INVESTIGATIONES LINGUISTICAE VOL. XXXVII, 2017
© INSTITUTE OF LINGUISTICS – ADAM MICKIEWICZ UNIVERSITY
AL. NIEPODLEGŁOŚCI 4, 60-874, POZNAŃ – POLAND

Automatic Diachronic Normalization of Polish Texts

Jassem Krzysztof, Graliński Filip, Obrębski Tomasz and Wierzchoń Piotr

ADAM MICKIEWICZ UNIVERSITY, POZNAŃ

jassem@amu.edu.pl

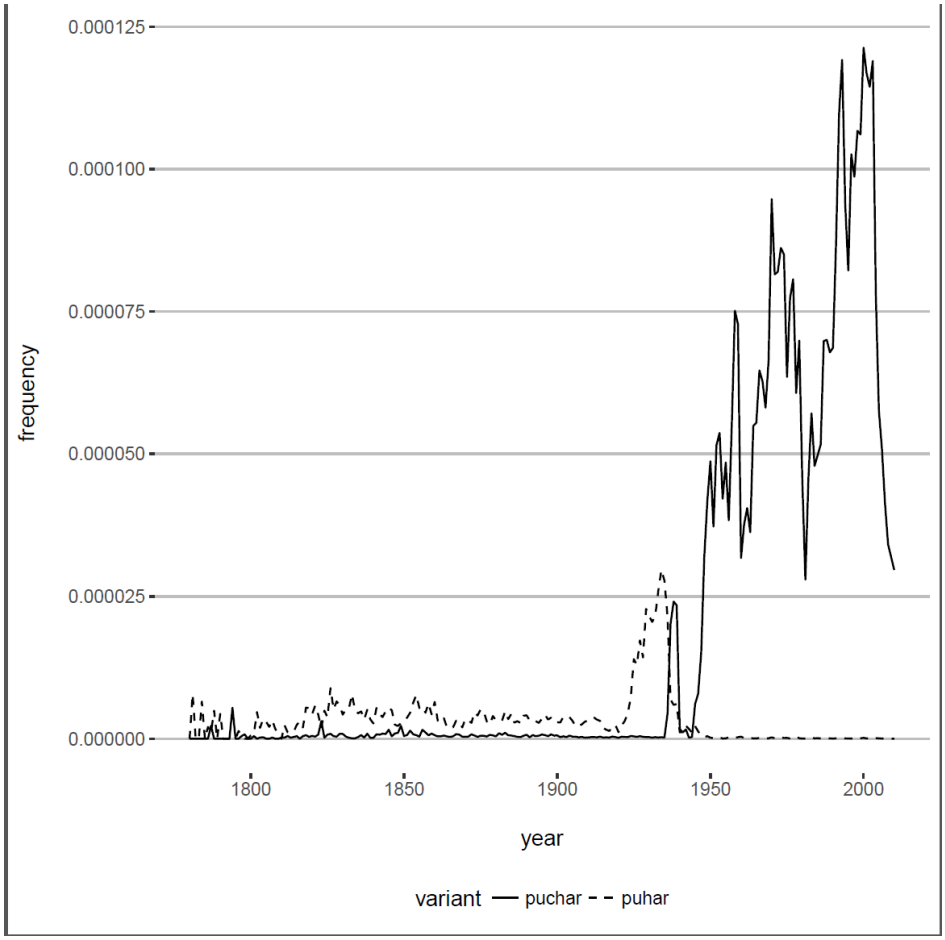
Abstract

The paper presents a method for the automatic diachronic normalization of Polish texts – the procedure, which, for a given historical text, returns its contemporary spelling. The method applies finite-state transducers, defined in a sublanguage of the Thrax formalism. The paper discusses linguistic issues, such as evolution in spelling of the Polish language, as well as implementation aspects, such as efficiency or testing the proposed method.

1. Introduction

Studies on orthography, particularly for the Polish language, are usually focused on changes in spelling over the course of time. The process of creating orthographical rules can be spontaneous or prescriptive. Spontaneous changes are indicated by a text corpus and the spelling rules are decided by language users (e.g. writing the particle *nie* (English: *not*) jointly or separately with participles). In the primary phase of its development, Polish orthography was strongly affected by technical limitations (e.g. sets of fonts) and preferences of certain Polish printers (Lisowski 2010). On the other hand, prescriptive changes (e.g. replacement of *j* by *i* in the Polish word *Anglia* (English: *England*)), represented by explicit sets of norms, are formulated by institutions constituted specifically for the task. Such organizations are usually tied to academia and, in Poland, usually represented by prominent professors of linguistics.

