RIGA TECHNICAL UNIVERSITY
FACULTY OF COMPUTER SCIENCE, INFORMATION TECHNOLOGY AND ENERGY
Institute of Digital Humanities

ARTIS OZOLS

Converting Old Latvian (Mid-18th to the first half of the 20th Century) Texts to Modern Latvian Writing

Master Thesis

Scientific adviser:
Dr. philol., MSocSc. professor Marina Platonova

Consultant:
MSc, researcher Valdis Saulespurēns

Developed within the framework of the project "Narrative, Form and Voice: Embeddedness of literature in culture and society" nr. VPP-LETONIKA-2022/3-0003 of the State Research Program "Letonika – Fostering Latvian and European Society"

Riga 2024

Work Performance and Assessment Sheet

The Master Thesis has been elaborated at the Faculty of Computer Science, Information Technology and Energy within the framework of the Academic Master Study Programme "Digital Humanities".

Author of the Master Thesis
First name, surname _____ (signature, date)

Supervisor
Academic position, scientific degree, first name, surname _____ (signature, date)

The Master Thesis has been suggested for the Viva Voce Examination at the meeting of the Master Thesis Assessment Committee of the Academic Master Study Programme "Digital Humanities".

Head of the Academic Master Study Programme "Digital Humanities"
Professor Dr. philol. Marina Platonova _____ (signature, date)

Head of the Institute of Digital Humanities
Professor Dr. philol. Marina Platonova _____ (signature, date)

The Master Thesis has been publicly presented at the meeting of the Master Thesis Assessment Committee on _____ and evaluated with a mark _____ .

Secretary of the Master Thesis Assessment Committee
Associate Professor Dr. philol. Tatjana Smirnova _____ (signature, date)

Declaration of Academic Integrity

I declare that this work is my own and does not contain any unacknowledged work from any source.

Signature

Date

RIGA TECHNICAL UNIVERSITY
FACULTY OF COMPUTER SCIENCE, INFORMATION TECHNOLOGY AND ENERGY
Institute of Digital Humanities

CONVERTING OLD LATVIAN (MID-18TH TO THE FIRST HALF OF THE 20TH CENTURY) TEXTS TO MODERN LATVIAN WRITING

Artis Ozols

Abstract

The goal of this thesis is to create a tool that could convert a text written in orthography of the mid-18th, 19th and first half of the 20th centuries to modern Latvian orthography. For that, four tasks were set:

- Study the history of Latvian orthography;
- Designed a conversion tool;
- Prepare a dataset for testing the tool;
- Test and compare the tool with other conversion tools.

After studying the history of Latvian orthography, conversion rules were derived from the sources and implemented in the conversion tool which is publicly available as a website. For testing the tool, text was extracted from 20 periodical pages and then cleaned from recognition errors. With the test set, the tool was tested. From all 18677 words and 98713 characters in the test set, the conversion tool was able to convert 87.01% words and 96.75% letters correctly, while the best of three chatbots to which the conversion tool was compared with was able to convert 65.84% words and 78.91% letters correctly.

The work contains 3 sections, 83 p., 25 537 words and 41 references.

RĪGAS TEHNISKĀ UNIVERSITĀTE
DATORZINĀTNES, INFORMĀCIJAS TEHNOLOĢIJAS UN ENERĢĒTIKAS FAKULTĀTE
Digitālo humanitāro zinātņu institūts

TEKSTU PĀRVEIDE NO VECĀS (18. GADSIMTA VIDUS LĪDZ 20. GADSIMTA PIRMĀ PUSE) UZ MŪSDIENU RAKSTĪBU

Artis Ozols

Anotācija

Šī maģistra darba mērķis ir izveidot rīku, kas spēj pārveidot tekstu, kas rakstīts 18. gadsimta vidus, 19. un 20. gadsimta pirmās puses ortogrāfijā, uz mūsdienu latviešu valodas ortogrāfiju. Šim nolūkam tika izvirzīti četri uzdevumi:

- Izpētīt latviešu valodas ortogrāfijas vēsturi;
- Izstrādāt pārveides rīku;
- Sagatavot datukopu rīka testēšanai;
- Testēt un salīdzināt rīku ar citiem pārveides rīkiem.

Pēc latviešu valodas ortogrāfijas vēstures izpētes no avotiem tika izgūti pārveides noteikumi un iestrādāti pārveides rīkā, kas ir veidots kā publiski pieejama tīmekļa vietne. Rīka testēšanai tika iegūts teksts no 20 periodisko izdevumu lapām un pēc tam tika attīrīts no atpazīšanas kļūdām. Ar testa kopu tika veikta rīka testēšana. No visiem testa kopas 18677 vārdiem un 98713 rakstzīmēm pārveides rīks spēja pareizi pārveidot 87,01% vārdu un 96,75% burtu, savukārt labākais no trim sarunbotiem, ar kuru rīks tika salīdzināts, spēja pareizi pārveidot 65,84% vārdu un 78,91% burtu.

Darbs satur 3 nodaļas, 83 lapu., 25 681 vārdus un 41 atsauci.

Table of Contents

INTRODUCTION

During most of the existence of Latvian writing, printed texts were written with Fraktur typeface in the old orthography. Some of these texts form the foundation of the Latvian state and its culture. Unfortunately, this cultural heritage for most of the contemporary readers are incomprehensible without a text conversion, therefor it is crucial to make a tool that could convert such texts automatically to modern Latvian. A tool like this could also serve as a framework for similar languages, such as Lithuanian.

Just as Lithuanians, in the beginnings of Latvian writing, there was not a stable, defined way how words should be written, thus everyone who knew how to write, wrote in their own way. Such writing varied greatly from author to author and even one authors writing style changed over time. Later, after the second translation of the bible into Latvian in 1739 the orthography started to become consistent, and till the early 20th century it had relatively few changes.

Since modern readers are able to read texts that were written after, the goal of this thesis is to create a tool that could convert texts written in this old orthography to modern Latvian orthography. In this context, an old is considered a printed text issued starting from the mid-18th until the first half of the 20th century. To achieve this goal, four tasks are set:

- Study the history of Latvian orthography;
- Designed a conversion tool;
- Prepare a dataset for testing the tool;
- Test and compare the tool with other conversion tools.

To accomplish that, the timeframe set is two years. For creation of the tool the general research methods are used, i.e., the rules implemented in the conversion tool is based on the information derived from the literature. In this thesis, there are 7 main groups of literature used. Each group and the number of its sources are following:

- 4 – fundamental scientific research and monographs;
- 4 – articles from scientific journals and conference proceedings;
- 3 – publications on results of basic and applied research;
- 6 – general and special literature, periodicals;
- 3 – laws, regulations of the Cabinet of Ministers, EC regulations and other regulatory documents, standards;
- 11 – different internet resources;
- 10 – text corpora, data bases (dictionaries, glossaries, encyclopedias).

The aim of this tool is to facilitate reading in the old orthography, not to convert the text perfectly accurate. This is especially true for texts written in the 18th century, where not only lexicon was quite different from the one we use today but also word endings and word grammatical gender. To leave as much as possible original style, this tool does not change outdated words to those used today. It also does not take context into consideration.

In order for the tool to be easily accessible, it is designed as a website which allows users to either upload an image of a text which then will be recognized or write the text by themselves. After the input is given, the text is converted and returned.

To measure the accuracy of the tool and compare it with other conversion tools, a test corpus of 19th and 20th century texts in old orthography is collected. Such corpus of historical texts in digital form has not yet been created and therefor it involves few very time-consuming processes – text recognition from images, text revision and OCR error correction. Because of that, it was necessary to extend the time frame to two years. This corpus has also never been converted to modern Latvian, thus it also involved another time-consuming activity – text conversion to modern Latvian.

The test corpus ir created to test the tool and to measure the accuracy of converted texts. Since there are no publicly available tools that could convert old Latvian text, the conversion tool is compared with three AI chatbots – ChatGPT, Google Gemini and Microsoft Copilot. After the test set is converted by four tools, the converted texts are then compared to the correct, human converted text. This allows to measure how much words and letters each tool converted correctly.

1. THEORETICAL PART

This section provides an overview on the main concepts used in the thesis, how the Latvian writing has transformed from ancient to modern times – what were the main change and what was their influence; and finally, existing solutions for old Latvian conversion are explored.

1.1. Foundation building

As time passes more and more Latvians are not able to read old Latvian orthography. One of the main reasons why old Latvian orthography is so different from the one we use today is the difference in alphabet. Although some have seen long “o” or soft “r” and have adapted to “w” in places where “v” is written, the main obstacles most modern readers encounter while reading old Latvian are:

- Vowel length denotation with “h”;
- Double “e” for diphthong “ie”;
- “c” written as letter “z”;
- And denotation of letters “z”, “s”, “š”, “ž”, “č”.

Another reason is the use of Fraktur typeface. For modern reader, it makes familiar letters look strange. As mentioned before, during most of the existence of Latvian writing, printed texts were written in Fraktur script jet only few young Latvians know what Fraktur is. Moreover, in many sources it can be seen that Fraktur is often mistaken and used as synonym for Gothic letters. To ensure understanding, next paragraph will describe the appearance of the Fraktur and the Gothic style and the difference between them.

According to calligrapher Jake Rainis in his article “The History of Blackletter Calligraphy” (cf. Rainis, n.d.) he says that both Fraktur and Gothic are two separate calligraphic styles which constitutes Blackletter group; while Blackletter, just as Serif or Sanserif is a calligraphic style type. In the article J. Rainis describes the history of Blackletter script. He names four main styles of Blackletter: Textura, Rotunda, Bastarda, and Fraktur. As the main reason why Blackletter emerged, the author mentions the significant increase in literacy rates in Europe during the 11th and 12th century which resulted in a notable demand for books; but since at that time printing was not jet invented, texts could only be multiplied by rewriting. As a result of that, there was a necessity to substitute the old script with such that could save time when rewriting books and save space on the writing material (which at that time was also quite

expensive). From there “Textura” emerged – a style considered to be the foundation of Blackletter and which is most associated with the “Gothic” letters. cf.

Around the time when the first larger written records were written in Latvian, at the turn of the 15th and 16th centuries Fraktur style was invented. J. Rainis (Rainis, n.d.) notes that with the invention of the printing press the advantages of Textura style was no longer necessary instead the need for new, more legible style increased which led to the emergence of the Fraktur typeface. While this style quickly became popular among Protestants, Germans and areas under German influence (including the territory of Latvia); the Catholics and rest of the Europe continued to print using various types of Antiqua. cf.

Around the same time when Fraktur emerged, German pastors began writing texts in Latvian. Those texts were one of the earliest written records in Latvian. According to the Cambridge Advanced Learner's Dictionary & Thesaurus (Cambridge Dictionary, n.d.) “orthography is the accepted way of spelling and writing words”. At the early periods there were no accepted norms of spelling and writing words in Latvian, therefor they wrote as they thought was correct. Their writing was, however, based on the German orthography but it varied greatly from author to author.

Eventually, however, the norms in Latvian writing were formed and the formation was deliberately done by various groups. For translation of the Bible in Latvian, German pastors and linguists were the first who held the meetings regarding Latvian orthography. After the Bible was released the accepted norm was mostly based on the writing and spelling after which the Bible was translated. Nevertheless, there were many authors who still spelled words in their own way. After 130 years later, with foundation of the Latvian Literary Society, new meetings were held. Later, Riga Latvian Society established Knowledge Commission which tuck over the influence on orthography, then Orthography Commission, followed by Spelling commission of the Ministry of Education. These organizations were the ones that introduced changes and determined the accepted way of spelling and writing words.

1.2. Changes in writing

As archaeologist and historian A. Vilcāne (Vilcāne, 2018, p. 90) states, “By the 16th century, consolidation of the ethnic groups living in Latvia had reached such a stage that it allows us to refer to the existence of the Latvian nation”. Together with nationality, language also consolidated. In early 16th century appears first written texts in Latvian, therefore this

section will cover main changes in writing from the 16th century until 1957, which marks the last major change in orthography implemented in the conversion tool.

Philologist P. Vanags has studied the history of Latvian language. In his paper "Latviešu valoda pirms Latvijas valsts" (Vanags, Latviešu valoda pirms Latvijas valsts, 2019), he divides the history of Latvian language into seven stages. While the main subject in the article is Latvian language, the first sections greatly focus on orthography as well; and therefor the names and the periods chosen for this section are based on this article (except for the last one since it will cover multiple stages from the article). So the changes in Latvian orthography in this section is divided in five periods:

1. Period of graphization and choice of language;
2. Period of initial codification of norm periods;
3. Period of norm stabilization;
4. Period of partial norm changes and modernization;
5. Period of current orthography.

1.2.1. Period of graphization and choice of language

In the article (Vanags, Latviešu valoda pirms Latvijas valsts, 2019), Vanags names this period as graphization and the choice of language since this were the time when first written records in Latvian language appeared. These records come from German pastors who transcribed their religious texts in Latvian using German alphabet. Although few written words and phrases occurred earlier, the first longer written text that remains today is Guzbert's 1507 Lord's Prayer. cf.

At that time there were no writing rules of Latvian, so pastors were the ones who defined orthography. They also chose the Middle dialect between other two (the Livonian and the High Latvian dialects) as the main written Latvian dialect. Choosing one dialect eventually unified Latvian language and Latvians, and it also made it easier for Germans to communicate with anyone throughout the territory of Latvia. For printed texts they used a calligraphic style Fraktur. This style in the Middle Ages, especially in Germany and Sweden was very common so it influenced writers in Latvia respectively. Until the late 19th and the first half of the 20th century most of the printed texts (with exceptions) were written in Fraktur while manuscripts were written in Antiqua.

While studying Augstkaln's and Blese's analysis on Latvian 16th century printed texts, A. Ozols in his book "Veclatviešu rakstu valoda" (Ozols, 1965, pp. 75-76, author's translation)

defined the most frequent rules of writing of this time period: “The first written language is based on Low German techniques: the length of vowels is not marked (some instructions, of course, can be felt, but there is no system, so that exceptions would be mistake), diphthongs *ie* is not distinguished from *e* (short, long, narrow, wide), softened consonants are not marked (rarely *i*, *y* and *g* are added to them), voiced and voiceless hissed and hushed sibilants are not distinguished, there is no order in the notation of affricates; in consonant notation utterly - letter jamming (doubles; combinations of different letters for the notation of one sound - especially for hissed sibilants and affricates; *h* used just for image enhancement) and the phonetic principle . .”, “… instead of the short vowels in word endings letter *e* is used, instead of *-u* also used *-o* ... sometimes the vowel is completely lost, sometimes the vowel is used, i.e., letter *e*; *ou* used instead of *au*; occurs *u* and *ue* instead of *i*; *sk* instead of *šķ*.”.

Although he and E. Blese in more depth have found regularities in writing, there are still far too many inconsistencies. For instance, in his book Ozols (cf. Ozols, 1965, p. 77) lists nine variations on how E. Volter in 1858 catechism writes letter *s*. He also rightfully quoted Augstkalns conclusion of earliest writings in Latvian: “In the first written language, one often had to say: absurd” (Ozols, 1965, p. 79, author’s translation). Due to the inconsistence in writing and particularly peculiar syntax (for example: incorrect word order and direct text transfer from German) and morphology (wrong word endings and suffixes) this written language differs greatly from the language that is used today.

One of the best in terms of orthography and the most influential writers from this period is Andreas Gecelius. He wrote a manuscript with translated psalms of David and Solomon's proverbs in 1628. According to Ozols (Ozols, 1965), Gecelius manuscript contains long vowels, has better spelling and grammar rules are used more often; but the most noteworthy is his contribution in lexicon and syntax. For long vowels Gecelius used long stroke above the vowel (however they are not used consistently). Although Gecelius was not the first one who used such length notation, he was the one who inspired Christopher Fürecker to denote long vowels in his grammar manuscript which was later used as the basis for translation of the Bible into Latvian, which then became the norm in orthography. cf.

Although Gecelius writing stood out significantly compared to others of that time, his writing is incomparable to the writers of the next period. This is also confirmed by the linguist J. Zevers who Ozols (Ozols, 1965, p. 118, author’s translation) quoted saying, “… it can be seen that Gecelius’ translation of psalms of David and Solomon's proverbs in 1628 completely coincides with the 16th century monuments in terms of language, but Mancelius’ Velemecum

of 1631 shows a completely different language, so that it can be said that a new stage in the history of the Latvian language begins."

1.2.2. Period of initial codification of norm

Vanags names this period as initial codification of norm since in this period the first norms in writing appeared. He mentions (cf. Vanags, Latviešu valoda pirms Latvijas valsts, 2019) that writing in previous period could not satisfy neither readers nor those to whom the text was read so it was clear that norms in writing had to be set. In this period written language become even more consistent.

This period begins with one of the most influential books of the 17th century – Georg Mancelius' "Lettisch Vade mecum" – a handbook of the Lutheran church. Seven years later in 1638 he published another highly influential book – the first dictionary that features Latvian – "Lettus". This book has 6000 translated words from German to Latvian with multiple variants. Augstkalns (cf. Augstkalns, 1930, pp. 102-103) in his analytical article defined the most frequent writing features of Mancelius' books – he:

- differentiates diphthong "ie" form "e" using "ee" for diphthong;
- differentiates wide from narrow "e" using "ä" for wide and "e" for narrow;
- differentiates vowels in word endings (except: for adverbs he uses "-e" instead of "-i"; for genitive plural – "-o" instead of "-u" and for compound in genitive – out of place "-a");
- differentiates all consonants (with some inconsistencies);
- for long vowels (including long "o") use "h" after vowel (with exceptions).

