJK=cjedila(0)_cijedila(5)` (the genitive singular form of *cjedilo* 'strainer' vs. the active participle of *cijediti* 'to strain');
  - `posede,.X+IJK=posjede(10)_posijede(0)_posijedje(0)_posjedje(0)` (forms of the noun *posjed* 'property' and tree different verbs: *posijedjeti* 'grow white', *posjedjeti* 'sit down', *posjesti* 'assign a seat').

– For an Ekavian form list of candidates more than one Ijekavian form exisr as well as a form that is not affected by different pronunciation:
  - `bega,.X+IJK=bijega(42)_bega(9)_bjega(0)` (*bega/bijega* 'runaway (genitive)' vs. *bega* 'bey (genitive)') vs. *bjega* 'to runaway (aorist)'.

Both systems use the same procedure for detecting words that should be "corrected" and producing lists of candidates for replacement, although, obviously, the resources they use are different. The results produced by two systems on two sample texts are presented in Table 8. In the case of Ijekavian source sample, three word forms had to be replaced – *vjerovatno* 'probably', *izmjene* 'change', *cijena* 'price' – and they were all correctly replaced by one offered solution. In the case of the Ekavian source sample, three different word forms had to be replaced – *posledica* 'consequence', *delu*, a form of *delo* 'work' and *deo* 'part', and *rešenje* 'solution'. Two forms were correctly replaced by one offered solution – *posledica* and *rešenje* – while the third form *delu*, which is homographous in Ekavian, has two different corresponding forms in Ijekavian – *dijelu* and *djelu* – and the system could not decide using rules what would be the right choice. Namely, a phrase *u nekom delu* is correct for both meanings of the form *delu* ('in some part' and 'in some work'). The frequency of use of forms *dijelu* and *djelu* in the Ijekavian corpus was not enough in favor of any of them (the right choice in both cases in the sample is *dijelu*).

5 Implementation and results

All systems for text mending presented in sections 2–4 use similar resources and are built using similar solutions:

– Electronic dictionaries are used to detect words that are candidates for change and to offer possible corrections. They are also used to build special dictionaries for solving some concrete problems (diacritic restoration, language variant switching).
– Detecting words that are candidates for change as well as the production of lists of candidates for replacement is done by finite-state transducers implemented in Unitex software (Paumier et al., 2016).
– All presented systems consist of two independent parts, both of which are implemented as cascades of finite-states transducers – in these cascades each FST works on a texts produced by a FST that directly precedes it

Ijekavian ⇒ Ekavian	
the source text	the output text
"Na Medicinskom fakultetu u Foči ove godine u planu je multidisciplinarni program pa će moći da konkurišu i studenti sa srodnih fakulteta, a što se tiče ostalih, vjerovatno bi u narednom periodu moglo doći do izmjene Pravilnika po tom pitanju. Zasad ostaje ovako", kaže on. Cijena školarine za godinu kreće se od 2.500 KM pa naviše.	"Na Medicinskom fakultetu u Foči ove godine u planu je multidisciplinarni program pa će moći da konkurišu i studenti sa srodnih fakulteta, a što se tiče ostalih, verovatno bi u narednom periodu moglo doći do izmene Pravilnika po tom pitanju. Zasad ostaje ovako", kaže on. Cena školarine za godinu kreće se od 2.500 KM pa naviše.

Ekavian ⇒ Ijekavian	
the source text	the output text
„Ne razmatramo mogućnost da Tanjug ponovo počne radi kao vladin medij, nego ispitujemo sve okolnosti koje će dovesti do nekih od mogućih posledica, a to je da Tanjug bude vladin medij u nekom delu, da Tanjug uopšte ne bude vladin medij u bilo kom delu i neko treće rešenje koje je između ta dva”, kaže ministar.	„Ne razmatramo mogućnost da Tanjug ponovo počne radi kao vladin medij, nego ispitujemo sve okolnosti koje će dovesti do nekih od mogućih posljedica, a to je da Tanjug bude vladin medij u nekom *(dijelu(751)_djelu(59)), da Tanjug uopšte ne bude vladin medij u bilo kom *(dijelu(751)_djelu(59)) i neko treće rješenje koje je između ta dva”, kaže ministar.

Table 8. The output of two procedures for switching from Ijekavian to Ekavian and vice versa.

and produces a new text for a FST that follows. These cascades are also implemented in Unitex (Friburger and Maurel, 2004). The first cascade produces an intermediate result with lists of all possible candidates, as illustrated in Table 5 in Section 3. The second cascade eliminates some or all multiple candidates. These two cascades are independent in all systems, which makes it easy to produce the second cascade as strict, relaxed or somewhere in between.

A similar approach that also relies on electronic dictionaries and FSTs implemented in Unitex was used for vowel restoration in Arabic (Neme and

Paumier, 2019). The approach presented in this paper is specific since it offers a solution for solving three different tasks.