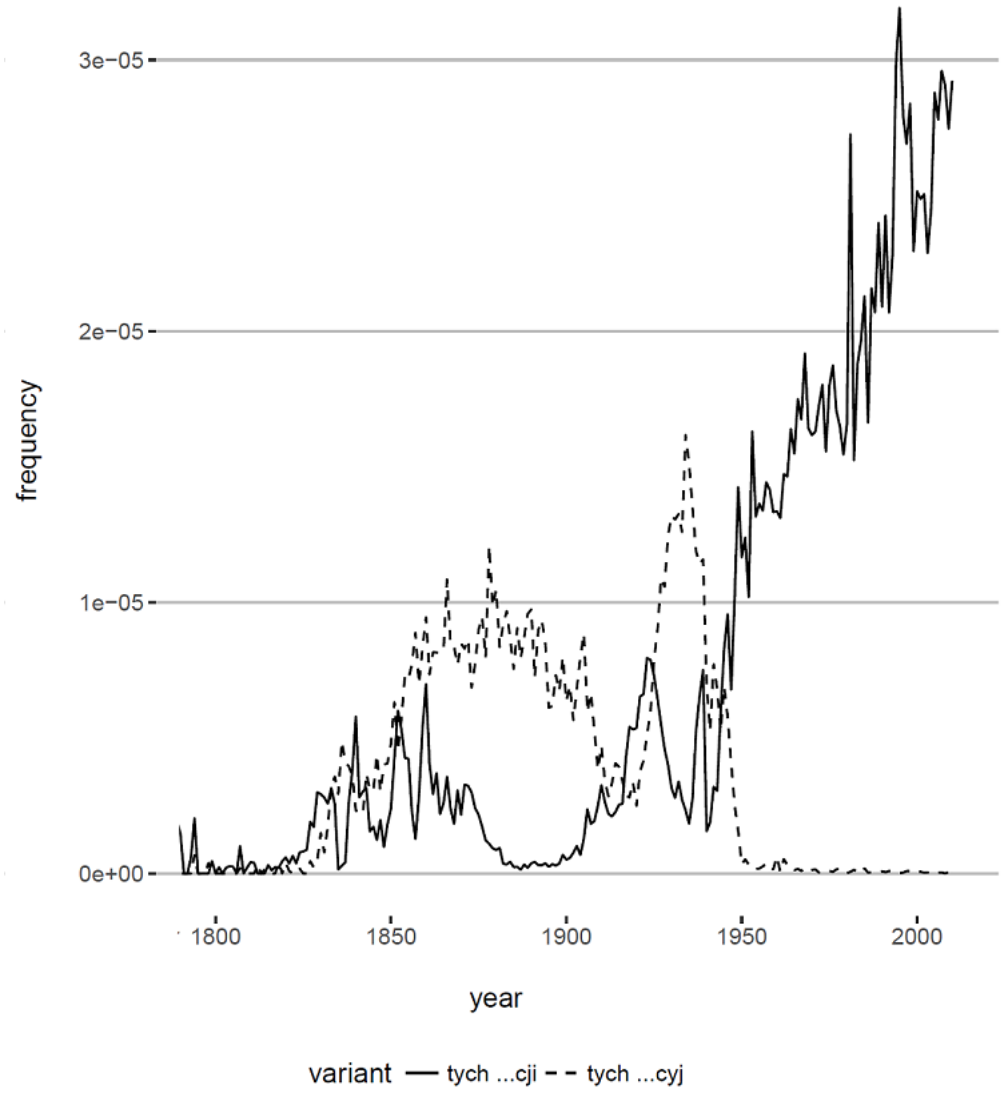
Picture 1. visualises a change in the spelling of the word *puchar* (English: *goblet*) due to a very controversial prescriptive decision made as part of the 1936 spelling reform. The horizontal axis indicates time flow and the vertical axis shows the usage of a word in the language modelled by a frequency of appearance in a versatile text corpus (Graliński, 2013).



Picture 1. Frequency oF differEnt spellings of the word "puchar"

The contemporary spelling (*puchar)* was less frequent before 1936, it was only after the spelling reform when the *puchar* spelling replaced *puhar*. After 1950 the previous spelling all but disappeared.

Picture 2. shows the evolution in spelling of words, which in their genitive plural forms ended with “cyj” in the past but their contemporary spellings end with “cji”. An example of such a word is *kolacyj – kolacji* (genitive plural form of *supper*).