One source (cf. valoda.ailab.lv, n.d.) mentions that Mancilius used "oh" only in word stem, while in word endings he used "ó" or "ô" and in suffix – "o". Bergmane and Blinkena in "Latviešu rakstu attīstība" (cf. Bergmane & Blinkena, 1986, pp. 71, 72) writes that Mancilius and other authors in prefixes, prepositions and first half of compound words sound [z] denotes with letter "s". Ozols adds that Mancilius used "ſ" (long s) for "z", "ſ" (long s with diagonal stroke) for "s" and so called "bunch of letters" – "ſch" for letter "š", "ſch" for "ž" (cf. Ozols, 1965, pp. 168, 169). Later, this notation was used for centuries.

Almost all of these principles listed constitutes the rules of the orthography used till the first half of the 20th century. From the listed principles, the only difference is that:

- wide from narrow "e" in later centuries was not distinguished;

- All vowels in word endings later were correctly differentiated;

As Augstkalns describes it, “1631 is an epoch. In this year, the paths of the Latvian written language turn, ‘from this year the current Latvian written language is counted’” (Augstkalns, 1930, p. 101, author’s translation).

Overall, these and other Mancelius books had remarkably improved consistence in writing, however he still sometimes had trouble to adhere to a certain law of writing. For example, Ozols (cf. Ozols, 1965, pp. 166, 168, 169) mentions that Mancelius:

- used “ie” or sometimes “y” for long “i”;
- although “y” was also used for letter “j” in verbs between vowels (e.g., “domayis”);
- He also had trouble in “s” and “z” as well as “ķ” and “ģ” distinction;
- In some cases, he used “ß” for “s”, while for “ķ” he often used either “gk” or “ģk”.

As with many influential writers, Mancelius had followers from which the most well-known are Christoph Fürecker and Henricus Adolphus. After Fürecker’s manuscripts Adolphus in 1685 published Latvian grammar book “Erster Versuch einer kurzverfasseten Anleitung zur lettischen Sprache” – the most important grammar book of the 17th century which later was used for translation of the Bible.

If before, the writers were the ones who influenced orthography the most, but after the publication of this book, writing was influenced by groups of writers and linguists. This book is the most influential book of the 17th century – the Bible which was first translation into the Latvian language. In a short article the history of the translation was described (Lingvistiskā karte, n.d.). In 1662 general superintendent of Vidzeme, Johann Gezelius expressed the necessity for translation of Bible in Latvian. He received financial promise from the king of Sweden. As a translator Ernst Glück was chosen. He, however, did not translate the Bible alone; he was assisted by a team of his own choice and parts of Bible were translated before already by other pastors. The writing was determined by several conferences with Courland’s un Vidzeme’s clergies held in Jelgava. As a basis for writing Fürecker-Adolphi’s grammar and Mancilius’ dictionary was used. After almost nine years in 1690 Glück finished translating Bible and in 1694 the king of Sweden issued an order for the distribution of the Bible. cf.

With the publication of the translated Bible, writing stabilized and became more uniformed. But just as other 17th century texts, it was still full of Germanism. Ozols (cf. Ozols, 1965, pp. 277-288) listed many characteristics of first edition of translated Bible; here are some:

- Pronoun “tas” was used as article;
- Words, especially adjectives were translated directly;
- Word genders were assigned incorrectly;
- In interrogative sentence most often “woj”, less often “wuj”, “waj”, in rare cases “ar” was used. (e.g., “Ar tu pahr mums tahdu Warru turrefi?”);
- Numerals were written in German style – from 21 to 100, tens were written after ones (e.g., “Ewangeljums trefchâ Swehdeenâ ohtrâ defmitî”);
- Although in 17th century this form was rare, in the Bible plural instrumental case can be seen in these examples: “ilgis”, “fahnis”, “pirmis”, “labbis”;
- Instead of using preposition, noun “kahrta” with ending -an, also -am were used (e.g., “Pirman kahrtan”). These endings were also used in several prepositions and adverbs (e.g., “papreekfcham”, “eekfchan”, “augfcham”);
- Plural nominative pronominal noun had ending -ie, while plural genitive – -uo;
- For nouns of 6th declination in dative ending -i was used instead of -ij (e.g., “tai fwehtai Telti”). This ending can also be seen in other feminine substantives;
- Often monosyllable present 2nd person verbs had ending -i (e.g., “ehdi”, “leezi”) and in 1st person – -mu (e. g., “ehmu”, “eemu”). Monophonic verbs in present 3rd person also had endings which now is not used (e.g., “rauda”, “pahrwarra”);
- “Lai” in imperative expressions in the 3rd person was spelled with “d” at the end (e.g., “Laid tohp”);
- At the end of the question word or if it is missing, at the next word, ending -gi or -g was used (e.g.: “Tuggi essi tasz Messias?”.

All these features listed show how different the orthography was at that time compared to the writing which was used in later centuries. For converting such texts, there would be needed completely different rules and approach in contrast to text conversion from later centuries. This is especially true for word stemming since word endings in this period as listed above can be way different than the ones used today and in later centuries. Of course, the conversion tool described in this thesis will be able to convert words in such texts as well, but the conversion accuracy will be quite low.

Despite the enormous effort put into improving the orthography and translating the text, the book was not highly demanded. This is evidenced by the number of copies sold – till the second edition from 1500 copies only 1400 bibles were sold.

1.2.3. Period of norm stabilization

This period Vanags names as the period of norm stabilization since after volume two of the Bible for the next 100 years Latvian grammar and writing had almost no changes. Glück found flaws in newly translated Bible, so he started to edit the text. Edgar Dunsdorf in his book "Pirmās latviešu bībeles vēsture" (Dunsdorf, 1979, p. 132) writes that During the Great Northern War he served as pope for three battalions. But despite the difficult war conditions Glück still had time and energy to work on the second edition of the Bible. Unfortunately, he did not manage to finish his work and even worse, his manuscripts disappeared. Only 45 years after the first edition, the sun of Vidzeme's general superintendent, J. B. Fisher together with pastors invited by the duke of Courland rewrote and release the second edition of the Bible in 1739 cf.

According to Ozols (Ozols, 1965, pp. 303-311) two main differences between the first and the second edition can be seen in lexicon and etymology. To better understand the orthography of that time, the next paragraph summarizes the main changes Ozols found in writing of the Bible's second edition:

- Nouns were not written with capital letters;
- Nouns were written with declinations that are used today (e.g., "naglas" instead of "nagles", "upuris" instead of "upurs");
- Noun endings was changed (e.g., "waddons" instead of "wadds");
- Numbers were spelled closer to how they are spelled today (e.g., "Ewangeliums peektâ padeſmitâ ſwehdeenâ" instead of "Ewangeljums peektâ Swehdeenâ pirmâ deſmitî");
- Pronouns "patti", "muhſu" and "juhſu" was correctly inflected (e.g., "Ka ſche jamirſt mums un muhſu (instead of "muhſeem") lohpeem");
- Pronouns "ikkuŗſch" was changed to "wiſſi", while "wiņņs" or "wiņņa" to "ſaws" (e.g., "Kam irr wiņſch dewis ſawus (instead of "wiņa") darbus izteikt?");
- Verb prefixes and suffixes were changed (e.g., "pagohdinajis" instead of "apgohdinajis"; "walkaht", "paglabba" instead of "walkoht", "paglabbo");
- Prepositions "pahr", "uhs" and "kahrt" was changed to "par", "us", "ar" (e.g., "Un ka tee pagani Deewu par (instead of "pahr") to ſchehlaſtibu teiktu"). cf.

As it can be seen, the orthography improved significantly. But even more than that improved lexicon. Some words were replaced with others to clarify the meaning. The writing had improved to such an extent that for the next hundred years, there were no demand to reform it.

1.2.4. Period of partial norm changes and modernization

This period starts with the first Latvian National Awakening. In this period German pastor were not the main influence in Latvian culture. Despite their efforts to continue writing in Mancelius' style, New Latvians not only changed Latvian writing but also wrote the first Latvian novel, first epos and started several Latvian magazines.

Ina Druviete in her book "Kārlis Mīlenbahs" (Druviete, 1990, pp. 178-179) mentions German pastors as those who first initiated discussion about the need for changes in orthography in the 19th century. In 1824 they founded the Latvian Literary Society. In this society they discussed on how they could improve Latvian orthography. They were also planning to create a committee that could realize that. The committee was not established but the initiative did not disappear. cf.

The Latvian Literary Society occasionally published collections of articles called "Magazin". In these collections they discussed on what Latvian orthography should be like. Newspaper "Latvijas Vēstnesis" (cf. Latīņu un slāvu alfabēts latviešu rakstībā, 1997) in 1997 mentions that in volume 2 of the "Magazin" R. Šulcs wrote that it would be a great idea to use Russian alphabet for Latvian texts, especially for such letters as "š", "ž" and "č" because that would facilitate learning to read in Latvian society; D. Vents supported it and wrote that for all sound in Latvian language there are corresponding letter in Russian alphabet, while K. Ulmanis in the same volume opposed it saying that no new letters are needed, except for long vowels, which could be denoted by a "roof" over the vowel; by doing so no needless second consonant after short vowel would be necessary. As it was common in that time, especially after open syllable, consonant that were after a short vowel was doubled.

As it can be seen, contrary to the Bible translation meetings, the Latvian Literary Society did not have a unanimous opinion on how orthography should be changed, even in such matters as letters form. In 1847 "Magazin" the first Latvian nationality linguist J. Bārs published a research on phonetics in Latvian language and how they can be denoted in more simplified way. Bārs proposed several changes in orthography from which some were later adapted in the orthography reform of 1908. According to I. Jansone's article in journal "Akadēmiskā dzīve" (Jansone, 2008, p. 63) he suggested:

- To use Latin letters (Antiqua);
- For "ʒ", "ſ", "ſ" to use "c", "s", "z" but kept "w" as "v";
- For "š", "ž", "č" to use acute sign above the vowel – "ś", "ź", "ć";
- For soft consonants to leave them crossed as before;

- Diphthongs “ie” and “o” write as “ia” and “ua”;
- Not to distinguish the narrow from the wide “e” and “ē”;
- To use two types of length marks.

Literatura.lv reveals more with quote from S. Kļaviņa (literatura.lv, 2023, author’s translation): “… The vowel length is marked with horizontal line above the letter if vowel is pronounced stretched (..) or with grave accent if the vowel is pronounced broken/pushed …”.

For people who lived at that time Antiqua was nothing new – it was used in certain parts of the Bible, in first multilingual Latvian dictionary, but most of all in manuscripts. Long vowel notation with macron (◌̄) was also nothing new – it was first used in 1561 A. Will’s Old Prussian catechism.

The first to support Bārs’ orthography was J. Alunāns. In 1856 in the last page before the table of contents of his most memorable work “Dziesmiņas, latviešu valodai pārtulkotas” he used Antiqua, “s”, “z”, “c” instead of “ſ”, “ſ”, “ʒ” and “š”, “ž”, “č” instead of “ſch”, “ſch” and “tſch”. However, proposal by Bārs’, just like proposals from others, were not widely implemented and remained only at the level of ideas. However, these ideas sparked discussion and influenced others to make new proposals.

In 1860s orthography discussion was especially live. Many, mostly teachers, expressed their view on orthography in press and majority was in favor of any orthography reform. Reforms happened, however, it was enforced by Russian empire. In 1864 in Vilnius Latvian alphabet with Russian letters was published and already in next year a law that forbids to print and distribute texts with Latin letters was passed. However, within the territory of Latvia this law applied only on Latgale and lasted until 1904.

marks the influence of the Slavic alphabet in the second half of the 19th century.

At first even new Latvians spoke favorably of Slavic letters. According to “Latvijas Vēstnesis” (Latīņu un slāvu alfabēts latviešu rakstībā, 1997), after K. Valdemārs suggestion in 1866 another Latvian alphabet with Russian letters was published, however later, he, just like another new Latvian, F. Brivzemnieks, admitted that such writing is too complex, imprecise and it is not possible to ensure the exact phonetic notation of Latvian words using Cyrillic letters. Then, Brivzemnieks with the help of K. Barons created a new Latvian alphabet. This alphabet is notable for the fact that consisted of Antiqua letters with “š”, “ž”, “č” as it is used today. It also included horizontal line above vowels, however it was used to distinguish syllable intonations. cf.

In the same article “Latvijas Vēstnesis” (Latīņu un slāvu alfabēts latviešu rakstībā,

1997) outlines how over time consonant doubling after short vowel disappeared. It started in 1867 in Riga where Latin letters were allowed. H. Alunāns wrote a book without double consonants after short vowels. K. Biezbārdis in 1869 also wrote a book not only without such double consonants but also without distinguishing soft from hard consonants and long from short vowels. Later though, he admitted that for some consonants diacritical marks should be used. In 1872 influenced by these authors newspaper "Baltijas Vēstnesis" also discards double consonants after short vowels. After "Baltijas Vēstnesis" more and more newspapers discarded them until they disappeared completely. cf.

In 1868 Riga Latvian Society and half a year later in 1869 the first Latvian scientific organization, Knowledge Commission of the Riga Latvian Society was founded. It was the main initiator and implementation center of spelling reforms. And just like the Latvian Literary Society, they also had a collection of articles.

In volume 1 of Knowledge Commission's collection of articles (cf. Rīgas Latviešu biedrības Zinātnības komisija, 1876, pp. 45-46) new orthography principles were made; the proposal was to use the same principles that were suggested by Bārs, except:

- For "š", "ž", "č" and soft consonants to use comma-like diacritic sign below the consonant;
- For long vowels to use acute. Long vowels to be denoted only in word stems and in word endings if it changes the word meaning;
- "o" for long vowel "o", "ŏ" if the consonant is pronounced short and as before "ô" in word endings if word is in the locative form.

Later in 1885 in volume 3 (cf. Rīgas Latviešu biedrības Zinātnības komisija, 1885, pp. 48-56) a proposal was made to indicate vowel length only in the word endings, but neither this nor the previous different notations of vowel length could satisfy Latvian writers.

Druviete (Druviete, 1990, p. 181, author's translation) in her book "Kārlis Mīlenbahs" describes those times with a quote from "Baltijas Vēstnesis": "A real strike on our linguists and writers for our orthography issues - almost everyone likes to write in their own orthography. ..Our writers and linguists stick to their \, /, h and other crumbs with the same heart and tenacity as the ancient hero who, lying on a balk, sang: "Better take my head than my ... orthography"." She adds that from time to time a poem or an article appears in periodicals written in new orthography to accustom the public to the new writing.

1.2.5. Period of current orthography

In 1908 Riga Latvian Society establishes a separate commission for orthography – Orthography Commission. With its chairman K. Mīlenbahs, its task was to develop and implement a proposal for a new Latvian orthography. In order to find out the public opinion, a survey in "Dzimtenes Vēstnesis" was made. According to Druviete (cf. Druviete, 1990, p. 185), the survey results, were following:

- 21,01% of respondents answered that they would like to keep Fraktur;
- 6,56% – waned to switch to Antiqua immediately;
- While 42,43% waned a gradual switch to Antiqua.

In the same year after 18 sessions the commission published an article "Invitation in the case of Latvian spelling reform" in newspaper "Latvija" inviting everyone to use the new orthography – that is to use:

- Antiqua letters instead of Fraktur;
- Macron (the horizontal line above the vowels) for long vowels in both word stems and endings, except in word locative forms where circumflex should be used;
- "š", "ž", "č", "dž" instead of bunch of letters (i.e., "ſch", "ſch", "tſch", "dſch");
- "s", "z", "c", "v" instead of "ſ", "ſ", "ʒ", "w";
- "ie" instead of "ee". (cf. Druviete, 1990, pp. 85-86)

These were not mandatory rules, they were more like an invitation. Although, this writing had its opponents, eventually it became the main Latvian orthography. In 1920, the newspaper "Latvijas Vēstnesis" began publishing paper in the new orthography, from 1934 joined the newspaper "Rīts" and later others, until the last one who switched to the new orthography in 1938. In 1909 this orthography was taught in schools while after the World War one and after the independence of Latvia in 1919 it became the official orthography of Latvia.

On December 30, 1920 again, new changes were introduced (Rīkojums par ortogrāfijas reformu, 1921, p. 218, author's translation): "Ordinance on orthography reform.

1. No later than July 1, 1921, Latin letters must be introduced instead of Gothic letters in all government and communal institutions and schools.
2. Instead of Gothic letters w, ſ, ſ, ʒ, ſch, ſch, tſch, dſ, bſch, letters v, s, z, c, š, ž, č, dz, dž should be used.
3. Softened l, r, n, k, g should be written as ļ, ŗ, ņ, ķ, ģ.

4. In foreign words, letter h (e.g., Hamburg) should be distinguished from ch (e.g., Chaŗkova).
5. Compound vowel “ee” should be written as “ie” instead.
6. To mark the length of vowels in the roots, suffixes and word endings, a horizontal dash (–) should be used, also length should be marked for all long-pronounced vowels and not just for some.

Riga, December 30, 1920

Prime Minister Ulmanis.

Minister of Education J. Plāķis.”.

So, the difference between this and the previous order in orthography was that macron was also used for words in locative form (instead of circumflex) and the distinction between letter “h” and “ch”.

Not all authors choose to follow these recommendations – some kept traditional denotation of long vowels, some followed all the recommendations but kept “ee” for dipthong “ie”, but some still refused to switched to Antiqua. Despite resistance from Latvian society, government kept this orthography. The writing of authors, teachers, and their students gradually became more consistent as time passed and since changes that followed in next decades were relatively small, they did not cause major difficulties. The most notable from these changes were long “o”, soft “r” and “h” denotation.