Not all of the systems presented were fully evaluated. The system for OCR correction is difficult to evaluate since each text (especially in the case of relatively old books) poses different problems and different lists of solutions may be offered. However, results after correcting some thirty novels show that after applying the system for OCR error correction the number of errors (precisely, unknown words) may be reduced by 5% to up to almost 90%, which depends, naturally, on the initial number of errors

A thorough evaluation results for the system for diacritic restoration was presented in details in (Krstev et al., 2018). The authors showed that on the average the precision of the system is $P = 98.93\%$, the recall $R = 94.94\%$ and $F_1 = 96.90\%$ when calculated on all occurrences (tokens) of words that were candidates for correction (“suspicious” words).

The system for transforming texts from Ekavian to Ijekavian pronunciation and vice versa was not evaluated since it is not fully developed yet. Namely, the Ijekavian corpus that was used to calculate frequencies needs to be enlarged and enriched with more versatile texts.

6 Future Work

In this paper we presented three operational systems for three text mending tasks for Serbian: correction of OCR errors, diacritic restoration and switching between two pronunciations. Although these systems were already successfully used for various tasks (for corrections of OCR errors see (Jaćimović, 2019) and for diacritic restoration see (Petković, 2019)) there is still much to be done. The treatment of unknown words in processed texts as well as the elimination of multiple candidates needs to be further developed. Options of using a hybrid approach that would merge a dictionary with machine-learning will be explored. Finally, a user-friendly interface that will enable the use of these systems on the Web is already under development.

Acknowledgment

This research was partly supported by the Ministry of Education, Science and Technological Development through projects 138006 and III47003.

References

Blair, Charles R. "A program for correcting spelling errors". *Information and Control* Vol. 3, no. 1 (1960): 60–67

Bledsoe, W. W. and I. Browning. "Pattern Recognition and Reading by Machine". In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern). New York, NY, USA: ACM, 1959, 225–232. http://doi.acm.org/10.1145/1460299.1460326

Friburger, Nathalie and Denis Maurel. "Finite-state transducer cascades to extract named entities in texts". *Theoretical Computer Science* Vol. 313, no. 1 (2004): 93–104

Henton, Caroline G. "Automated transformation from American English to British English", 2004, uS Patent 6,738,738

Jaćimović, Jelena. "Textometric methods and the TXM platform for corpus analysis and visual presentation". *Infotheca – Journal for Digital Humanities* Vol. 19, no. 1 (2019): 30–54. https://infoteka.bg.ac.rs/ojs/index.php/Infoteka/article/view/2019.19.1.2_en

Kolak, Okan and Philip Resnik. "OCR error correction using a noisy channel model". In *Proceedings of the second international conference on Human Language Technology Research*, 257–262. Morgan Kaufmann Publishers Inc., 2002

Krstev, Cvetana. *Processing of Serbian – Automata, Texts and Electronic dictionaries*. Faculty of Philology, University of Belgrade, 2008

Krstev, Cvetana, Ranka Stanković and Duško Vitas. "Knowledge and Rule-Based Diacritic Restoration in Serbian". In *Proceedings of the Third International Conference Computational Linguistics in Bulgaria (CLIB 2018)*, Koeva, Svetla. Sofia, Bulgaria: Institute for Bulgarian Language "Prof. Lyubomir Andreychin", 41–51. Bulgarian Academy of Sciences, 2018

Kukich, Karen. "Techniques for automatically correcting words in text", *Acm Computing Surveys (CSUR)* Vol. 24, no. 4 (1992): 377–439

Lazić, Biljana and Mihailo Škorić. "From DELA based Dictionary to Leximirka Lexical DataBase". *Infotheca – Journal for Digital Humanities* Vol. 19, no. 2 (2019): 00–00, https://infoteka.bg.ac.rs/ojs/index.php/Infoteka/article/view/2019.19.2.4_en

Miller, George A and Elizabeth A Friedman. "The reconstruction of mutilated English texts". *Information and Control* Vol. 1, no. 1 (1957): 38–55

Neme, Alexis Amid and Sébastien Paumier. "Restoring Arabic vowels through omission-tolerant dictionary lookup". *Language Resources and Evaluation* (2019): 1–65

Ostrogonac, Stevan, Branislav Popović and Robert Mak. "The Use of Statistical Language Models for Grammar and Semantic Error Handling in Spell Checking Applications for Serbian". In *12 th International Conference on Electronics, Telecommunications, Automation and Informatics, ETAI*, 2015

Paumier, Sébastien, Sebastian Nagel Marschner and Johannes Stiehler. "UNITEX 3.1 User Manual". *Université Paris-Est Marne-la-Vallée.* (2016)

Peterson, James L. "Computer Programs for Detecting and Correcting Spelling Errors". *Commun. ACM* Vol. 23, no. 12 (1980): 676–687. http://doi.acm.org/10.1145/359038.359041

Petković, Ljudmila. "Creation and Analysis of the Yugoslav Rock Song Lyrics Corpus from 1967 to 2003". *Infotheca – Journal for Digital Humanities* Vol. 19, no. 1 (2019): 5–29. https://infoteka.bg.ac.rs/ojs/index.php/Infoteka/article/view/2019.19.1.1_en

Salloum, Wael and Nizar Habash. "Elissa: A dialectal to standard Arabic machine translation system". In *Proceedings of COLING 2012: Demonstration Papers*, 2012, 385–392