Picture 2. Frequency of words ending with "cji" vs "cyj"

It may be noticed that the contemporary spelling was also dominant around 1800. In the 19th century the "cji" spelling occurred more frequently than "cyj". The two lines cross around 1940 (where both spellings were used with similar frequency), since when the contemporary spelling has been significantly dominant. It is worth noting that the spontaneous shift followed the prescriptive change in 1936[1], which ordered the usage of the "j" letter in the combination with the consonant "c".

[1] S. Jodłowski, W. Taszycki, *Zasady pisowni polskiej i interpunkcji ze słownikiem ortograficznym,* wyd. 2, rozszerzone według uchwał komitetu Ortograficznego Polskiej Akademii Umiejętności z 21 kwietnia 1936 r.

This paper presents an algorithmic approach to the evolution of the Polish orthography, focusing on changes in the 19th and 20th centuries. Broadly speaking, the goal of the study is to provide new philological and historical (what changes took place and when) as well as operational (the construction of an algorithm) insights.

Our research aims at the creation of an algorithm, which, for a given historical Polish text, returns its contemporary spelling. We call this process, **diachronic normalization**. The process is executed based on a set of prepared time-labelled transformation rules, which will further enable the user to determine the origin date of an analysed document.

We believe that the research under description will have a significant impact on the development of Polish humanities (history, sociology, linguistics, etc.) because it will:

1) allow linguists to verify both qualitative and quantitative hypotheses concerning the developmental processes of vocabulary and, as such, will serve as a basis for humanistic research,
2) become a source of information for historians, sociologists and cultural specialists (who will be helped in searching texts written in old orthography without any prior knowledge of the orthography of the time),
3) help improve methods dealing with Natural Language Processing.

An important aspect of this study is that it will make it possible to ascertain the time in which the text was created. This will automate the process of language chronologisation, or **linguochronologisation.** The term denotes assigning a piece of chronological information to an object of a linguistic study (a word, a syntagma, etc.). This piece of information is simply a unit of time, i.e. a year, month or even a minute. The chronological accuracy of time description, depends on the adopted time parameter. The predominant parameter in literature is a year (1901, 1939, 1953, etc.). Linguochronologisation helps ascertain which objects do not appear in a linguistic model before a predetermined date. This, in turn, allows for the description of language evolution, especially from the lexical or morphological perspective. (An example of morphologically-motivated research might be the analysis of appearance, in Polish texts, of affixes such as: *anty-* (English: *anti-*), super- (English: *super-*), krypto- (English: *crypto*)), Chronologisation of linguistic objects of interest allows for the determination of language peculiarities against the flow of time.

To test the prepared algorithms, a corpus of time-labelled historical texts has been collected. A part of the corpus was acquired relatively easily: Some texts had been equipped with metadata in the Dublin Core standard (with additional date-of-publication – although this information is sporadically false,

inaccurate or using an atypical convention), a part of the material came from digital libraries (most of which are based on the dLibra system). Although the challenge of acquiring the texts (with corresponding metadata) is more technical than academic, it was nevertheless necessary to overcome it to create a core set of data, on which the algorithms described below have been tested.

The reported task was not limited just to the relatively well structured resources of digital libraries. Different versions of the same text (e.g. *The Lord's Prayer*, *Pan Tadeusz* by A. Mickiewicz, etc.) are usually referred to in various ways. Comparing the orthography of these texts from different time periods allowed for the extraction of fundamental differences in the spelling of Polish words.

The study is focused on the description of orthographical rules from the 19th and 20th centuries. In principle, most of the 19th century is characterised by a lack of standard orthographical rules for the Polish language. The situation did not change after Poland regained independence in 1918. The first conscious and widely spread orthographical reform took place in 1936. An Orthographical Committee (*Komitet Ortograficzny*) was formed by prof. Jan Rozwadowski. After his death the committee was headed by prof. Kazimierz Nitsch. The most important decisions of the committee were:

1) the spelling of foreign loanwords, such as *ortografia* (English: *orthography)* or *radio* (English: *radio*) should use "i" instead of "j", unless preceded by an "s", "z" or "c", as in words: *kompresja* (English: *compression*), *daglezja* (English: *douglas fir*) or *lekcja* (English: *lesson*);
2) inflected adjectives forms should end with "-ym / -im" in the singular number (e.g. *dobrym*, English: *good*), and end with "-ymi /-imi" in the plural number (*dobrymi*), rather than the with the older morphemes *"-em"* (singular) or *"-emi"* (plural);
3) the spelling of clusters "ke" and "ge" should be differentiated in "foreign sounding" words such as "ke" in *kelner* (English: *waiter*) or "ge" in *geometria* (English: *geometry*) from "fully assimilated words". e.g. "kie" as in *kielnia* (English: *trovel*) or "gie" as in *giełda* (English: *stock exchange*).