According to A. Bergmane (Bergmane & Blinkena, 1986, p. 62), in 1908 reform diphthong “o” was also proposed to be written as “uo”, but it was rarely used in real life so in 1920 reform it was not introduced. Regarding long vowel “o” Bergmane writes that:

- In the end of the 18th and the first half of the 19th century in printed texts internationalisms and foreign proper nouns with long vowel “o” was denoted with “oh” but could be pronounced either as long “o” or as diphthong;
- In the mid-19th century till 1870s it was denoted with both “oh” and “o”, but later from the end of the 19th century till the first half of 1930s – only with “o”;
- From 1930s in spelling dictionaries and printed texts again long “o” appeared denoted as “ō”;
- In 1938 the Spelling commission of the Ministry of Education replaced the “ō” with “o”, but in 1939 allowed it only when translating classical literature.
- After 1940 the commission again introduced “ō” in the alphabet, but from later in 1946 the long vowel “o” together with the soft consonant “r” was removed

from Latvian alphabet. cf.

Vanags in Nacionālā Enciklopēdija (cf. Vanags, latviešu valoda, 2023) writes that in 1938 Spelling commission issued a law that forbids the use of all long vowels in foreign words and removes "ŗ" from the Latvian alphabet but after half a year these changes were annulled; then in 1946 these changes have come into effect again until 1957 (except soft "r") when also "ch" was replaced with just "h".

1.3. Previous studies

So far publicists or researchers have converted books and other texts on their own from old to modern Latvian. Unfortunately, for now there are no publicly available conversion tools that would be able to convert such texts automatically. There is however some research made regarding automated conversion.

In 2012 researchers from Institute of Mathematics and Computer Science of University of Latvia published an article (Pretkalniņa, Paikens, Grūzītis, Rituma, & Spektors, 2012) in which they describe the system they developed for converting words from old Latvian Fraktur to contemporary Latvian. They mention that the system is an ongoing work which was designed for the National Library of Latvia. Their aim was to add this system to the user interface of the digital library of historical periodicals, but the National Library of Latvia decided not to implement it due to poor conversion quality and the fact that the system cannot process multiple words at once. The system returns several variations of the word in modern Latvian. This system consists of following stages (Pretkalniņa, Paikens, Grūzītis, Rituma, & Spektors, 2012):

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

> variants (both wordforms and lemmas) can be ranked according to their frequency in a text corpus."

Their goal was to create a set of transliteration rules for texts from 1750 to 1880. Now this goal is being realized by Pretkalniņa in another project. In 2002 researchers from Faculty of Humanities of the University of Latvia, Latvian Language Institute and Institute of Mathematics and Computer Science of the University of Latvia collected and are still collecting earliest texts written in Latvian. They have created corpus "SENIE" which consists of scanned and digitally recognized texts from 16th, 17th and 18th century. These texts are manually corrected from errors. In 2003 "SENIE" website (http://senie.korpuss.lv) was created which enables users to search for a word or a phrase in all corpus texts. Since these texts are in old Latvian, currently this feature is possible only by entering search phrase in old Latvian. Since text authors form the 17th and especially the 16th century wrote very inconsistently and since word spelling from modern Latvian differs considerably, researchers decided to facilitate their search engine. They decided to convert the corpus as close as possible to modern writing while keeping the characteristics of the old texts. In 2022 they wrote an article in collection of articles "Aktuālas problēmas literatūras un kultūras pētniecībā" (cf. Andronova, et al., The conversion of early written Latvian texts into modern spelling: previous experience and automatization attempts, 2022) in which they described necessary steps to convert the corpus automatically:

1. Create conversion rules for a text or specific group of texts;
2. Use the conversion rules to automatically convert a text or part of it;
3. Search for errors in converted text and update conversion rules if necessary;
4. Repeat this process from step two until desired quality is reached.

Compared to previously mentioned system where Pretkalniņa contributed, for this project they decided to use only conversion tables because most of source texts in "SENIE" corpus are older than texts used in previously mentioned system and therefore with more inconsistent writing and larger morphological variety. Besides, lexicon in these texts are even more different thus their dictionary would also not be suitable for this project, therefore they decided to closely analyse texts instead.

There is, however, an old Latvian dictionary "Historical Dictionary of Latvian (16th-17th Centuries)" (www.tezaurs.lv/lvvv) created by the same authors (except E. Skrūzmane). This dictionary contains entities with a word or words in modern Latvian and their variations in old Latvian, examples in sentences and for some entities – grammatical and etymological description, illustrations, explanation and more. The source for this dictionary also comes from

“SENIE” and just like for “SENIE” conversion, this dictionary was also created by closely analyzing each word. So far on March 1, 2023, there are 1845 entities available on their website but as Andronova, et al. group mentioned in other publication (Andronova, et al., User-friendly Search Possibilities, 2022): “[it] is still too small (only ca 2000 entries) to be of significant help for large-scale transliteration”.

Later in article they reviled three main conversion rule groups and their subgroups (Andronova, et al., User-friendly Search Possibilities, 2022, p. 6):

1. “Unambiguous graphemic correspondences:
   1) grapheme-to-grapheme conversion, e. g., à>ā (Dahrſà>dārzā ‘in a garden’);
   2) grapheme combination to letter, e. g., tſch>č (Lahtſchus>lāčus ‘bears’ Pl.Acc.);
   3) letter to grapheme combination, e. g, x>ks (attmaxaht>atmaksāt ‘to repay’);
   4) grapheme combination to grapheme combination, e. g., ee > ie (peedärr>pieder ‘belongs’).
2. Positional (graphemic and morphemic) correspondences:
   1) depending on the position in a word, e. g., in the beginning or in the middle of the word: tz>c (Tzilwäki>cilvēki ‘men’), in the middle or at the end of the word: tz>c (tapetz>tāpēc ‘therefore’, Swetze>svece ‘candle’); at the end also tz>ts (ſälltz>zelts ‘gold’) or tz>ds (Ghalltz>galds ‘table’);
   2) depending on neighboring letters, e. g., aya>āja (ißghaya>izgāja ‘went out'); but ty>tī (nackty>naktī ‘at night’).
3. Individual (lexical) correspondences:
   1) word roots, e. g., ſwäht>svēt (ſwähtitam>svētītam ‘blessed’), here we also deal with position in the word, e. g., beginning of the word tytcz>tic, (tytczam>ticam ‘we believe’), but at the end of word tytcz>tīts (raxtytcz >rakstīts ‘written’);
   2) separate lexemes, e. g., ſöv>sev ‘for oneself’.”

Regarding the tool described in this thesis, the method of text conversion is similar – text is converted using conversion tables. Since texts from the second half of the 18th and especially 19th and early 20th centuries are written in orthography which is much more closer to modern Latvian, for creation of conversion tables mostly the group of unambiguous graphemic correspondences was used.

The researchers have concluded that earliest texts require more conversion rules than texts written later, but it also depends on text size – the smaller the text, the less rules are needed. Overall, conversion rules making is a time-consuming process – as they say, it is a small

scientific research. Answering the question in e-mail if they plan to create a conversion tool for converting old to modern Latvian, E. Andronova replied that they will, but at first the tool will be a pilot-convertor only for 18th century texts. She added that it will be a wonderful benefit if this convertor would work also for the 19th century texts but texts from the 16th and 17th centuries differ from author to author and even between one author over time, therefore it would be too hard to create a universal tool that could convert texts from these centuries. They plan to finish this tool by the mid-2024.

A large research of old Latvian writing was done by previously mentioned A. Bergmane and A. Blinkena. In 1986 they wrote a monograph "Latviešu rakstības attīstība" which describes the development of phoneme notation, norms of orthography and pronunciation, letter capitalization and interpunction in Latvian language over time. This book contains valuable information for rule making using historical notation of phonemes and these rules will be used to create the conversion tool. A compendium of all the laws can be found in section 2.2.

2. ANALYTICAL PART

In this part the conversion tool will discussed in details starting from its methodology, specification, limitations as well as the specific rules and libraries used in its development.

2.1. The conversion tool specification and limitations

In this master thesis the process of text conversion from old to new Latvian orthography is described. Texts in such way can be transformed in several ways. Researchers from previously mentioned project on corpus "SENIE" conversion from old to new Latvian in their publication (Andronova, et al., The conversion of early written Latvian texts into modern spelling: previous experience and automatization attempts, 2022, pp. 348-350) lists three main approaches of the text conversion:

- "Transliteration is the replacement of Gothic letters and diacritical marks used in ancient texts with Antiqua letters and diacritical marks (for example, ŧ → k, ʒ → z ...)";
- Transcription is the replacement of the graphemes used in ancient texts with graphemes used in modern Latvian language (for example, ah → ā, tsch → č, x → ks, w → v). When transcribing such texts, the doubling of consonants is disregarded, which usually indicates the shortness of the preceding vowel in the writing of ancient texts. The more commonly used morphological principle of modern writing is also widely applied in transcription, which replaces the more frequently used phonetic principle in ancient texts (for example, wezz → vecs, prahtz → prāts) ...";
- "Adaptation is the approximation of an ancient text to modern Latvian language."

Researchers for their own translator are using all three previously mentioned approaches – transliteration, transcription and adaptation. All three are used in the tool described in the thesis (which later in the text will be called the conversion tool) as well.

As mentioned before, they use conversion tables to convert the text. The same method is used in the conversion tool as well. But to avoid converting some words that should not be converted, it has also been supplemented with exception word processing. Unlike in their project, for finding exception words an online dictionary was used. As mentioned before, for their project lexicon was quite different from the one used in modern Latvian, while for mid-

18th till mid-20th century lexicon is already similar. Therefore, online explanatory and synonym dictionary Tēzaurs (Spektors, et al., 2023) was used. The authors of the dictionary provide an open data of the dictionary's content in CLARIN (Common Language Resources and Technology Infrastructure) repository. This dataset contains 397,000 entities with its explanation and for most entities, there is word morphological annotation given. This makes it is easier to group words which can be stemmed from words which should not. With this information it is possible to process exception words in any inflection which makes conversion process even easier.

Corpus "SENIE" researchers encountered a terminological issue – how to call this process. Currently, there is no designation introduced for naming this process, therefore in this thesis, just as in other papers from researchers of University of Latvia, this process of transforming texts form old to new Latvian will be called conversion.

The process of conversion can sometimes be an author's interpretation on how text should be transformed. There are things that author cannot know, like the way proper nouns should be converted to modern Latvian. For instance, "Johanna" can be converted into three different names that are registered in the Office of Citizenship and Migration Affairs of the Republic of Latvian (OCMA):

- into "Johana" by removing double consonant after short "a" (1 person registered (OCMA, 2023));
- into "Joanna" by converting "oh" into "o" (75 persons registered (OCMA, 2023));
- into "Joana" by applying both of these conversion rules (25 persons registered (OCMA, 2023));
- Or the name can be unchanged – "Johanna" (42 persons registered (OCMA, 2023)).

Decisions like these or whether to change word order in sentence or "ŗ" to "r" and "ō" to "o" largely depend on the intended audience. For instance, for corpus "SENIE" researchers strive to keep as much original text and style in conversion as possible since their corpus main audience are scholars. On the other hand, the main audience for the conversion tool are mostly laypeople/non-experts, therefore this tool strives to convert text as close as it can to modern Latvian.

To clarify what will and what will not be converted, next paragraph will describe conversion principles of the tool:

- Conversion will be performed in character and word level, therefore context will not be considered;
- Conversion will approximate the text as similar as possible to modern Latvian. This also means that some words that are written correctly may be converted to a different or even incorrect word, but overall such cases should be very rear, especially because exception wordlists are used.

Since the conversion tool do not take context into consideration word order in a sentence, lexicon, capitalization of words and punctuation use will not be edited, thus the text will also be converted quite close to original text. Just as with some proper nouns, other words may also be converted in multiple ways. Without knowing the context, the incorrect may be chosen, for example: word “mahzihtajs” can be converted:

- In “mācītais”, assuming that the word is an adjective and ending “ajs” should be converted to “ais”;
- Or in “mācītājs”, assuming that the word is a noun and that the ending should be changed to “ājs”.

The intended language for the conversion is mainly Latvian standard language or standard dialect. There are three main dialects in Latvia: the Middle, the Livonian and the High Latvian dialect. According to the article in valoda.ailab.lv (valoda.ailab.lv, n.d.), the Middle dialect became as the bases for standard dialect which over time enriched with words and expressions from other dialects and regions. They mentioned that the Middle dialect was chosen because at that time central Latvian cities were economically and culturally more developed than others. By their definition, “Standard language is multifunctional; it is cultivated and normalized as a standard language not only in spoken but also in written form; it serves the entire nation as a means of communication and cooperation in any sphere of life”. Although first books in Latvian were written in this dialect, the language in which they were written was not cultivated; this process deliberately began only in the second half of the 19th century. As long as a text is written in the Middle dialect, the conversion tool should work relatively fine also for texts written before, however texts in the Livonian and the High Latvian dialects can cause conversion quality deterioration.

The conversion tool converts text from old Latvian to new Latvian. In this context, the term "old Latvian" is used to refer to the orthography of printed texts written:

- Starting from the implementation of the orthography in which the second edition of the Bible in Latvian was written, i.e., around the mid-18th century;

- Till the implantation of the most influential Latvian orthography reform of 1908, i.e., around mid-20th century.

The "new Latvian" or "modern Latvian" refers to the orthography of contemporary Latvian.

It is important to note that the adoption of the suggested changes in writing were not immediate – it takes time for writers to learn and to get used to the newly proposed guidelines. And just as writers, readers also have to accustom to the new orthography, therefor transitioning to is gradual. For that reason, the old orthography does not have defined start or end date. The example:

- For the end point of the old orthography is fact that the last Latvian newspaper which adopted the changes of switching from Fraktur to Antiqua letters was 30 years after the invitation to do so (though for these changes, readers themselves requested a gradual transition);
- While for the starting point, despite being the most influential book of that time, the second translation of the Bible with its new standard of writing also did not changed orthography immediately. Furthermore, the conversion tool is able to convert texts that are even older than mid-18th century, however the quality of the conversion may deteriorate significantly.

There are 4 other changes in orthography which were introduced after the reform of 1908 and which were implemented in the conversion tool: the use of macron also in word locative forms which was added in 1920 reform; "ŗ" replacing with "r" and "ō" replacing with "o" from 1946 reform and abandoning the separation of "ch" and "h" which was introduced in 1957 reform.

For conversion tool this point of reference was chosen because:

- Before the second translation of the Bible into Latvian the orthography was inconsequential and way different from the one used today, while after, the orthography had few changes;
- For contemporary readers reading a texts written in this old Latvian causes far more difficulties than texts that were written in later decades. That is because the changes that were introduced later were relatively insignificant or quite rare compared to those introduced in 1908 reform, for example, changes in long vowels use in foreign words or some word spelling in cases of consonant alliteration.

As mentioned before, the conversion tool mostly uses conversion tables and exception wordlists

to avoid unnecessary conversion of words. Before these tables are used, the text is split into words. The exception to that are conversion of:

- Long vowels;
- “w”, “z”, “ſ” and “ƒ”;
- “ſch”, “ſh” and “ƒch”.

These conversions are done by applying conversion rules one by one to the whole text at once. By doing so, the conversion process is faster compared to the rest of the conversions where text splitting into words and every word processing is necessary. To increase the conversion speed for the rest of the conversions, only unique words are converted (i.e., two or more equal words from the text are converted only once) and then every instance of the unconverted word is replaced in the text with the converted version of it. Text splitting into words are necessary to be able to apply exception wordlists and conversion rules to specific part of the word, like prefixes or word stem.

For conversion table creation online sources were studied and from them conversion rules were derived. For the proposed task, this method is the most suited, since it takes little time resources and is quite precise compared to other methods like finding conversion rules by analyzing old Latvian (like corpus “SENIE” researchers); for one person who is not specialized in this task it would take a lot of time and the result would not be as accurate as the findings of other scientists.

Another method would be to build a machine learning model which would convert the text with a technique similar to that used in the 3.2. testing part with artificial intelligence chatbots. This method would not be suited for such tool for three reasons:

- It would require to recognize images of a text in old Latvian; correct the OCR errors; and by using these texts, make a large dictionary of old Latvian – modern Latvian words by manually converting each. These tasks involve time-consuming, monotone steps, where there is a high chance of making mistakes. In addition, interpretation by human convertor can influence the conversion;
- With this method, there is a high chance of missing such words that require special processing (such words in the conversion tool are processed with exception wordlists);
- This method lacks transparency. It is not possible to see conversion steps thus provide conversion options for the user.

2.2. Conversion rules

In this section, the rules for making conversion table are listed together with the source from which rules were derived. These rules will be used as bases for the conversion tool.

According to 1921 Monthly magazine of the Ministry of Education (Rīkojums par ortogrāfijas reformu, 1921) one of the steps for converting a text from old to modern Latvian is transliteration – replacing Fraktur with Antiqua letters. As mentioned in the introduction, the conversion tool has two options for inputting text – it can be either entered by keyboard or an image of a text can be uploaded. In the first case, the text it is already typed in Antiqua, while if user uploads an image of a text, by recognizing the text using previously mentioned Tesseract with the Fraktur data model, the text conversion from Fraktur to Antiqua is already done by Tesseract, i.e., the text is already denoted in Antiqua.