Furthermore, a set of rules was introduced, which determine whether specific word chunks should be written separately or jointly – the rules gave preference for separated spelling, e.g *przede wszystkim* (English: *above all*) or *na pewno (*English: *for sure)*. Additionally, the use of upper-cased and lower-cased letters was re-defined. The rules put forward by the Orthographical Committee applied until 1996, when a set of new rules was developed, including the joint writing of the particle *nie* (English: *not*) with participles, e.g. *nieśpiewany, nienarysowany* (the scope of this spelling reform was much smaller than that of 1936).

2. Related work

The problem of transliteration of Polish historic texts was first tackled by Jakub Waszczuk in 2012[2] for the sake of the SYNAT project[3] (a reference to the SYNAT project may be found in (Mykowiecka, Rychlik, Waszczuk, 2012), where the authors report on the paper-into-electronic conversion of an Old Polish dictionary). The solution proposes a formalism, as well as its interpreter, called *hist-pl-transliter*, designed for the creation of transliteration rules, which allows for:

- Substitution of character strings
- Definition of character classes
- Concatenation of character strings or character classes
- Alternative of character strings or character classes

Additionally, the author of the formalism publicizes a list of rules that have been used for the IMPACT project[4], which consisted in the preparation of full-text versions of Polish historical texts stored in selected Polish digital libraries. The rules focus mainly on transliteration of diacritic letters, non-existing in the contemporary writing (such as "à" "á" "á" "â", "ã" "ä", "ȧ"), but a few context-dependent rules are included there as well, as shown in Listing 1.

```
(1)  , vowel >+> ("y" #> "j")

(2)  , vowel >+> ("i" #> "j") >+> vowel

(3)  , hardConsonant >+> ("y" #> "yj") >+> vowel

(4)  , softConsonant >+> ("y" #> "ij") >+> vowel
```

Listing 1. Context dependent rules in the hist-pl-transliter package

The rules in Listing 1. are processed by the interpreter in the following way:

(1): substitute “y” by “j”, if preceded by a vowel;

(2): substitute “i” by “j”, if preceded by a vowel and followed by a vowel;

(3): substitute “y” by “yj”, if preceded by a “hard consonant” and followed by a vowel;

[2] https://hackage.haskell.org/package/hist-pl-transliter

[3] http://synat.nlp.ipipan.waw.pl/

[4] http://dl.psnc.pl/activities/projekty/impact/results/

(4) substitute "y" by "ij", if preceded by a "soft consonant and followed by a vowel.

The most recent effort in the area has been carried out within the KORBA project[5] (Bronikowska, Radziejewski, 2017). The task consists in the transliteration of Polish texts originating from the 17th and 18th centuries. The solution applies the AMEBA SUPERTOOL[6], originally designed for the aforementioned IMPACT project. The solution applies a large set of context-dependent rules defined by means of regex-based formalism, as shown in Listing 2.

```
(1)     ^u      mowic       $       mówić
(2)     ^wy     mow         i*$     mów
```

Listing 2. Context-dependent rules in the AMEBA SUPERTOOL

The rules in Listing 2 are interpreter by AMEBA SUPERTOOL as follows:

(1): Substitute „mowic by mówić", if preceded with the letter „u", which begins the word, and followed by the end of the word;

(2) Substitute „mow by mów", if preceded with the string „wy", which begins the word, and followed by the letter "I" (which may or may not end the word).

3. Rule-based Diachronic normalization

3.1. Source of Rules

The set of rules for diachronic normalization is based on the historical description of Polish orthographic changes presented in (Malinowski, 2012). Our task consisted in:

- creating a machine-readable rule formalism,
- re-writing the orthographic rules in the created formalism.

One of the orthographic changes is defined in (Malinowski, 2012) as follows: "Introduction of the letter *j* in non-initial syllables of foreign-origin words". A part of machine-readable re-formulation of the rule is given in Listing 5.

Our approach resembles the one proposed in *hist-pl-transliter*, as it requires less human effort in rules creation than that of AMEBA SUPERTOOL. In comparison to *hist-pl-transliter* our solution delivers some new operators, applies an up-to-date package for the processing of finite-state transducers,

[5] http://clip.ipipan.waw.pl/KORBA

[6] https://bitbucket.org/jsbien/pol

suggests treatment of "exception words" and takes advantage of the Malinowski's work.

3.2. Finite State Transducers

Diachronic normalization transforms one character string into another. The most common approach to automatize such a task is the usage of finite-state transducers (FSTs). An FST verifies if a string satisfies specified conditions and, in the positive case, performs defined transformations on it. A finite-state automaton (FSA) is a finite-state transducer that verifies if a string satisfies specified conditions and, in the positive case, returns its unchanged copy.

Finite-state automata and transducers are commonly used in a wide range of tasks related to various subdomains of Natural Language Processing: dictionary representation, text normalization, tokenization, morphological analysis, speech recognition, speech synthesis or optical character recognition. FSTs are regarded as very efficient for string transformation. This is very important from our point of view as we intend to normalize large volumes (counted in gigabytes) of texts.

A very efficient C++ toolkit for building and manipulating FSTs is called *OpenFst* (Allauzen et all, 2007). Its usage requires basic programming skills. Our goal is to propose a formalism for diachronic normalization rules that takes advantage of FSTs but does not require a programming background. We have decided to build the rules on an intuitive subset of functions that are offered by *Thrax* (Sproat et all, 2007). *Thrax* is a library of tools for compiling grammars into finite state transducers, which was developed by researchers from Google Research and New York University. It is an open source project distributed under the Apache license.

Thrax-compliant grammars are expressed as lists of symbols, regular expressions or context-dependent rewrite rules. The *Thrax* compiler translates grammar specification into weighted finite-state transducers in the *OpenFst* format and, making use of the *OpenFst* functionality, allows for performing various operations on those transducers. *OpenFst* supplies a wide range of operations on weighted finite-state transducers, such as union, intersection, composition, projection, path search, etc.

The *Thrax*-compliant sublanguage, reduced for the needs of diachronic normalization, is defined as follows:

A) Regular expressions are allowed on character strings and FST declarations (point C), e.g.
"ab" denotes a string *ab,*
("a" | "b") denotes either *a* or *b*,
a* denotes an empty string or one or more consecutive *a*s (aa...)
a+ denotes a string of one or more *a*s.

B) Replacement rules are defined on strings, e.g.
("ab" : "cd") denotes the replacement of the string *ab* by the string *cd*
C) FSAs nad FSTs are declared by means of the "=" character, e.g.
Letter = ("a" | "b", ...),
NonEmptyString = Letter+
D) Subtraction of automata are marked by the "-" character, e.g.
Vowel – "i" is the automaton that verifies the appearance of a vowel other than "i"
E) Two special labels are introduced:
"[BOS]" and **"[EOS]"** stand for the beginning and the end of the string respectively
F) Rewrite Rule:
CDRewrite[Rule, LeftContext, RightContext, Alphabet] is a directive to apply *Rule* repeatedly provided that the constraints on the left context (*LeftContext*) and the right context (*RightContext*) are satisfied. The directive leaves all other characters defined in *Alphabet* unchanged.

4. Rules for Diachronic Normalization

4.1. Automata declarations

Automata declarations serve for defining character classes. The largest character class, *Any*, consists of all characters (weather alphanumerical or not) that may appear in a UTF-coded Polish text. Less numerous classes correspond to the set of Polish alphabetical characters (*Letter*), vowels (*Vowel*), consonants (*Consonant*), any non-empty string of alphabetical characters (*NonEmptyString*), etc.

Automata declarations may also serve for distinguishing specific classes of Polish letters. Listing 3. shows an example:

```
Followed_by_j = "c" | "s" | "z" | ;

Followed_by_i = "d" | "f" | "g" | "h" | "k" | "l" | "m" | "n" | "p" | "r" |
          "t" | "w" ;
```

Listing 3. Automata declarations for specific classes of Polish consonants

The classes defined in Listing 1. are applied in the rule that normalizes the archaic spellings *kolacyja* and *tragedyja* into *kolacja* (Eng. *supper*) and *tragedia* (Eng. *tragedy*) respectively. If the stem ends with a letter belonging to the first class ("c", "s" or "z"), the modernized ending starts with "j", if, however, the stem ends with *"d"* etc., the ending starts with the *"i"*.