In theory, the exception here is letter "ſ" or long s with stroke. Although historically long s ("ſ", "S") once was a part of Antique letters (but disappeared over time), long s with stroke ("ſ", "ß") appeared later only in Fraktur texts. To be able to distinguish this letter from round "s", the long s with stroke are kept until later in next conversion parts where it will be replaced with the round "s". In Figure 2.1. the Fraktur to Antiqua conversion table can be seen – Fraktur letters (in the 1st and 3rd column) with the corresponding Antiqua letters (in the 2nd and 4th column). By this conversion table Tesseract transliterates the text.

Aaâà	Aaâà	Mm	Mm
Bb	Bb	Nn	Nn
Cc	Cc	Nn	Ņņ
Dd	Dd	Ooõ	Ooô
Eeêè	Eeêè	Pp	Pp
Ff	Ff	Rr	Rr
Gg	Gg	Rr	Ŗŗ
Gg	Ģģ	Ss	Ss
Hh	Hh	ſ	ſ
Iiîì	Iiîì	Sf	ß
Jj	Jj	Tt	Tt
Kk	Kk	Uuû	Uuû
Kk	Ķķ	Vv	Vv
Ll	Ll	Ww	Ww
Ll	Ļļ	Zz	Zz

Figure 2.1. Fraktur-Antiqua conversion table used in Tesseract (frakturs.lnb.lv, 2019)

The next step is transcription and adaptation – convert old graphemes to those used today. To do so, according to the "invitation in the case of Latvian spelling reform" in newspaper "Latvija" in 1908 (Druviete, 1990) and 1921 Monthly magazine of the Ministry of Education (Rīkojums par ortogrāfijas reformu, 1921):

- Graphemes "Tſch", "tſch", "Sch", "ſch", "Sch", "ſch", "S", "ſ", "S", "ſ", "Z", "z", "W" and "w" should be replaced with "Č", "č", "Š", "š", "Ž", "ž", "Z", "z", "S", "s", "C", "c", "W" and "w";
- Diphthong "Ee", "ee" should be replaced with "Ie", "ie";
- Vowel with following letter "h" should be replaced with vowel with macron (◌̄), just as vowels with diacritic marks used in old Latvian for long vowels, i.e., acute (◌́), grave (◌̀), circumflex (◌̂) and others.
- Word endings and suffixes which contain short vowels but should contain long vowels must be replaced with such containing long vowels.

According to "latviešu valoda" (Vanags, latviešu valoda, 2023):

- No diacritic marks should be used for long "o" and soft "r";
- "h" from "ch" should not be distinguished and "h" should be used;

According to "Latviešu rakstības attīstība" (Bergmane & Blinkena, 1986, pp. 70, 71, 72, 73, 75, 76):

- After replacing "S" and "ſ" with "Z" and "z", letters "Zt", "zt", "Zp", "zp", "Zd" and "zd" should be replaced with "St", "st", "Sp", "sp", "Sd", "sd";
- Not only "Tſch", "tſch" but also "Tſch", "tſch" should be replaced with "Č", "č";
- Not only "Sch", "ſch" but also "Zh, "zh" should also be replaced with "Ž", "ž";
- The last letter of prefixes and prepositions which after transliteration ends with letter "s", i.e. "līds", "ais", "bes", "is", "us", should be replaced with "z";

According to "Latviešu valodas gramatika" (Bērziņa-Baltiņa, 1994, p. 35) double consonants in words, except nasals and sonorous, i.e., "mm", "nn", "ņņ", "rr", "ll", "ļļ", should be replaced with a single consonant, except in compound words and cases when the first letter of word matches the last letter of prefix.

2.3. Technical solution

As most of the users encounter texts in old Latvian writing on the internet, the most convenient option for them to reach the conversion tool would be using a website. The only

limitations for using a website is that users must have an internet connection to reach the tool and they must have contemporary web browser. There are, however, users that no not like to read electronic texts. For them it would be great if they would be able to convert the text by taking a photo of it. A quick lookup is very beneficial, for example, when reading texts from the period when Latvian orthography was switching from old to modern Latvian; in this time many periodicals included texts written in both orthographies. For that, the conversion tool allows users not only to type the text but also to upload an image of a text.

The conversion tool can be divided into two parts: OCR and text conversion. In OCR part a text image provided by a user is recognized and transliterated in digital text written in Antiqua, while in text conversion part either the recognized text or text entered by a user is converted to modern Latvian.

The process of acquiring text from physical paper, there are many OCR (optical character recognition) software that recognize Latvian Fraktur, like ABBYY FineReader and ABBYY Flexicapture and many others like ExperVision, Omniscien language studio, Forcepoint DLP, YSoft SAFEQ who are partnering with ABBYY. Unfortunately, all of these are paid services. Although, there are several OCR programs which are open source, none of them provides a language module for Latvian Fraktur. Luckily, in late 2019 Valdis Saulespurēns launched a website (https://frakturs.lnb.lv) in which users can train language model to recognize Latvian Fraktur (Saulespurēns, 2022). This language module is made for Tesseract OCR engine – the most popular open source engine for optical character recognition of 2023 (affinda, 2023) which can be used in both websites and computer programs. Since most of Latvian fraktur letters are in German fraktur alphabet, V. Saulepurēns used German fraktur language module as a basis and supplemented it by training the module with the rest of the Latvian fraktur letters. Training was done by users who were asked to evaluate lines of text that OCR had recognized and edit them if these lines were recognized with errors. So far, since November 19th, 2019 till August 1st, 2023 with a help of crowdsourcing V. Saulespurēns has collected 144 544 reviewed and edited lines of text (frakturs.lnb.lv, 2019).

Creation of such model includes fine-tuning machine learning model so that it can perform better and better over time. After each prediction model makes, it is fine-tuned. One iteration is one complete cycle of training module on a dataset which users have created by editing text lines. So far (November 19th, 2023), the last version of this language module has 500 000 iterations.

In OCR part the Tesseract OCR engine with Latvian Fraktur language module version

Frak_LV_280721_500k is used to recognize images user provides. Images must contain text in Latvian Fraktur to be recognized correctly.

As for any OCR software, according to the article in ABBYY's support website (Nefedova, 2023), to achieve the best accuracy of recognition:

- Image resolution should be 300 dpi unless the size of a font is 10 pt – in such case the text will be recognized better using 400-600 dpi setting;
- If document is scanned, brightness should be set to 50% and image should be in grayscale.

The article mentions that for document that is photographed, in addition:

- Flashlight should be disabled;
- Anti-shake mode or tripod should be used;
- Optical zoom with manual focusing and manual aperture mode or aperture priority mode should be used.
- When taking a photo, the document should fit within the frame and the lens should be parallel to the document and perpendicular to the text, i.e., in the center of the photo;
- The document should be well lit with preferably daylight; if it is not possible not – with two sources of artificial lights;
- Preferably, the white balance should be set for the camera. cf.

However, from these recommendations, the only one that be applied automatically by the tool is to convert image to grayscale; the rest depends on the user. To do so, images uploaded by users are converted in grayscale before they are recognized with Tesseract. This is achieved by recalculating the image byte values from color to grayscale.

Another recommendation for users of the conversion tool is to take a photo of a text which contains only one text column. Otherwise the recognized columns will be merged together. And of course, instead of taking a photo of a screen, it is recommended to take a screenshot. For Windows users (which have Windows 10 or newer version) it can easily be achieved by pressing windows button together with "Shift" and "S" and then selecting the area for a screenshot, while for macOS users, the combination is "Command" with "Shift" and "4". When the image is created, it can be pasted in the website by pressing "Ctrl" together with "C" buttons for Windows users, while for macOS – "Command" and "C". Another way is to save the image and then upload it by pressing the upload image button in the website.

Since Tesseract is not a web-based software, to integrate it in web environment

Tesseract.js version 5 is used. Tesseract.js is a JavaScript library that works as a port for Tesseract engine and web environment. It is achieved by compiling Tesseract program from its C and C++ programming languages to binary WebAssembly code which then web browsers can use. WebAssembly (also known as WASM) is supported by all latest versions of web browsers, except Internet Explorer. In cases when WebAssembly is not supported, ASM.js is used. Just like with WebAssembly, using ASM.js Tesseract's source code from C and C++ programming languages are converted into a language that web browsers recognize, in this case JavaScript, using ASM.js rules and guidelines for optimized execution time.

After text is recognized or entered, it is converted into modern Latvian using Python programming language. Since Python is also not web-based, to use it in web environment PyScript is used. As it is said in their website (PyScript, n.d.), "PyScript is a framework that allows its users to create rich Python applications in the browser using HTML's interface and the power of Pyodide, WASM, and modern web technologies". Just as Tesseract.js, Pyodide is a JavaScript library that works as a port for Python and web environment. This is achieved by compiling python's CPython interpreter to WebAssembly binary code from which then Python code can be executed.

Starting from PyScript version "2023.11.1.RC3" another interpreter is introduced – MicroPython. According to the PyScript documentation (PyScript, n.d.), "MicroPython, when compressed for delivery to the browser, is only around 170k in size - smaller than many images found on the web". This interpreter is intended to be used on devices with limited resources, like microcontrollers or sensors. This means that the website will perform well even on slower devices and will be delivered to a user faster than with Pyodide. MicroPython, however, comes with fewer standard Python libraries than Pyodide and unlike Pyodide, it cannot import external libraries like "LatvianStemmer" which is used in the conversion tool. There however is a workaround – the library files can be downloaded, added in the file system and then the link to those files can be provided in the configuration.

PyScript is also planning to add a third option – SPy, a Python version where variables are typed statically (variable types must be defined) and is compiled directly to WASM. The static typing might, again, mean better performance in web browsers compared to Pyodide.

With Pyodide compiler, when users open website, they must wait few second until PyScript environment initializes. When user visits the conversion tool website for the first time, the websites loading time is around five to six seconds, but since some information necessary for the PyScript is stored in users web browser, each time user revisits the website, it takes only

around 3 second to load.

For text conversion, one python library (prewritten code that serves for some specific task) is used – “LatvianStemmer”; two modules (prewritten functions, classes, and variables, that can be used by other files) from Python Standard Library – “string” and “re”; and five self-made modules that contains various exception words, conversion dictionaries and verbs.

“LatvianStemmer” is a library created by Karlis Kreslin for his PhD thesis “A stemming algorithm for Latvian”. This library contains function “stem”, which stems nouns (except for plural genitive words of 5^{th}, masculine words of 6^{th} and all words in plural of 2^{nd} declination where consonants are palatal) and adjective morphology: for nouns – word case and number; for adjectives – number, gender, and definitiveness. For this function only one parameter is used – the word to be stemmed.

Originally this code was written in Java. The lightweight version of this tool can be found in Apache Lucene Github repository (https://github.com/apache/lucene). To Python environment this lightweight version was ported by Rihards Krišlauks and can be found in “PyPI” (Python Package Index, a package repository). For the conversion tool, the latest version, which for now is 1.0.2 of Aug 2, 2020, is used.

To remove needless punctuation marks and numbers from words when processing them, “string” module is used. “string” is a module that provides a collection of several useful constants, functions and classes for working with String type elements. In case of the conversion tool, from this module the variables “punctuation” and “digits” are used. “punctuation” contains a string constant of punctuation characters (“!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~”), while “digits” – a digits from 0 to 9.

To replace unconverted word in a text with the converted word, “re” module is used. “re” is a module for regular expression support in Python. Regular expression is a pattern with which “re” can match objects in text and then perform tasks like returning the objects found or text splitting based on the pattern. In case of the conversion tool, text replacing using function “sub” is used with 3 parameters – pattern from which a word will be found in a text, a word with which it will be replaced and the text in which the changes will be made.

3. PRACTICAL PART

By the previously mentioned tools, conversion rules and requirements the conversion tool can be made. In the first section of this part, the practical application of these tools within the conversion tool will be discussed, as well as the tool's workflow and the functions in which the conversion rules are embedded. In the second section, the conversion tool will be tested on texts from each decade starting from 1820 to 1930 (including) and the conversion accuracy will be compared with texts converted by three chatbots.

3.1. System architecture and design overview

To make it more accessible to the public, as mentioned before, the conversion tool is available for use on website. To make this website publicly available, it is hosted on GitHub Pages. GitHub provides its users with hosting services for free. This also includes free of charge domain which is under "github.io" domain name. The address for this website is https://artisozols.github.io/vd-jd/ and the files of the website are stored in GitHub repository ArtisOzols/vd-jd (https://github.com/ArtisOzols/vd-jd).

The repository consists of one HTML file ("index.html") and two folders:

- "global" in which program dependencies are stored and which consists of Tesseract's fraktur data model "Frak_LV_280721_500k.traineddata";
- And "resources", which consists of assets that are utilized by the website.

Folder "resources" consists of:

- "css" folder with "main.css" file for website's visual style;
- "js" folder with "main.js" file for website's functionality;
- "py" folder with "old_writing_to_new.py" file for converting old to modern Latvian texts and folder "lang_resources".

"lang_resources" is a folder containing assets that are utilized by the "old_writing_to_new.py" file. It consists of five Python modules:

- "letter_conversion.py" which contains dictionaries for grapheme conversion and related exception wordlists;
- "verbs.py" which contain verbs in infinitive form without the ending and which ends with a long vowel;
- "vcc_wordlist.py" which contains various lists of words and prefixes containing

a short vowel followed by double consonants;

- "old_w_dictionary.py" which contains a dictionary of often used words in old Latvian and their conversion;
- "w_prefixs_endings.py" which contains prefix and word ending related exceptions.

The visual representation of the file tree can be seen in figure 3.1.

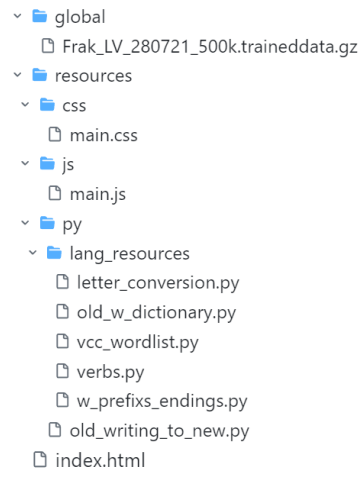

Figure 3.1. File structure of the website.

3.1.1. User interface

The visual style of the website is minimalistic. It consists of one page containing heading and two columns – one for text or image input, the other for converted text output. Since there are users who may want to use the website from their phones, a responsive design is developed. For mobile view, i.e., devices with screen size smaller than 800 px, the input column is aligned above the output column, while in desktop view, i.e., devices with larger screens, the input column is positioned on the left and the output column on the right side.

The input column consists of three buttons – "options", "Atpazīt attēlu" (recognize an image) and "clear all"; input text area and below that – several buttons by which old Latvian letter can be inserted.

To convert a text from old to modern Latvian, users have two options:

- They can upload an image of a text. To do this, user must either clicking on the button "Atpazīt attēlu" and select an image from the file storage or if the image is stored in the clipboard, user can past it. In both cases, "main.js" reads the image, creates HTML canvas with the image and displays it in place of input text area which is hidden beforehand. Then another canvas is created for

Tesseract. This canvas image is converted to grayscale and recognized. Finally, this text is passed to text conversion tool which returns the converted text and adds it into the output text area;

- The second option is to add the text manually. Besides typing with a keyboard or pasting text from a source, user can also add old Latvian letters by pressing the buttons at the bottom of the text area. This functionality is also implemented with "main.js" which, just as in previous case, passes the inputted text to conversion tool and outputs the converted text into the output text area.

The conversion happens immediately, while text recognition is a process that may take some time depending on the amount of text in the image.

In the input column there are two more buttons:

- "clear all" button by which user can remove both output and input, and if any, remove the canvas also unhiding the input text area;
- And "options" button by which user can open a dropdown list with text conversion options.

There are two text conversion options:

- The first option converts only "ee" to "ie" (and "ŗ" to "r" if the second option is selected) in a text. This is done because in the first half of the 20th century there were texts printed in an orthography very close to modern with exception – the diphthong "ie" was written with "ee".
- The second option allows users not to convert the "ŗ" to "r". This option is useful for users like previously mentioned scholars who wish to study linguistic elements of the text more in depth.

The output column, however, consists of only one element – output text area. In this element users, just as in the input text area, are able to edit the text. This is quite useful for two reasons: if user spots an incorrectly converted word which may also be possible due to OCR errors or when recognized text paragraphs are in the wrong order.

The structure and visual style of the website is achieved by index.html and main.css files. To better understand the layout of the page visually, the main visual components and their HTML tags can be seen in figure 3.2. While to better understand the interaction between the user and the website, the workflow diagram can be seen in figure 3.3.