Listing 4. shows automata declarations that serve to mark the beginning / end of a word:

```
NonLetter = Any - Letter ;

Beginning  = "[BOS]" | NonLetter ;

End = "[EOS]" | NonLetter ;
```

Listing 2. Automata declarations for marking the beginning / end of a word

NonLetter is defined to match all non-alphabetical characters. The *Beginning* automaton marks the start of an alphabetic string, the *End* automaton marks the end of an alphabetic string.

4.2. Replacement rules

Replacement rules look for the occurrence of specific character strings in a defined neighborhood and replace them with their modernized versions.

Each rule consists of two parts: The first part is a definition of a transducer for string replacement, with optional constraints on neighboring characters. The second part is a directive to apply the replacement globally in a text with optional additional constraints on the context.

CONTEXT-FREE REPLACEMENT RULES

In context-free replacement rules the directive part applies the replacement independently of the context.

```
Jota_before_vowel = ("y" : "j")
                    (Vowel - "i") ;

Rule001_07 = CDRewrite [ Jota_before_vowel,
                 AnyString,
                 AnyString,
                 Any* ] ;
```

Listing 3. A replacement rule independent of context

The first part of the rule in Listing 5. serves for the replacement of “y” by “j” if “y” is followed by a vowel other than “I”. The directive part puts no constraints on the context (*“y”* may be preceded and followed by *any string*). (The argument *Any** tells the rule to copy any other character.)

CONTEXT-DEPENDENT REPLACEMENT RULES

In context-dependent replacement rules the directive part applies the replacement only if additional constraints on the context are satisfied.

```
Infinitive_z = "y" ("dź" : "ć")          |
               ("ie" | "a") ("dź" : "ść")  |
               ("e"  | "ó") ("dz" : "c")   ;

Rule001_02 = CDRewrite [ Infinitive_z,
                 NonEmptyString,
                 End,
                 Any* ] ;
```

Listing 6. A context-dependent replacement rule

The rule in Listing 6. converts older forms of selected verb infinitive forms into their modernized forms. The first part (*Infinitive_z*) specifies the transducer: "dź" should be replaced by *"ć"* if it follows "y" (e.g. *bydź* -> *być*; English: *to be*); "dź" should be replaced by "ść" if it follows either "ie" or "a"; "dz" should be replaced by "c" if it follows "e" or "ó". The second part is a directive to apply the replacement, provided that the location of the replacement is preceded by a non-empty string and followed by the end of the word.

MULTIWORD RULES

Replacement rules may merge or divide words.

```
SeparateModal = "nie"
                ("" : " ")
                      ("można" | "trzeba" | "potrzeba" | "wolno"
                       "powinno") ;

Rule005_12 = CDRewrite[ SeparateModal,
                 Beginning,
                 End,
                 Any* ] ;
```

Listing 7. A word-division rule

The rule given in Listing 7. serves for the separation of the particle *nie* (Eng. *not*) from modal verbs such as *można* (Eng. *one may*), *trzeba*, *potrzeba* (Eng. *one should*), *wolno* (Eng. *it is allowed*), *powinno* (Eng.: *one should*).

```
Preposition_czem = ("zaczem" : "za czym")    |
                   ("poczem" : "po czym")    |
                   ("przyczem" : "przy czym") ;

Rule005_06 = CDRewrite[ Preposition _czem,
                 Beginning,
                 End,
                 Any* ] ;
```

Listing 8. A word-merging rule

The rule given in Listing 8. separates the archaic combined spelling of some prepositions: *za* (Eng. *behind*), *po* (Eng. *after*), *przy* (English: *at*) from the conjunction *czem (*locative, older form of the conjunction *what* — not only the words were merged, also the form of the pronoun needed to be modernized).

4.3. Exceptions to the rules

The major drawback of the adopted approach is that the rules undesirably convert existing contemporary words whose spelling comply with the rules. We call such words *exceptions*.

To obtain an exhaustive list of exceptions we have run the following procedure: All word-forms from the contemporary dictionary of the Polish language[7] are processed by the normalization rules. The word-forms that undergo modification by the rules are regarded as exceptions.

The initial set of merely 32 replacement rules generated no less than 61027 exceptions. A major part of the exceptions was generated due to too week constraints on the context. For example, the rule that modernized archaic imperative endings of some verbs ("-iej" -> "-ij") caused unwanted changes of correct adjective word-forms. This observation convinced us to revise the rules, with more attention paid to the constraints.