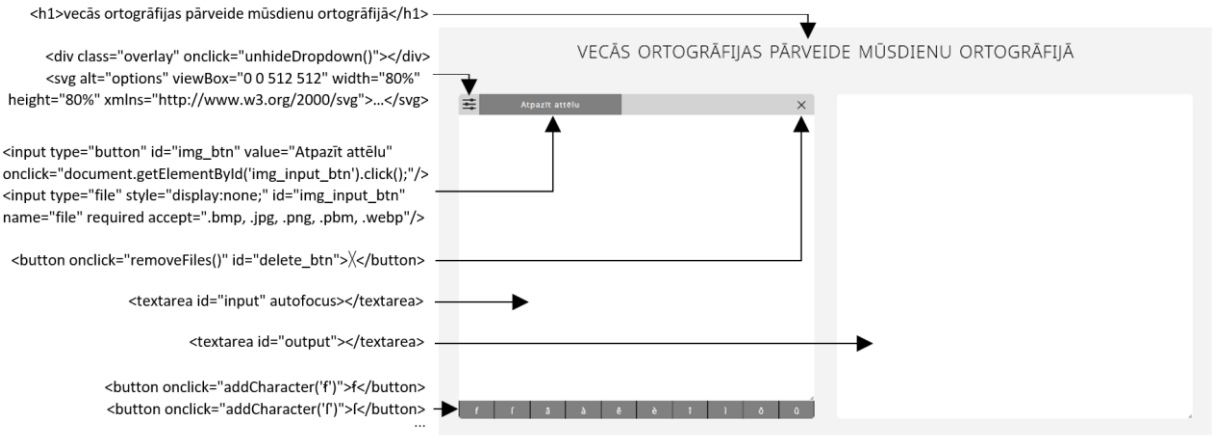

Figure 3.1. Main visual components of the website and their HTML tags

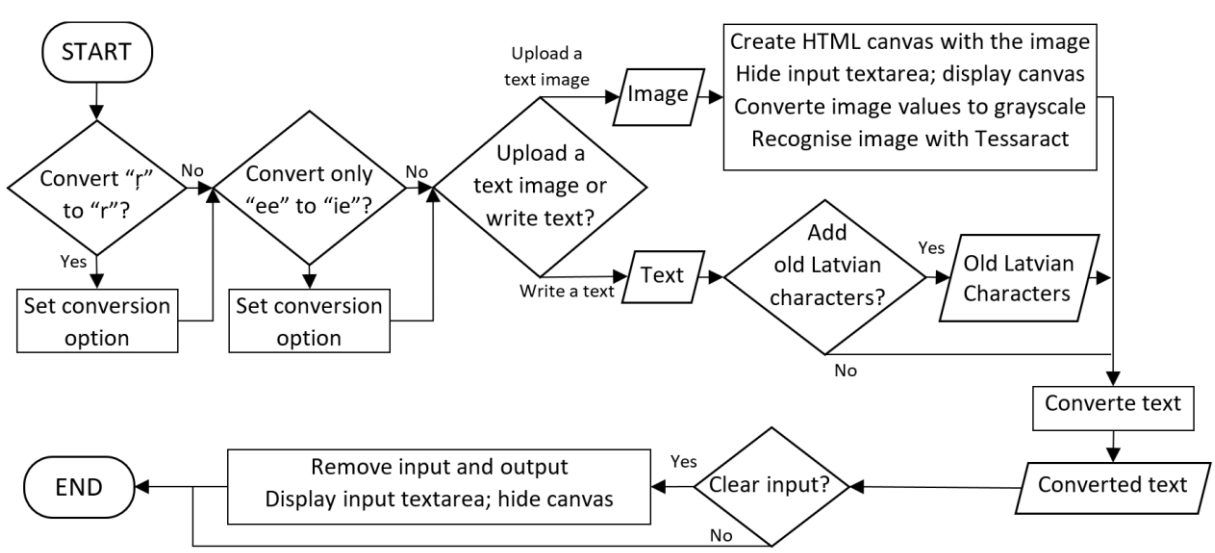

Figure 3.3. Workflow of the website

3.1.2. HTML

Besides the websites structure, index.html also includes "main.css" file in which the style of the websites is defined, and imports JavaScript libraries – Tesseract.js and PyScript. To import these libraries, two lines must be added in "index.html" within the "head" tag: "<script type='module' src='https://pyscript.net/releases/2024.1.1/core.js'></script> for PyScript and "<script src='https://cdn.jsdelivr.net/npm/tesseract.js@5/dist/tesseract.min.js'></script>" for Tesseract.js functionality. With these lines, PyScript version 2024.1.1 and Tesseract.js version 5 is imported. PyScript also invites users to include their CSS style in the website which can be seen, for example, when the PyScript is loading. This style, however, is not used since the

loading time is too short – users will not be able to manage to type in a text or upload an image before the PyScript finishes loading. Besides external libraries index.html at the end of the closing “body” tag also imports “main.js”: “<script src="./resources/js/main.js"></script>”.

3.1.3. PyScript

But before PyScript can be used, it must be set up. To do so, two more things must be added before the closing “body” tag – PyScript configuration and a “script” tag with a path to the Python file.

Since the conversion tool imports the “LatvianStemmer” library and five Python modules, PyScript first must prepare these files for use. The configuration can be defined in two ways:

- Within a “py-config” tag;
- Or within a “script” tag specifying three things – the path to a separate JSON or TOML file in which the configuration settings are defined and PyScript compiler with Python file to process (the last two will be discussed in next paragraph).

In this case “py-config” tag is used. The configuration settings for the conversion tool using the tag consists of two things:

- “packages = ["LatvianStemmer"]” which defines a list of Python packages that will be installed from “PyPI”;
- And “[[fetch]]” after which in new line follows all the paths to the modules used – “files = ["./resources/py/lang_resources/letter_conversion_dict.py", …].

The last thing PyScript needs is a path to the Python file that will be used in the website for conversion. For that, a “script” tag with two attributes (and attribute “config” in case a separate file for configuration is created) must be added:

- Attribute “type” specifying the Python interpreter that will be used – for now, either “py” for Pyodide or “mpy” for MicroPython. For this attribute “py” was set, since the conversion tool imports external libraries and since even after downloading library files, MicroPython raises errors;
- Attribute “src” specifying a path to the Python file for PyScript to process which in this case is “./resources/py/old_writing_to_new.py”.

3.1.4. JavaScript

Some HTML tags contains functions. These functions just as other functionalities are defined in “main.js” file. Main “main.js” parts are: conversion options; text conversion from the input text area; image processing, image recognition; old Latvian letter insertion; and input removal.

In the first part user sets the parameters for the conversion. The functionality for this part consists of three things:

- Dropdown menu opening – when user clicks on the “options” button, a function “unhideDropdown” is called which unhides the dropdown menu if it was hidden;
- Closing – event listener waits until user clicks anywhere else except the dropdown menu and hides it if it was displayed;
- Checkbox marking – when user clicks on a dropdown button, a function “checkUncheck” with parameter “ee” or “r” is called which changes the value of JS variable (which will be used later for conversion) and the checkbox value in HTML.

In the part of the conversion from the input text area, event listener waits until a text is added. When the value of the input text area changes, both the text and the checkbox variable information is passed as parameters to “convert” function, a function which converts a text from old to modern Latvian. This function was added in JavaScript by PyScript and is defined in the “old_writing_to_new.py” file. Finally, the converted text (text in modern Latvian), is set as the value of the output text area.

If user wants to add old Latvian letters, it can be done by clicking on the letter button at the bottom of the input text area. Each button calls function “addCharacter” with the character as the only parameter. This function:

- Reads the text from the input text area;
- Then, inserts the character where user’s cursor in the text was left;
- And returns the text by setting the input text area’s value to the new text.

In image processing part consists of two ways user can add an image; the image processing; the text conversion and text returning. For image uploading there are two event listeners:

- One waits until a file is added by the input button;
- The other waits until user pastes and checks if the last item in the clipboard is an image.

In both cases a function “processImage” is called with the image as parameter. The “processImage”:

- Changes the visual style of the output text area to let user know that an image is received and processed;
- Then another event listener waits until the image is completely read and when that happens two HTML canvases with the image are created – one is displayed in the place of input text area (which is hidden beforehand), while the other’s image is converted in grayscale and passed to function “image_to_text”.

The image conversion to grayscale is achieved by recalculating image pixel data. Each value in pixel data array consists of 4 elements – red, green, blue and alpha channels. To make the image grayscale, the average value of red, green and blue is calculated and replaced in each of the three. Then this new data is placed back on canvas and converted to data URL, a base64-encoded representation of the image data.

Lastly, this data is passed to the image recognition part where function “image_to_text” creates a Tesseract worker by this line:

“const worker = await Tesseract.createWorker("Frak_LV_280721_500k", 1, {langPath: "https://artisozols.github.io/vd-jd/global"}).

According to Tesseract.js documentation (Balearica, API, 2023), the worker takes three parameters:

- “langs” (in different documentation page also referred as “langCode”) – the name of the language model or models that will be used;
- “oem” – the OCR engine mode;
- “options” – an object with customized settings;
- “config” – an object with customized settings which are set before the worker is initialized cf.

In this case, first three are used. As “langs” parameter, the fraktur language model is specified, for “oem” – LSTM engine mode, while for options in dictionary “langPath” value is set to the URL in which the language model is stored. The documentation explains, “langPath – A string specifying the location of the tesseract language files. Language file URLs are calculated according to the formula langPath + langCode + '.traineddata.gz'. If langPath is not specified by the user, then the correct language data will be automatically downloaded from the jsDelivr CDN” (Balearica, Local Installation, 2023).

When Tesseract’s worker is created it can be used to recognize images. To do so, from

worker the “recognize” function is called. This function takes image data URL as it’s only parameter and returns the recognized text. This text along with the variable holding the information of the dropdown options is then passed to the “convert” function, and the result is placed in the output text area. To free up the memory after the image is recognized, Tesseract.js recommends to terminate all workers. This is done with the “terminate” worker function.

The last part clears the input and output. When user clicks on the “X” button in the input section, function “clearAll” sets both input and output text area values to nothing and removes the canvas if there is any. In this part, just as in most other parts, HTML element visual styles are changed as well.

3.1.5. Function “convert”

The file that PyScript imports which contains the function “convert” is “old_writing_to_new.py”. For JavaSript to be able to import “convert”, two lines must be added in the “old_writing_to_new.py” document: “import js” and “js.convert = convert”. While for “convert” – at the beginning of the document all five dictionary and exception word modules, together with “string” and “re” module and function “stem” from “LatvianStemmer” must be imported.

“convert” can be divided into two parts. Each part corresponds to a function – “edit_text” and “split_and_edit_words”. “convert” takes one String parameter “text”, which is the text that will be converted; and two Boolean parameters – “r” with default value true and “ee_only” with default value false. If “r” is true, all occurring “ŗ” in the text are replaced with “r” using Python’s “replace” function with parameters “ŗ” and “r”. The same is done with uppercase “Ŗ”. If parameter “ee_only” is true, all diphthongs “ee” are changed to “ie” using function “split_and_edit_words” with “text” and true as parameters (more on that in the third paragraph of the next subsection). The result is then returned without making any other conversions. Besides this, function “convert” contains two more functions that were mentioned before – “edit_text” and “split_and_edit_words”. Both of these functions take “text” as parameter, converts it and rewrites “text” value. After the second conversion, variable “text” is returned.

3.1.6. Function “edit_text”

The first main part that converts the text is function “edit_text”. Unlike the other,

“split_and_edit_words” function, this function makes changes in entire text at once. Since the text is not split into words and each word checked and then edited individually, the conversions in “edit_text” in most cases are completed much faster than “split_and_edit_words”. Despite that, with this it is not possible to limit conversions for exception words.

“edit_text” converts upper and lower case:

1. “ah”, “eh”, “ih”, “oh”, “uh”, “w”, “z” and “ſ” with function “change_key_val” and two parameters – the text and the “lengthmarks_w_z_ſ” dictionary from “letter_conversion.py” file (one of five module files);
2. “ſ” and its uppercase “ẞ” using Python’s “replace”;
3. “ş”, “zch”, “zh” and “sch” with function “change_key_val” and two parameters – the text and the “s_ž_š”, a dictionary from “letter_conversion.py” file.

Function “change_key_val” is a function that takes two parameters – a text that will be converted using “replace” function and a dictionary which values will be used for “replace” function. These values are first, used to converted lowercase letters; then, these letters are converted to uppercase with Python’s “upper” function and passed to replace again; and in third time if the value length is more than one character long, they are converted to title case using function “title” and used again in function “replace” to convert the text. For “ſ” conversion, function “change_key_val” is not used, because “upper” function does not convert “ſ” to its uppercase “ẞ” (the function converts “ſ” to “ſ”); therefor “ſ” is converted using “replace” with “ſ” and “s” as parameters while uppercase is converted with dictionary “s_ž_š”.

“lengthmarks_w_z_ſ” contains:

- “ah”, “â”, “à”, “á”, “ä” and their value “ā”;
- “eh”, “ê”, “è”, “é”, “ë” and their value “ē”;
- “ih”, “î”, “ì”, “í”, “ï” and their value “ī”;
- “ô”, “ò”, “ó”, “ö”, “oh” and their value “o”;
- “uh”, “û”, “ù”, “ú”, “ü” and their value “ū”;
- “w” and its value “v”;
- “z” and its value “c”;
- “ſ” and its value “z”.

“s_ž_š” contains:

- “ş”, “ꞩ” and their value “s”;
- “zch”, “zh” and their value “ž”;
- “sch” and its value “š”.

The order in which characters are replaced is very important: “z” and “Z” must be converted to “c” and “C” before “ſ” is converted to “z”; and its uppercase “S” must be converted to “Z”, before “ẞ” is converted to “S”. Only after these conversions, “zch”, “zh” and their uppercase version can be converted to “ž” and “Ž” and “sch”, “Sch” to “š”, “Š”.

The other part and function from “convert” is “split_and_edit_words”. This function splits the text into words and preform conversions on each word. It takes two parameters – “text”, a text that will be split, edited and returned, and a Boolean parameter “ee” with default value false.

If “ee” is set to true, “text” is split by space and each unit which contains “ee” when converted to lowercase is stored in a list. From each unit punctuation marks as well as numbers are removed from both sides of the unit. This list is then converted into a set, thus keeping only unique items. Since some words can contain punctuations in the middle of the word, these punctuations are retained (e.g., “km/h”, “Vīķe-Freiberga”). To remove the punctuation marks from both sides of the word, from built in module “string” variable “punctuation” and “digits” is used as a parameter in function “strip”. After creating a set of words, in variable “conv_fun” a function “change_ee” is stored.

If “ee” is false, a set of words is created in the same way, except they do not have to contain “ee”. Here “conv_fun” holds a function “edit_w”.

When “conv_fun” is set and a set of words is created, each word from the set is then converted to modern Latvian with function from “conv_fun” and then the word is replaced in the text with the converted word.

To avoid cases when replacing a small word in text replaces a part of a word which contains this small word (e.g., if all “ais” in the text is replaced to “aiz”, words that contain “ais” like “plaisa” will also be affected and incorrectly converted), from built in library “re” function “sub” is used with parameters “rf'\b{re.escape(w)}\b'” as the pattern, the modified word with which the match will be replaced and “text” in which replacing will be made. The pattern consists of “rf” which indicates that backslashes in the quotes are not part of escape characters and that within curly braces non string elements will be used; “\b” in quotes marks the boundary of a word and “re.escape(w)” between them ensures that “w” do not contain any special characters of regular expression.

“text”, just as set of words, can contain one and the same word in various letter cases. Since function in variable “conv_fun” (i.e., “change_ee” or “edit_w”) processes only words in lowercase, the word case is determined using functions “islower”, “istitle” and “isupper”, then

the word for the conversion in lowercased and the case of returned word is reverted. Finally, words (in original case) is replaced in the text. These three functions determine whether a word is lowercase, title case or uppercase, but unfortunately words in other word cases (like camel case) will be converted to lowercase.

3.1.7. Function “edit_w”

This function, just as all the following functions, takes “w”, a word to convert, as parameter; modifies it if needed and returns it. “edit_w” consists of several functions and a dictionary for converting a word:

1. “change_prefix” – a function that converts “s” to “z” in prefixes “is”, “us”, “ais”, “bes”, “līds” and “lids”; “z” to “s” in prefixes that ends with "s" followed by "t", "d" or "p" from word stem; and converts short to long vowel in prefix “lidz” and if the following letter is a vowel, in prefix “ja”;
2. “change_tš_tž” – a function that converts “tš” and “tž” to “č”;
3. “change_ee” – a function that converts “ee” to “ie”;
4. “change_zt_zd_zp” – a function that converts “z” to “s” if the following letter is “t”, “d” or “p”;
5. “change_ch” – a function that converts “ch” to “h”;
6. “change_vcc” – a function that converts double consonant to a single consonant if the previous letter is a short vowel;
7. “change_verb_ending” – a function that converts short to long vowel in the suffix and ending of a verb;
8. “change_suffix” – a function that converts short to long vowel in suffix “ak”, in suffix “ib” if the previous letter is a consonant, before suffix “šan” if the letter before the vowel is a consonant and both in suffix “taj” and one letter before it if before the vowel and prefix is a consonant;
9. “o_w_dict” dictionary from the “old_w_dictionary.py” module for word conversion.

The order in which these functions are used is important. Since all of them use exception wordlist (i.e., list of words to which conversions must not be carried out), these lists must also contain words in partially converted form to interrupt the conversion. For the first exception wordlists, the variants of one word can be quite many but for next wordlists, these variants will be fewer and fewer.

For example, if a word “islandietis” should be converted, “change_prefix” would not convert “is” to “iz” at the beginning of the word, since this word is in prefix exception wordlist. The same should be true if the word to be converted is “islandeetis”. Since diphthong “ee” is converted to “ie” after prefixes are converted, the word in this form is most likely to be converted by “change_prefix” than word in previous form and thus also requires to be added in the prefix exception wordlist. A visual representation of the example can be seen in figure 3.0.

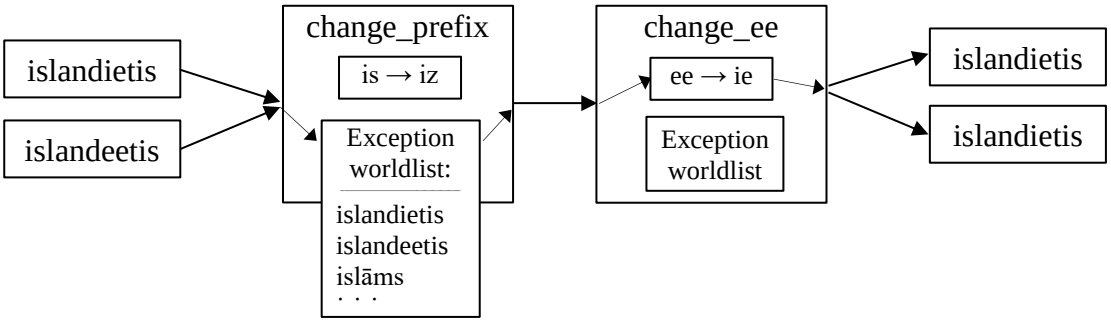

Figure 3.0. Example of unconverted words and their necessity in exception wordlist.

Exception wordlists

All previously mentioned eight functions use exception wordlists – lists of words to which a conversion must not be applied. These wordlists are stored in 4 modules:

- “letter_conversion.py” for functions “change_tš_tž”, “change_ee”, “change_zt_zd_zp” and “change_ch”;
- “vcc_wordlist.py” for function “change_vcc”;
- “w_prefixs_endings.py” for functions “change_prefix”, “change_v_taj”, “change_c_ib” and “change_ak”;
- “verbs.py” for “change_verb_ending”.

The last module is previously mentioned “old_w_dictionary.py”.

All wordlists, except the lists in “verbs.py”, were created using open data provided by the online explanatory and synonym dictionary “Tēzaurs”. To create a set of exception wordlists, first, a wordlist with words and their parts of speech tags is created. To do so, first the dictionary data should be downloaded in XML file found in “Tēzaurs” dataset. From this file each “form” tag is found and iterated. In each iteration the text from “form” child tag “orth” (which is the entity) together with part of speech annotation is stored in a list “entities”.

This dataset contains words with such part of speech annotations: adverb, verb,

participle, interjection, noun, uninflected word, particle, preposition, residual, conjunction, abbreviation, numeral, pronoun, proper noun and adjective. From these annotations, words with such can be stemmed: noun, adjective, numeral, pronoun, proper noun, verb and participle.

The value of part of speech depends on elements with attribute “Nelokāms vārds”, “Īpašvārda veids” and “Vārdšķira” found in “form” tag:

- If there is a child with attribute “Nelokāms vārds”, the part of speech annotation is “Nelokāms vārds”;
- If there is a child with attribute “Īpašvārda veids, the part of speech annotation is “Īpašvārds”;
- If there are none of the previous two elements in the “form” tag, the part of speech annotation is the text of a child element with attribute “Vārdšķira”, while if the element does not contain any text, the value is “” (nothing).

Then “entities”, the list of entity with its part of speech tag, is iterated. Unfortunatly, each entity does not contain only one word – they can contain multiple (e.g., “Alūksnes ezers”). Therefor in each iteration, entity is split by space (if any) and each part that ends with a dot is added in the list “wordlist” with dot removed, together with “Saīsinājums” as annotation, but parts that do not are added with either its part of speech annotation if the entity contained only one word or “” (nothing) if the entity contained multiple words. The illustration of this “entities” list iteration with examples can be seen in figure 3.5.

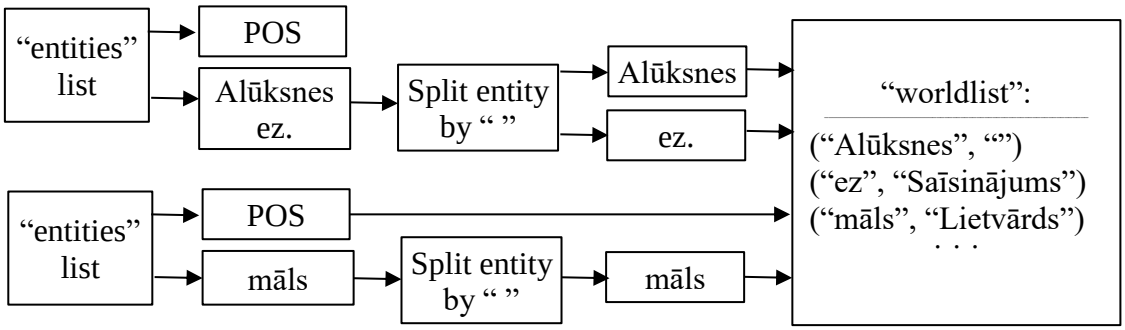

Figure 3.5. Example of two iterations of “entities” list

From the “wordlist”, two other lists are created: “st_wordlist” – a list with words which part of speech annotations is either “Lietvārds”, “Īpašības vārds”, “Skaitļa vārds”, “Vietniekvārds”, “Īpašvārds”, “Darbības vārds” or “Divdabis”; and “nost_wordlist” which contain the rest of the “wordlist” words. Then all words in “st_wordlist” list is stemmed. From these two lists, exception wordlists are created.

Function “change_prefix”

This function consists of four parts. The first part converts words with at least 5 letters which start with “vizp”, “vizd”, “vizt”, “puzt”, “puzd”, “puzp”, “līdzāzt”, “līdzāzd”, “līdzāzp”, “vispuzt” or “vispuzd”. If “w” (word to be converted) starts with one of these elements (e.g., “puztumšs”), in its element part, the “z” in “zt”, “zd” or “zp” at the end (i.e., one letter before the last in the element) is replaced with “s”. Since the last two elements are combination on two, elements in word are checked in such order to first convert separate element and only then element combinations, otherwise word may not be converted completely. These letter conversions for now are applied only to word beginning but for the rest part of the word they will be performed later with function “change_zt_zd_zp”.

The second part of “change_prefix” converts a short to long vowel in prefix “ja” if the following letter is vowel (except letter “u” since there are far too many words in Latvian that start with “jau”). To do so, if “w” starts with “jaa”, “jae”, “jai”, “jao”, “jaā”, “jaē”, “jaī”, “jaū” and “w” is not in “ja_v_nost” and stemmed in “ja_v_st” exception lists, the first two letters of “w” is rewritten with “jā. For “w” stemming, from LatvianStemmer function “stem” is used with “w” as the parameter.

“ja_v_nost” and “ja_v_st” are exception wordlists that are created from previously mentioned “st_wordlist” and “nost_wordlist” lists. “ja_v_nost” consists of words from “nost_wordlist” which starts with “ja” and a vowel, except “u”, while “ja_v_st” is made of words from “st_wordlist” with the same criteria.

The third part converts prefixes “pee” to “pie” and “ee” to “ie”. To do so, first prefixes are removed from “w” using function “remove_prefix” with parameters “w” and true. This function will return two values: “w” without all modern Latvian prefixes; and the prefixes that were removed. The returned values are stored in “w” and “pref”. If “w” starts with “pee”, its first three letters are replaced with “pie”, while if “w” starts with letters “ee”, they are replaced with “ie”. Finally, “w” is rewritten with “pref” concatenated with the value of “w”.

Prefixes from words are removed using function “remove_prefix”. It takes two parameters: “w”, a word to be split, and “rm_all”. If “rm_all” is false, only one prefix from “w” is removed, while if true – all prefixes are removed. This parameter by default is set to true. “remove_prefix” takes each element from “pref” list and checks if prefix is at the beginning of “w”. If it is not, then next prefix is taken, while if it is, this prefix is removed from “w”. The next step is to check if the word has two or more characters. If it has less or if the “w” is checked with all the elements from “pref” list and none matched, unmodified “w” and “” is returned,

else the modified word and prefix is return in case if “rm_all” is false. But if “rm_all” is true, function “remove_prefix” is called again with “w” (to which a prefix is removed) and the result is stored in variables “new_w” and “new_p”. Finally, “new_w” and the removed prefix concatenated with “new_p” is returned. The workflow of function “remove_prefix” can be seen in figure 3.6.

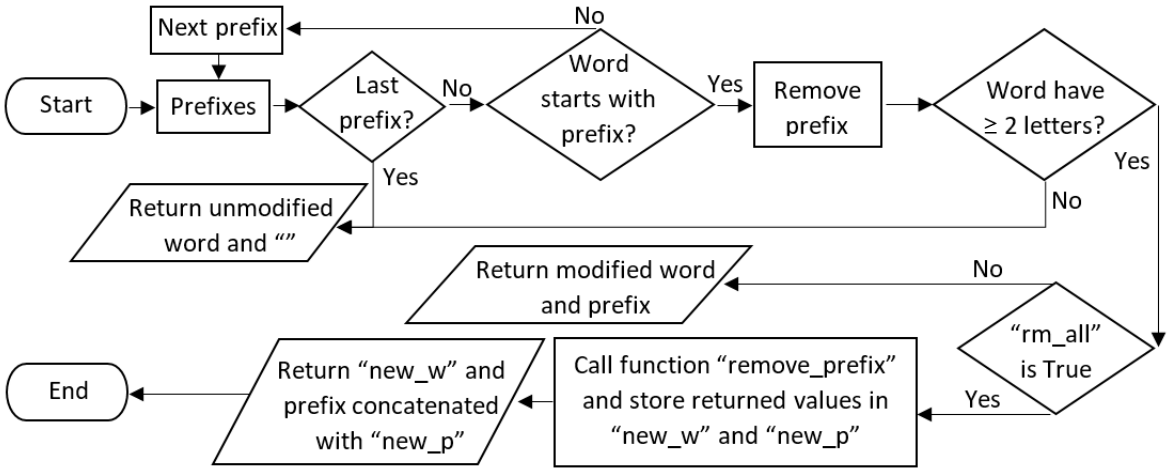

Figure 3.6 Workflow of the function “remove_prefix”.

“remove_prefix” requires “pref”, a list in “w_prefixs_endings” module, which contains all prefixes with negation, debitive form, superlative and prefixoids. For simplicity, all of these elements will be referred to as prefixes in the following text. According to Daiga Deksne in Latvian literary language there are 11 prefixes which can be attached to words, as well as several prefixoids cf. In the words of Deksne: “… prefixoids – language units that have their own lexical meaning and are used as prepositions in word combinations (through a window), but they can join verbs as prefixes and change the value of the verb type (lookthrough).” (Deksne, 2021, pp. 8, 18, 146-150, author’s translation) The most common prefixoids she mentions are “caur”, “līdz”, “līdzās”, “pus”, “pret”, “pretim”, “atpakaļ”, “priekš” and all eleven prefixes are “aiz”, “ap”, “at”, “ie”, “iz”, “no”, “pa”, “pār”, “pie”, “sa”, “uz” (Deksne, 2021, pp. 8, 18, 146-150) cf.

The last part of the “change_prefix” converts prefixes “is”, “us”, “ais”, “bes”, “lids”, “lidz” to “iz”, “uz”, “aiz”, “bez”, “līdz” and again “līdz”. To do so, for each prefix to be converted (later they will be referred to as old prefixes) exception wordlists together with the modified version of the prefix is stored in variable “d”. Then prefix from “w” is removed and stored in variable “pref”. Finally, if these conditions are met, “w” is rewritten with the modified version of the prefix (“d[4]”) concatenated with the value of “w” from which old prefix is

removed beforehand. These conditions are:

- “w” must start with old prefix;
- It should not be in exception wordlist “d[0]” and stemmed in “d[1]”;
- “pref” concatenated with “w” should not be in “d[2]” and their stemmed version in “d[3]” wordlists.

The visual representation in a workflow for the last part of function “change_prefix” can be seen in figure 3.7.

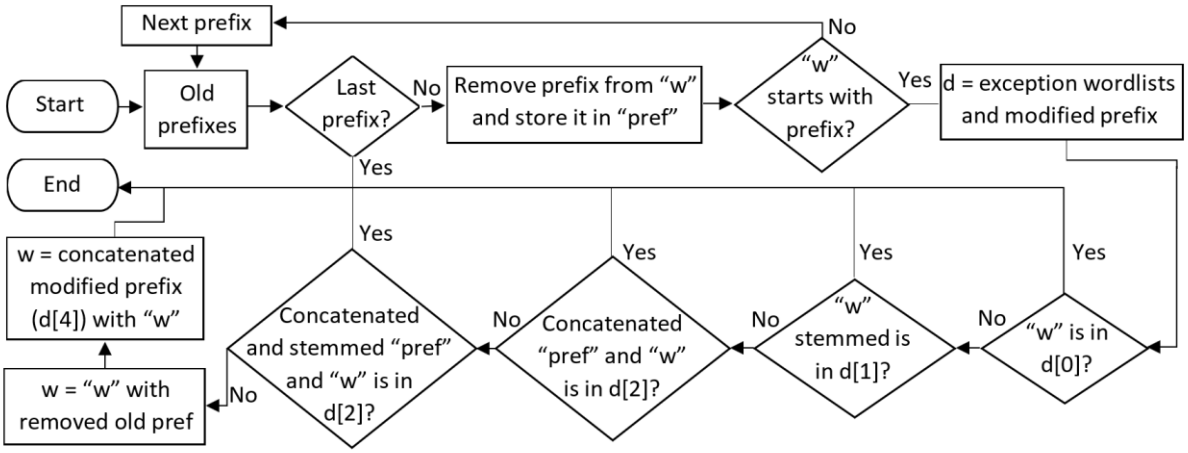

Figure 3.7. Workflow for the last part of function “change_prefix”.

Variable “d” contains four exception wordlists. For prefix “is”:

- In “d[0]” the “is_nost” list is stored. This list contains words that start with “is” (e.g., “istaba”). This list is created from “nost_wordlist” by taking all the words that have more than two characters and starts with “is”.
- “d[1]” is the “is_st” list. It contains stemmed words that start with “is” and is acquired in the same way but from “st_wordlist” wordlist.
- “d[2]” is the “pref_is_nost” list and it consists of words from “nost_wordlist” that start with one or many items from “pref” list followed by “is” (e.g., “paisums”).
- “d[3]” is the “pref_is_st” list and it is created in the same way as “pref_is_nost” but from “st_wordlist” list.

With function “remove_prefix” word conversion is possible even if a word contains multiple prefixes (e.g., “pusaisvērts”). However, it may also remove parts of word that are not actually prefixes. For instance, if the word “salīdzināt” is passed to the function, it would return “ināt” as word and “salīdz” as prefix. For that reason, wordlists stored in “d[2]” and “d[3]” are needed

– to check if such words (with prefix followed by old prefix) are not correct words and to which conversion should not be applied.

In the same way exception wordlists are created for “us”, “ais”, “bes”, “lids” and “lidz”. Since there are no words in whole dataset that starts with “pref” elements followed by “lidz”, “lids” from “nost_wordlist” and from “st_wordlist” list – “lidz”, “lids”, “ais” and “bes”, the specific lists for these elements are left empty.

Function “change_tš_tž”

This function converts “tš” and “tž” to “č”. After function “edit_text” converted letters “š” and “ž”, it is now possible to convert letter “č”. Since all prefixes are converted, they can be correctly separated from words. This is done with function “remove_prefix” and “w” as parameter. The rest of the word is stored in “w” but the prefix part, in “pref”. Now these two are checked for multiple things:

- If the last letter of “pref” is “t” and the first of “w” is “š”;
- If “t” concatenated with “w” is not in “tš_nost_pref” and stemmed in “t_š_st_pref” lists.

If so, “w” is rewritten with three things that are concatenated together:

- “pref” letters without the last;
- Letter “č”;
- “w” letters without the first one.

Then prefix is again split from “w” and it is checked:

- If “w” contains “tš”;
- If “w” is not in “tš_nost” and stemmed in “tš_st” exception lists.

If so, “tš” in “w” is replaced with “č”. Now instead of “š”, the same is done with “ž”. After these conversions, “pref” is concatenated with “w” and returned.

With the same method wordlists were made. Each word from “nost_wordlist” were divided in two with function “remove_prefix” and the returned values were stored in “w” and “pref”. Then “t” concatenated with “w” letters, except the first, are added in two lists:

- “tš_exc_nost_pref” list if “pref” ends with “t” and “w” starts with “š”;
- “tž_exc_nost_pref” list if “pref” ends with “t” and “w” starts with “ž”.

The same is done with words from “st_wordlist” list and stored in “tš_exc_st_pref” and “tž_exc_st_pref” lists. Finally, each word from “nost_wordlist” if it contains “tš” is added into

“tš_exc_nost” list while if it contains “tž”, it is added into “tž_exc_nost” list. The same is done with words from “st_wordlist” list and stored in “tš_exc_st” and “tž_exc_st” wordlists.

Function “change_ee”

This function converts diphthong “ee” to “ie”. To do so, prefixes are removed from word and word is stored in “w” and prefix in “pref”. To ensured that words which starts with “nie” (e.g., “niedre”) would not be converted, it is necessary to check:

- If “pref” ends with “ne” and “w” starts with “e”;
- If “ni” concatenated with “w” is in “nie_nost” or stemmed in “nie_st” wordlists.

In such case two elements are concatenated and returned by this function:

- “pref” to which the last letter is replaced with “i”;
- “w” to which “ee” is replaced with “ie” beforehand if there are any occurrences.

Wordlists “nie_nost” and “nie_st” contains words that starts with “nie”. “nie_nost” was created from “nost_wordlist”, while “nie_st” from “st_wordlist”.

Function “change_zt_zd_zp”

This function converts the rest of the “zt”, “zd”, “zp” to “st”, “sd” and “sp”. For that, first, all prefixes are removed from the word with “remove_prefix” and its values stored in “w” and “pref”. Then three things are checked:

- If “w” contains “zt”;
- If “w” is not in “zt_nost” and stemmed in “zt_st” wordlists.

If so, “zt” in “w” is replaced with “st”.

Within these exception wordlists there are words that contains one of “zt”, “zd” or “zp” that should and one that should not be covered, for example “strazds”; in old Latvian after previous conversions this word could be written as “ztrazds”. To convert the correct element, it is first checked:

- If “w” contains “zt”;
- If “w” is in “z_tdp_dict_nost” or stemmed in “z_tdp_dict_st” dictionaries.