The exceptions (i.e. words that should not be transformed although they comply with the rules) include words:

- containing the characters "é" or "á", e.g. variétés, (these diacritic vowels were used in Polish before the reform in 19th century[8], nowadays they can be encountered only in some non-assimilated loanwords);
- containing the character *"y"* as a consonant (e.g. *cowboy*, *yacht*); the diachronic rules transform "y" into "j";
- containing the letter "j" following a consonant (e.g. *podjudzać*; English: to *instigate*); the rules incorrectly transform *j* into *i*;
- comparative forms of some adjectives; (e.g. *chudszy*; English: *thinner*); the rules treat such words as archaic forms of past participles and inserts the letter "ł" before the ending (the modification is desired for the archaic participle *poszedszy*,, which should be normalized into *poszedłszy*; English: *having gone*)
- common names, such as *Kołomyja* (the name of a Polish town), which, according to the rules, is unwantedly normalized to *Kołomja*.

[7] http://sjp.pl

[8] "Rozprawy i wnioski o ortografii polskiej przez deputacyą od Towarzystwa Warsz. Przyjaciół Nauk wyznaczoną", Warszawa, 1830 (English: Conclusions on the Polish orthography by the deputation founded by the Warsaw Society of Friends of Science, Warsaw, 1830)

To overcome the problem of unwanted transformations of existing words, we have compiled the set of exceptions into a finite-state automaton. The normalization procedure does not affect words accepted by the exceptions automaton.

On the other hand, there are also words that should be normalized and are not because their archaic versions are stored in the *sjp* dictionary (e.g. *decyzyj*, English: *decisions*). As a solution, we have decided to remove the archaic forms from the contemporary dictionary (i.e. the ones that are easy to detect). This leaves us with some archaic words, which are homophonic with modern wordforms (e.g. *kładź* – the archaic form of the verb *kłaść (*English: *to put*) and the Polish common noun *footbridge*). Currently, they are treated as exceptions in our approach.

5. Implementation

5.1. Testing procedure

As the rules for the diachronic normalization are not trivial and their interaction may lead to unexpected results, a proper testing procedure is crucial. We decided to embed test cases within the rules with a special mark-up („>” at the beginning of line) as given in Listing 9.

```
## Rule: Letter 'i' before vowel
> iayko -> jajko

Jota_at_beginning = (("i | y") : "j")
                    Vowel ;
```

Listing 9. A test case preceding a rule

Here, we specify that the expected output for *iayko* is *jajko* (English: *egg*). Test cases precede the related rule. The embedded test cases are intended for both humans and machines:

- for humans, they form a part of the documentation, as it is easier to understand the meaning of a rule with a clear example given;
- for machines, they are computer-readable test cases that can be verified automatically (akin to unit tests used in software engineering).

All such test cases are extracted automatically from the source codes of the rules and verified automatically in a continuous manner. If any of the test cases fails, the given rule (or its test case) *must* be modified before acceptance to the project.

Currently, the number of test cases is 61. “Soft” testing with a corpus of real texts (i.e. checking accuracy on a test text corpus) is planned for the near future.

5.2. Efficiency issues

FST composition combines two FSTs, A and B, into one FST. The composition applies transformations defined by A to an input string, and then applies transformations defined by B to the string returned by A.

FST subtraction applies transformation defined by A if the input is not accepted by B.

In theory, the diachronization procedure could be implemented as a single FST, constructed as a composition of all replacement rule FSTs, from which the exception transducer is subtracted. In practice, however, such a simple and elegant approach is not feasible. *Thrax* fails to generate the transducer representing the composition of all rules because of memory excess error.

To overcome this problem another processing scheme was implemented, namely on-the-fly composition. The input string is processed by the cascade of rule transducers r_1, r_2, ..., r_n connected in such a way that the output of r_i is passed to the input of r_{i+1} (precisely speaking, output of r_i is composed with r_{i+1}).

The equivalence of those two solutions follows from the associativity property of transducer composition operation:

$$input \,.\, (r_1 \,.\, r_2 \,.\, r_3 \,.\, \ldots \,.\, r_{n-1} \,.\, r_n) = (((((input \,.\, r_1) \,.\, r_2) \,.\, r_3) \,.\, \ldots) \,.\, r_{n-1}) \,.\, r_n$$

This solution makes it possible to avoid the construction of large transducers.

The problem of memory having been solved, the problem of time complexity appeared. Neither *Thrax* nor *OpenFst* package supply ready-to-use tools that allow for implementing on-the-fly composition with satisfactory processing speed. The problem was eventually solved by writing a special purpose low-level program using the *OpenFst* C++ programming interface. The implemented solution made it possible to avoid unnecessary repetition of several time-consuming operations, such as loading a transducer each time a rule is applied. Eventually, the processing speed increased to a reasonable level.