If true, the value of “w” is rewritten with the value from the dictionary. After that, instead of “zt” this process is repeated using “zd” and then “zt” with their wordlists and dictionaries. Finally, after “w” is converted, it is rewritten with “pref” concatenated with the value of “w” and then returned.

So, “zt_nost” contains words:

- Which are not stemmed;
- Contain “zt”;
- Contain “zt” to which “st”, “sd” or “sp” are converted to “zt”, “zd, “zp” if any.

“zd_nost” and “zp_nost” contains “zd” and “zp” accordingly, while “zt_nost”, “zd_nost” and “zp_nost” – the same but containing words that can be stemmed.

To create both wordlists and a dictionary, for words containing “zt”:

1. All prefixes from words of “nost_wordlist” list is removed.
2. If such words contain “zt”, they are added in the “zt_nost” wordlist.
3. If such words also contain “st”, “sd” or “sp”, variable “w_old” is created which contain the word to which “st”, “sd”, “sp” is replaced with “zt”, “zd” and “zp”.
4. In “z_tdp_dict_nost” dictionary, “w_old” is added as a key, and the word (without conversions) as a value.
5. The value of “w_old” is also added in the “zt_nost” wordlist. To ensure that a word when converted would reach the “z_tdp_dict_nost” dictionary.

For “zt_st” the process is the same, except “st_wordlist” wordlist is used. In the same way wordlists and dictionaries are created for “zd” and “zp”.

Function “change_ch”

This function converts “ch” to “h”. It does it by checking if “w” contains “ch”. If it does and if “w” is not in “ch_nost” and stemmed in “ch_st” wordlists, the function returns “w” to which “ch” is replaced with “h”.

To create “ch_nost” wordlist, “nost_wordlist” list is split in two sublists – one containing words with letters “ch” (“ch_words_nost” wordlist) and the other that does not (“no_ch_words_nost” wordlist). These two lists are needed, to avoid adding words in exception wordlist that are outdated. Unfortunately, “Tēzaurs” dataset also contains some words with “ch” which today are written with “h”, (e.g., “chronometrs”). Therefor from “ch_words_nost” list (words containing “ch”) each word is checked if it when converted (“ch” replaced with “h”) can also be found in “no_ch_words_nost” list (between the rest of the words). If so, that means that the word is outdated and not needed, while if not – this word is added in the “ch_nost” wordlist. The same is done to make “ch_st” wordlist, except words from “st_wordlist” list is used.

For example, “Tēzaurs” dataset contains word “architekts” but this word will not be

added in the exception wordlist since the word, when converted, matches word “arhitekts” which also can be found in the dataset.

Function “change_vcc”

This function converts double consonants to a single consonant if the previous letter is a short vowel. This function works for words longer than 2 letters; words that are shorter are returned without changes. First, prefixes are separated from the word with function “remove_prefix” and the output is stored in “w” and “pref”. This function consists of three parts. Each part converts double consonant to a single:

- If “pref” ends with short vowel and a consonant, and “w” starts with the same consonant;
- If “pref” ends with short vowel and “w” starts with the double consonant;
- If a short vowel followed by double consonant is in “w”.

The first part checks “pref” if it ends with a short vowel and a consonant and if “w” starts with the same consonant. If they do and if “w” is not in “pref_vcc_w_nost” or stemmed in “pref_vcc_w_st” wordlists, the last letter of “pref” is removed. A visual schema of this the beginning of the “change_vcc” function and its first part can be seen in figure 3.8.

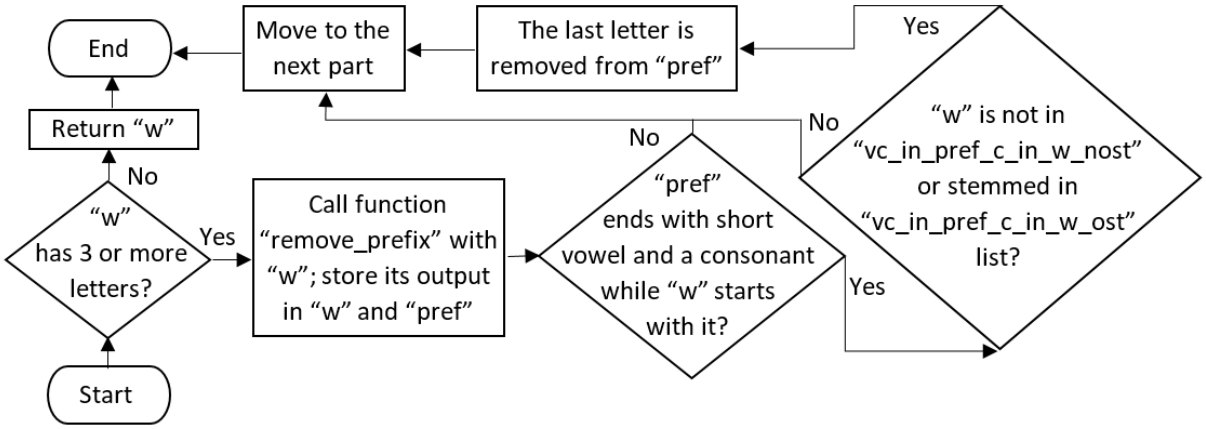

Figure 0.8. The schema of the beginning of the “change_vcc” function and its first part.

The wordlists are created in similar way. For “pref_vcc_w_nost” wordlist prefixes from “nost_wordlist” words are separated. If prefix ends with a short vowel followed by a consonant and if “w” starts with the same consonant, “w” is added in the “pref_vcc_w_nost” list. For “pref_vcc_w_st” the method is the same but “st_wordlist” words are used.

Unfortunately, this conversion also corrupts words that are not mentioned in the exception wordlists. These two wordlists contain only those words that in Tēzaurs dataset started with prefix elements (e.g., “prettiesisks”) but word with prefix can be formed from all the nouns, adjectives or verbs. Ideally this list would contain all of these words but if so, the list would be too long – each word processing would take a very long time. Although this conversion corrupts some words, after performing a test of the test dataset, this part of the function increased the accuracy of the end result and therefor it was kept.

Next part converts double consonants to a single one if it can be found in the last letter of “pref” and the first two letters of “w”. For that it is checked:

- If “pref” ends with a short vowel
- If the first two letters of “w” are the same and are consonants
- If “w” is not in “vcc_in_pref_w_nost” and stemmed in “vcc_in_pref_w_st” lists.

If it is so, the first letter from “w” is removed. The visual representation of this part can be seen in figure 3.9.

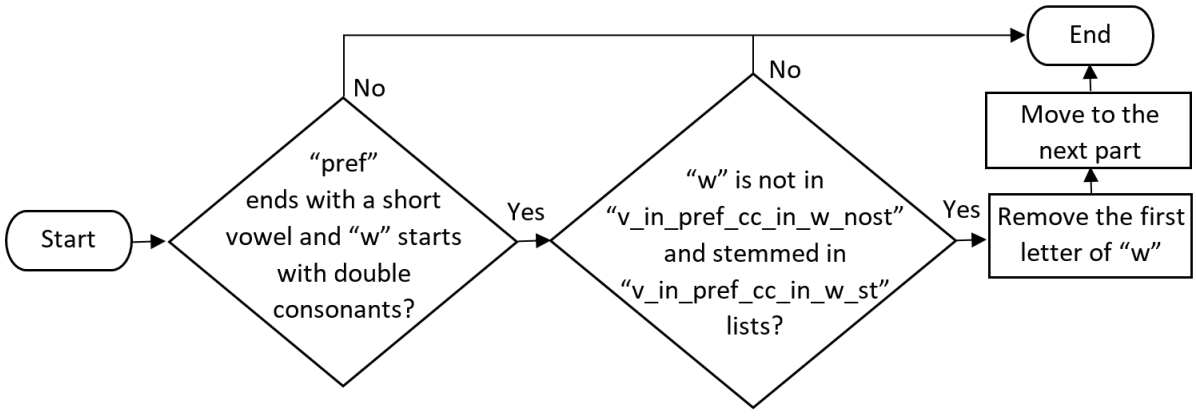

Figure 3.9. The schema of the second part of the “change_vcc” function.

Mainly these wordlists are made of words to which function “remove_prefix” removed word beginning and which without it starts with double consonants (e.g., “savvaļa”, “pannēt”). For “vcc_in_pref_w_nost”, words from “nost_wordlist” list were used. From each word prefix was removed. If such word starts with double consonants, the word was added in “vcc_in_pref_w_nost” list. The same was done to make “vcc_in_pref_w_st” list, except words from “st_wordlist” list were used.

The last part check if “w” contains a vowel followed by double consonants. This is done within a loop where:

- First, the function “check_vcc” is called and its output is stored in “vcc”;
- Then, it is checked if “vcc” does not contains any value and if function “check_vcc_exc” when called with “w” as parameter returns true;
- If any of the cases are true, “pref” concatenated with “w” is returned and the loop stops there;
- If not, the first occurrence of “vcc” in “w” is replaced with the first two letters of “vcc”.

Then this whole process is repeated – function “check_vcc” is called; if it did not return any value or if “check_vcc_exc” returns true, the loop ends by returning “pref” and “w” concatenated; if not – another occurrence of “vcc” (the output of “check_vcc”) is replaced with its first two letters; and so on.

Function “check_vcc” checks each letter of the word passed. If a letter is the same as previous letter, if both are consonants and if a letter before these are a short vowel, the vowel with both consonants are returned.

Function “check_vcc_exc” checks if the word passed is in one of the exception lists. This function:

- First stems the word;
- From the end of the stemmed word suffix “ošāk”, “āk” or “oš” is removed if any;
- Then, if stemmed word is in “st_vcc” or non-stemmed word (with suffixes) is in “nost_vcc” wordlists, true is returned.

The workflow of this part of “change_vcc” can be seen in figure 3.10.

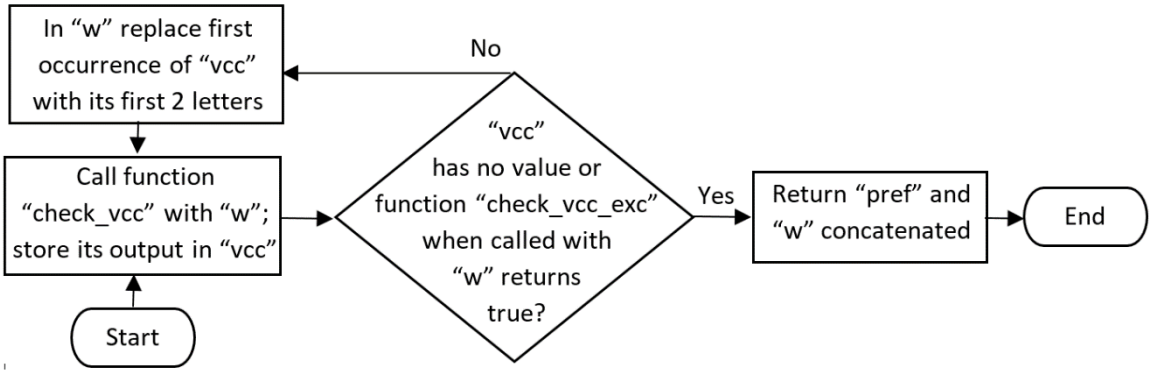

Figure 0.10. The third part of the function “change_vcc”.

“st_vcc” wordlist is made from “st_wordlist” words containing vowel followed by double consonants. This list is updated with another group of words – the second, fifth and sixth

words in plural form. This is necessary since “LatvianStemmer” stems these words different depending on their inflection, e.g., word “diennakts” will be stemmed as “diennakt” while word “diennakšu” will be stemmed as “diennakst”. This group consists of words that ends with “is” and “e” (representing words of second and fifth declination) and such words that in Tēzaurs dataset were marked as words of the sixth declination. Since inflection “s” is an inflection of both sixth and fifth declinations, there are no rules that could be applied to distinguish these two declinations, therefor for the sixth declination only those words in dataset whose declension was marked were used. From this group, only those words containing vowel with double consonant were kept. Then each word was manually reviewed; supplemented with a plural form; stemmed and added to “st_vcc” wordlist.

“nost_vcc” is made from words of “nost_wordlist” list which contain vowel followed by double consonants. Another group of words were added to this list. This group consists of words from “wordlist” list (list of all words of the dataset) which ends with vowel followed by double consonants (e.g., “buss”). Such words if stemmed would lose the last consonant, therefor they are also added in the non-stemmed wordlist.

Function “change_verb_ending”

This function changes short to long vowel before suffixes and in inflections of verbs. It converts short to long vowel:

- In verbs of all conjugation infinitive forms if the vowel is before inflections “ties” and “t” (e.g., “strādat” to “strādāt”);
- In all inflections of verbs of III conjugation, except for 3rd person future form with verbs which in infinitive form ends with “ēt” and “ūt” (e.g., “lasis” will not be converted to “lasīs”);
- In inflections of II conjugation, except for verbs in 3rd person future and present (which is also 2nd person present) forms, which in infinitive ends with “āt”, “ēt”, “ūt” or “īt” (e.g., “doma” will not be converted to “domā”); and forms with suffix “j” followed by long vowel (since these forms, which correspond to the past tense, without the long vowel i.e., unconverted, match the present form).

Exceptions in verbs which in infinitive ends with “āt”, “ēt”, “ūt” or “īt” are made, since some verbs in previously mentioned inflections when not converted (with short vowel in the ending) matches nouns. Since it is not possible to distinguish both without a context, verbs in these forms will not be converted. For example, word “būves” should not be converted to third

person future form “būvēs” since “būves” is also a plural noun. The formation of nouns from verbs of III conjugation is quite rare compared with verbs of II conjugation. These infinitive verb forms were chosen because nouns could not be formed from other verbs.

The other exception is made for those inflections of II conjugation verbs that ends with the suffix is “j” and the following letter is “ā”. This exception is made because when the verb is not converted, the inflection matches other forms of the verb. For example, “žēlojāt” is a verb in past plural form; without the long vowel in inflection this word matches its present form. Since such words already exist in modern Latvian, the verb inflections are not converted.

Verb conversion in non-infinitive forms are done only to verbs of II and III conj. since I conj. verbs can have multiple variations in verb stem depending on tense or inflection. To convert such words, for some there would be needed individual rules of conversion. Instead of making individual rules, other rules can be made which would impact the conversion to a greater extent.

It is also worth mentioning that the conversions are made to only those verbs found in Latviešu valodas ilgtspējas atbalsta partneri (Latviešu valodas ilgtspējas atbalsta partneri, n.d.) website. In march, 2024 on this website there were 3137 verbs. These verbs are categorized by their conjugation and each are inflected in all forms.

The verb conversion is done in two parts. The first part converts verbs in infinitive, II and III conj. past and future forms and II conj. present form. It is done by first checking if “w” ends with one of “verb_end” endings.

“verb_end” is a list of all II and III conj. verb:

- Inflections (including infinitive);
- Suffix with inflections;
- And both written with short vowels, to ensure that not only vowels before the endings are converted but also endings themselves.

From this list endings “ēs”, “ās”, “īs”, “os” and “ūs” with their short vowel forms are removed, since these endings correspond to a 3rd person future form (III conj. verbs with ending “ās” will be converted in the second part of this function) and since “os” do not contain a vowel that should be converted. Finally, this list is sorted by the length of the ending to ensure that the whole ending of the word is matched and not just a part (like only inflection).

So, if “w” ends with one of “verb_end” endings, it is then checked if the ending is “ties” or “t”. If so, the word is modified with function “mod_verb” with parameters “w”, “all_verbs” (a list of verbs in infinitive form to which the end “ties” or “t” is removed and which then ends

with a long vowel) and the ending matched. If this function returns a value, the whole function ends by returning it. If the ending of "w" is not "ties" or "t", function "mod_verb" is called with "w", "ii_iii_verbs" ("all_verbs" list without the I conj. verbs) and the ending. If this function returns a value, "w" is rewritten with it.

To convert the past tense verb inflections of verbs of III conj. in both 3rd person and plural forms, it is checked if "w" without the ending is in "iii_verbs" list ("ii_iii_verbs" list without the II conj. verbs) and if it ends with "jamies", "jaties", "jam", "jat" or "jas". If so, the first two letters of the ending in "w" are replaced with "jā". Then this function continues with "w" on the second part.

Function "mod_verb" converts short to long vowel before the ending if "w" after conversion is in the given list. This function takes three parameters – "w", "verb_list" and "end". It first, removes "end" from the end of "w". Then it is checked if:

- "w" contains two or more letters;
- Its second to last letter is consonant;
- Its last letter is a short vowel.

If so, the last letter in "w" is converted to a long vowel. Then if such word is in the "verb_list" list, "w" with "end" are concatenated and returned. If it is not – with function "remove_prefix" and parameters "w" and false, one prefix is removed from "w". If prefix is not returned, "mod_verb" returns nothing, else it is checked if "w" without a prefix is in "verb_list" list. If it is, all prefixes with "w" and ending are concatenated and returned, but if not – another prefix is removed, and so on.

The second part of "change_verb_ending" converts "a" to "ā" in verb endings of the III conj. present forms, i.e., 3rd person and plural of the III conj. 1st group. (in infinitive form these verbs end with "īt", "īties", "ināt", "ināties"). This is done by first checking if "w" ends with one of "verb_end_III_first" endings.

"verb_end_III_first" is a list of verb endings of III conj. 1st group in:

- Present and past plural forms;
- 3rd person present form if in infinitive verb ends with "īties" and "ināties".