6. Evaluation

The evaluation experiment was carried out in 4 steps:

- Preprocessing
- Human verification
- Generation of a testing list
- Human cleaning
- Evaluation

In the pre-processing stage, a corpus of OCR-ed texts was processed by means of simple regex replacement rules (an example is given in Listing 10).

```
new Rule("([dfglmnprt])[jy]([aąeęiou])", "$1i$2"),
```

Listing 10. A Preprocessing rule that changes "j" or "y" into "l" in specific contexts

The aim was to eliminate frequently occurring errors of optical recognition, as well as to apply very simple and regular rules on character replacement in the diachronic normalization.

The result of pre-processing on the corpus was verified by humans, who diachronically normalized already pre-processed texts.

In the third step, the corrections made by humans were automatically compilated into a list of pairs: <original word, corrected word>. The list consisted of 3298 pairs.

Human cleaning consisted in selecting such items from the list, whose presence was due to the diachronic normalization process (and not OCR errors). The resulting list consisted in 759 pairs. These pairs were submitted to the evaluation process. The results are listed in Table 1.

Tool	Modified	Correct modifications
HIST-PL	247 / 759 (32.5%)	145 / 759 (19.1%)
AMEBA	144 / 759 (19.0%)	123 / 759 (16.2%)
DIACHRONIC	464 / 759 (61,1%)	324 / 759 (42.7%)

TABLE 1. Comparison of three diachronic normalization tools for Polish

Note that the low percentages (in general) in Table 1. may be justified by the fact that the data under examination consisted solely of the "hard words" that had not been changed by the pre-processor.

Our application (Diachronic) shows the highest recall on the one hand, but commits the highest number of errors on the other. This indicates that the Malinowski's work is a good starting point for rule creation, however the rules should be carefully re-examined to assure higher accuracy of the normalization tool.

7. Conclusions

This paper reports on a FST-based method for the diachronic normalization of Polish texts. The method gives satisfactory results in the aspects of efficiency

and quality. In the future, we intend to improve the results by defining new replacement rules on the one hand (by means of the analysis of normalized archaic texts) and putting stricter constraints on the existing rules on the other. We also plan to optimize the procedure that deals with exceptions. Finally, our goal is to define an automatic procedure, which, based on the rules used in the normalization process, predicts the origin date of the text.

References

Allauzen C., Riley M., Schalkwyk J., Skut W. and Mohri M., OpenFst: A General and Efficient Weighted Finite-State Transducer LIbrary, ***Proceedings of the Twelfth International Conference on Implementation and Application of Automata, (CIAA 2007), Lecture Notes in Computer Science***, Vol. 4783. pp. 11-23. Prague, Czech Republic. Springer.

Bronikowska R., Modrzejewski E. The enrichment of the lexical information and the corpus resources by using the results of the morphological analysis of historical texts, http://www.elexicography.eu/wp-content/uploads/2017/03/ (downloaded on 2017-06-15).

Graliński F., 2013, Polish digital libraries as a text corpus, in: Zygmunt Vetulani and Hans Uszkoreit (eds.), Proceedings of 6th Language & Technology Conference, pp. 509-513. Fundacja Uniwersytetu im. Adama Mickiewicza.

Klemensiewicz Z., 1963, (ed.), Pisownia polska. Przepisy – słowniczek, Warszawa – Kraków – Wrocław – Łódź, Zakład im. Ossolińskich.

Lisowski T., 2010, Economic calculation and Polish alphabetic writing, in: Sekiguchi T. (ed.), The International Academic Conference "Meetings of the Three Polish Studies Centres in Asia – China, Korea, Japan", pp. 195–204.

Malinowski M., 2012, Ortografia polska od II poł. XVIII wieku do współczesności. Kodyfikacja, reformy, recepcja; praca doktorska, Uniwersytet Śląski w Katowicach.

Mykowiecka A., Rychlik P., Waszczuk J., Building an Electronic Dictionary of Old Polish on the Base of the Paper Resource, in: Petya Osenova, Stelios Piperidis, Milena Slavcheva Cristina Vertan (eds.), Proceedings of the Workshop on Adaptation of Language Resources and Tools for Processing Cultural Heritage at LREC 2012, European Language Resources Association (ELRA), 2012, pp. 16-21.

Tai T., Sproat R., Skut W., 2011, Thrax: An Open Source Grammar Compiler Built on OpenFst, in: Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop, IEEE, Piscataway, NJ.