These endings include inflection, suffix and the letter before the suffix, except for inflections "ām", "āt" and "ās". For all of these ending, "ā" in inflection is replaced with "a" since this list is used to find verbs that need conversion in inflections. This list is also sorted by the length of the ending to match the whole word ending and not just a part.

So, if "w" matches one of the endings, it is checked if "w" ends with "kļūdas", "šaubas"

or “izbrīnas”. If so, then, just as in cases when “w” does not contain any of the endings, “w” is returned without any other conversions. After manually checking “iii_first_g” (list of III conj. first group stemmed verbs), these three were the only ones found which in 3rd person present form with short “a” in ending matches a noun.

If the previous condition is not met, the ending is removed from “w” and within a loop it is checked if “w” is not in “iii_first_g” list. If it is, “w” is concatenated with the ending to which “a” is replaced with “ā” beforehand and then returned. If it is not, just like in part one, with “remove_prefix” and parameters “w” and false one prefix is removed from “w”. If the function did not return a prefix, word is concatenated with unmodified ending and returned. If the prefix was returned, the loop starts again by checking if “w” without a prefix is in “iii_first_g” list. If it is, all prefixes, “w” and the ending to which “a” is replaced with “ā” beforehand are concatenated and returned. If not, another prefix is removed and so on.

Function “change_suffix”

This function changes short to long vowels in suffixes and before them. It consists of four parts. Each part checks if conditions are met, stores stemmed “w” with the converted ending in the variable “st_w_mod”, while in “exc_nost” and “exc_st” – exception wordlists. These variables will be used at the end of this function.

The first part converts suffix “ib” to “īb” if the letter before is a consonant. Before this part, “w” is stemmed and stored in “st_w” variable. For the first and second part it is necessary for “st_w” to be longer than two letters, so “st_w” length is checked. If “st_w” contains two letters or less, “w” is returned unchanged; in other case the first part begins by checking if “st_w” ends with consonant followed by “ib”. If so “st_w” to which last two letters are changed to “īb” is stored in “st_w_mod” while in “exc_nost” and “exc_st” – “ib_nost” and “ib_st” wordlists. If this condition is not met, it is proceeded to the next part. Since there are no words in Latvian with vowel before the suffix “īb”, the consonant was added to the condition to better filter out unnecessary words.

For creation of “ib_st” wordlist, from “st_wordlist”, first prefixes from words were removed (at the end of this function, prefixes will be removed from the processed word as well); then it was checked if the word ends with consonant followed by “ib” and if so, the word was added in the “ib_st” list. The same method was used to make “ib_nost” list, except “nost_wordlist” words were used and each word was checked stemmed. This was done to ensure that if non-stemmable word will be converted (when actually it should not), after the

word is stemmed at the beginning of this function, this word can be within the exception list; thus, even with inflection or letters that are usually used in inflection, “ib” would not be converted to “īb”. For example, word “talibi” at the beginning of function “change_suffix” will be stem as “talib”; since this word ends with consonant and “ib”, it could be converted, but it will not since it was added in the “ib_nost” wordlist.

If the first part does not apply, the second part checks if “st_w” ends with “ak”. Such word prefix will later be converted to “āk”. If “st_w” ends with “ak”, variable “st_w_mod” stores “st_w” to which last two letters are changed to “āk”, while “exc_nost” and “exc_st” stores wordlists “ak_nost” and “ak_st”.

These two wordlists are made similarly to the first part’s wordlists – “ak_st” is made from “st_wordlist” list words from which prefixes are removed. If such words contain three or more letters and end with “ak”, they are added in the “ak_st” wordlist. For “ak_nost” wordlist “nost_wordlist” is used. From each word prefixes are removed. Then it is checked if they when stemmed contain three or more letters and end with “ak”. If so, such words non-stemmed, without prefixes are added in the wordlist.

If neither first part nor second part conditions are met, it is proceeded to the third part. For third and fourth part it is first checked if “st_w” contains four or more letters. If so, part three checks if “st_w” ends with consonant followed by short vowel and suffix “šan”. The short vowel before suffix will later be converted to a long vowel. In cases when this requirement is met, “st_w_mod” stores “st_w” value where the fourth letter from the end is converted to a long vowel (using a dictionary of short to long vowels), while variables “exc_nost” and “exc_st” stores exception wordlists “c_sv_san_st” and “c_sv_san_nost”.

Just as previous two exception wordlists, “c_sv_san_st” and “c_sv_san_nost” are made from “st_wordlist” and “nost_wordlist” words to which prefixes are removed. For “c_sv_san_st” wordlist, if word is longer than five letters, ends with consonant, short vowel and “šan”, the word is added in the list; while for “c_sv_san_nost”, if these conditions are met with stemmed words, the word is added in the “c_sv_san_nost” list.

Since such “c_sv_san_st” wordlist contains only 10 words, it was updated with verbs from Latviešu valodas ilgtspējas atbalsta partneri website from which nouns with “šan” were formed. Since no verbs in infinitive end with a consonant and a short vowel after removing ending “ties” or “t”, it was checked if verbs which lose a consonant when forming nouns (with 'šan') end with a consonant and a short vowel before this suffix. According to Latviešu valodas resursi cf. (Latviešu valodas resursi) verbs of 1st conj. present tense indicative form which stem

ends with “t”, “d”, “s” or “z” lose the last letter when forming present form with “st”; and when forming nouns from them with “šan” they are lost as well (e.g., “mešana”, “vešana”).

To find such words, all first conj. verbs were found that ends with “st”. This ending then was removed. If such words end with consonant followed by a short vowel, all prefixes from words are removed and “šan” is added at the end of the word. Finally, all of these words are added in the “c_sv_san_st” list.

Finally, “change_suffix” is proceeded with the fourth part, if none of the previous part conditions are met. Unlike the previous parts, this part will not prepare “st_w_mod” and the wordlists but will convert the word within itself. It converts suffix “taj” to “tāj” and a short vowel to a long vowel before the suffix if needed.

To do so, it is first checked if “st_w” ends with “taj” or “tāj” and if the letter before the ending is a vowel. If so, all prefixes from “st_w” are removed and stored in “st_w_nopref”. To make sure that this word is not an exception (e.g., “vidutājs” or city “Getaja”), it is checked if:

- “st_w_nopref” is in “sv_taj_st” or in “sv_tāj_st” wordlists;
- “w”, to which prefix is removed, is in “sv_taj_nost” or in “sv_tāj_nost” wordlists.

If either is true, “w” is returned unmodified.

To find out if only the suffix “taj” to “tāj” should be changed in the word, variable “st_w_nopref_tāj” is made. It consists of “st_w_nopref” to which the last three letters are replaced with “tāj”. Then it is checked if:

- “st_w_nopref_tāj” is in “sv_tāj_st” wordlist;
- “w”, to which prefixes are removed and letters of “st_w_nopref” are replaced with “st_w_nopref_tāj”, is in “sv_tāj_nost” wordlist.

If either is true, the suffix of “w” is modified and returned, that is, the function returns “w”, where the letters of “st_w” are replaced with “st_w” to which the last three are changed to “tāj”. “w” is converted in such way to modify only the suffix and not other “taj” occurrences in the word.

Since in Latvian literary language there are no words with long vowel and “taj” before an inflection, the last thing to check is if the vowel before the suffix needs to be converted. To do so, a variable “st_w_mod” is made with the value of “st_w” to which the fourth letter from the end is converted to a long vowel and the following letters are “tāj”. Then the function returns “w” to which “st_w” is replaced “st_w_mod”.

All four wordlists used in this part were created in similar way. For “sv_taj_st”:

- If “st_wordlist” words contain four or more letters;
- Fourth from the end is a short vowel and following are “taj”;
- The word without prefix is added in the “sv_taj_st” list;
- While if the last three are “tāj”, the word without prefix is added in the “sv_tāj_st” wordlist.

For “sv_taj_nost” and “sv_tāj_nost” lists, “nost_wordlist” words were used and the words were checked stemmed. That is, each word from “nost_wordlist” was checked if it when stemmed:

- Contain four or more letters,
- The fourth from the end is a short vowel.

If so, and the last three were “taj”, the word (non-stemmed), to which prefixes were removed, was added in the “sv_taj_nost” list; while if it ends “tāj” – the word, without prefixes, was added in the “sv_tāj_nost” list.

The first, second or third part created a “st_w_mod” and added the exception lists. If these variables contain any value, the conversion is completed by checking if “w”, to which prefixes are removed, is in the “exc_nost” and stemmed in “exc_st” wordlists. If it is in any of the lists, “w” unmodified is returned, while if it is not, the function returns “w” to which “st_w” letters are replaced with “st_w_mod”.

“o_w_dict” dictionary

After all previous conversions “w” in each function is rewritten with either modified or the same value. Then, finally, it comes to “o_w_dict” – a dictionary of common words (mostly adverbs) written in old Latvian and their conversion to modern Latvian. If “w” is in the dictionary, it is replaced with the converted word.

After that, function “edit_w” returns “w” back to function “split_and_edit_words” which then checks the correct word casing and replaces all occurrences of “w” in the text. After that, the text is returned to function “convert” which then returns it.

3.2. Testing

To test the accuracy of the tool, a small text corpus is collected and converted using the conversion tool and three AI chatbots – ChatGPT, Google Gemini and Microsoft Copilot. To measure conversion accuracy, all converted texts are compared with the text which is converted

manually by human. Since none of the previously mentioned AI tools are able to recognize old Latvian characters, the testing is carried out only on text conversion and not OCR part.

Since there are no other tools that would be able to convert old Latvian texts to modern Latvian, in order to compare the conversion tool three AI chatbots were used. According to Moosend (Dimitriou, 2024) and Tidio (Stefanowicz, 2024) in February/March of 2024 the best three AI chatbots are ChatGPT, Google Gemini and Microsoft Copilot. Assuming that most of the conversion tool users do not have access to paid versions of chatbots, for these systems free of charge versions were used.

First, the text corpus is prepared. It consists of 16 pages from newspaper "Latviešu avīzes" and 4 pages from magazine "Atpūta". From each decade, from 1820s till 1930s two pages were taken, except for decades from 1880 till 1910. In these four decades only one page was taken since the amount of text in them is more than twice as much as in pages from other decades. For the test set "Latviešu Avīzes" was chosen since this is the oldest newspaper published in Latvian which dates back to 1820s. But since it ceased to be issued in 1915, in decades of 1920s and 1930s one of the most popular magazines of the first half of the 20th century, "Atpūta", was used. From all 20 pages, the images were collected and the text from them was retrieved. Then the OCR errors were manually corrected by comparing the text with the image. The wrapped words in the text were concatenated back again to ensure that the text contains only whole words. In some pages there were words that were split between pages; such word parts were removed, just as advertisements which appeared in some "Atpūta" pages. As mentioned in the introduction, this is very time-consuming process, which together with manual text conversion tuck about the third of the whole allocated time.

With the recognized text, the conversion was made with the conversion tool and with all three AI tools. During conversion with AI tools it was found that in converted texts some punctuation marks, words and even sentences were missing and that the sentences were converted not word by word, but by only retaining their main idea. Since such conversion is too superficial and since it was not possible to make AI tools to convert the text retaining original style, words, their order, punctuation marks etc. (because after requesting that, the tools still kept converting sentences by retaining only the main idea), the text conversion with AI tools were performed differently.

The text was split in words and a set of unique words was created, just as in the conversion tool. Then to each chatbot it was asked to convert the words. For each request, the output was a list of unconverted words and their conversions. If unconverted word was in the

set of words, it was added in a dictionary, but if not, it was asked again to convert it. Finally, when all words from the set was converted and the dictionary contained all set words and their conversions, each word in the text was replaced using the dictionary retaining the original word case. This is done just as in the conversion tool.

For all three chatbots the request was following: “I have a list of words from Latvian periodicals issued from 1823 to 1936. Letters are already converted to Antiqua and letter "s" is denoted with "s" or "ſ" (unicode U+1E9C); "S" with "ꞩ" (unicode U+A7A9); "z" with "ſ" (unicode U+017F); "Z" with "S" and long vowels with a vowel followed by "h" or with various diacritic marks. But still, these words are in old orthography. Please, help me to convert them to modern Latvian. I want you to take each word and return them in both unconverted and converted form in a list of two columns and no additional information can be added. Here is the list of words:”. At the end of this request a list of words were added where each word was written in new line. The output was then entered in a Python dictionary.

At first, it was asked to converted 100 words, then another 100 words. Gemini was able to return only 83 and 81 words, Microsoft Copilot – 70 and 65, while ChatGPT – all 100 in both requests. In order to not miss any words chatbots forgets to output, a simple tool was created which returns 100 words from the set which are not jet added in the dictionary. Thus, even if some word is not returned by the first request, in later requests it will be.

For some words chatbot returned more than one word. After repeatedly asking to write only one word for each old Latvian word, it was not able to do so, therefore when adding words to the dictionary, the conversions with multiple words were added concatenated with dash “-”.

After creating three dictionaries using ChatGPT, Gemini and Copilot, three converted texts were created. To compare and measure the quality for all three converted texts and the text converted using the conversion tool, texts were compared with the ground truth – the text converted by human. The ground truth is made by the same method that was used for AI tools – by taking the set of words, manually converting them in modern Latvian, adding unconverted words and their conversions in a dictionary and then by using the dictionary each old Latvian word in the text was replaced with the modern Latvian word retaining the word case.

This method, just as the method used in the conversion tool, convert text word by word without taking the context into account. Without knowing the context, it is problematic to convert a text precisely. For example, it is impossible to determine the right word endings. For some words in the text there were no word endings (e.g., “Juhni”). Since such words can be converted in several ways, in the ground truth they were left without the ending. Even with the

ending, for some words it is impossible to know the correct word inflection. For instance, “domajat” can be converted either in “domājāt” or “domājat”. Without the context it is also impossible to know how to convert word stems. For instance, “dabbu” can be converted in “dabu” (nature) or “dabū” (get). In such cases words were converted with minimal possible conversions. The exception is in cases when from a feminine word a male surname can be formed, i.e., “Priedem” could be left unconverted (as it is a male surname in dative) but in ground truth it is converted to “Priedēm”. When creating the ground truth, outdated words that are no longer used were not replaced with words used today; and pronouns which in modern Latvian are written as two words, like “kautkas”, were not separated.

After manually converting all 7838 unique words, these words were replaced in the text of recognized periodical pages. With all five converted texts comparing can be made. First, all words from each text are retrieved. Each wordlist contains 18677 words, just as the original (unconverted) text in old Latvian. Then word by word, from wordlists of the conversion tool, ChatGPT, Gemini and Copilot each word was compared with the words from the ground truth. If a word match, the correct word count for the tool is increased by one. In either case, from both words character by character are compared. If they match, the correct character count is increased by one. If one word is shorted than the other, characters are compared until the shortest word has no letters to compare. To get the percentage of the words correctly converted, the count must be divided by 18677 (words in each wordlist and test set) and multiplied by 100, whereas to get the percentage of the characters correctly converted, the count must be divided by 98713 – the sum of letters in each word of the ground truth wordlist and multiplied by 100.

The results after comparing are following:

- The conversion tool was able to convert 87.01% (or 16250 from 18677) of words and 96.75% (or 95498 from 98713) of characters correctly;
- Google Gemini was able to convert 65.84% (or 12297 from 18677) of words and 78.91% (or 77899 from 98713) of characters correctly;
- ChatGPT was able to convert 52.22% (or 9754 from 18677) of words and 73.56% (or 72618 from 98713) of characters correctly;
- Microsoft Copilot was able to convert 51.65% (or 9647 from 18677) of words and 71.70% (or 70782 from 98713) of characters correctly.

From these results, it can be concluded that the conversion tool is able to convert words from the test set better than three the most popular AI chatbots: by 21.17% better than Google Gemini; by 34.79% better than ChatGPT and by 35.36% better than Microsoft Copilot, while

characters by 17.84% better than Google Gemini; by 23.19% better than ChatGPT and by 25.05% better than Microsoft Copilot.

From chatbots the best result was for Gemini. Even ChatGPT which has GPT version 3.5 performed better than Copilot which is based on the latest version. The result for chatbots can vary depending on how the request is formulated. But since the request was the same for all three chatbots so the results should be comparable.

Based on the test results, chatbots are able to convert text in good quality. However, if these texts are read, it is hard to grasp the main idea. When comparing chatbot converted texts with the text converted by the conversion tool, in a small qualitative analysis it can be concluded that in the texts converted by chatbots the incorrectly converted words are very different from the original word. On the contrast, the words converted incorrectly by the conversion tool do not interfere with reading. Typical mistakes made by the conversion tool are short vowels instead of long vowels in word endings and in suffixes, from which the most common was suffix “īg” (e.g., “laimigs”).

CONCLUSION

In this master thesis a tool was designed that converts printed texts written between the mid-18th till the mid-20th century to modern Latvian. The texts written in the early writing are significant cultural heritage. For modern readers to be able to read these texts, it is important to develop such a tool which would convert the text to modern Latvian. To do so, four tasks were set:

- Study the history of Latvian orthography;
- Designed a conversion tool;
- Prepare a dataset for testing the tool;
- Test and compare the tool with other conversion tools.

In the theoretical part Latvian orthography were examined in five periods:

1. The period of graphization and choice of language. In this period the orthography started to develop and the writing was very inconsistent.
2. Period of initial codification of norm periods. In this period emerged the kind of writing that formed the basis for the next two periods.
3. Period of norm stabilization. During this period the orthography solidified and remained with almost no change.
4. Period of partial norm changes and modernization. During this period, dissatisfaction with orthography led to extensive discussions and recommendations for improving orthography.
5. Period of current orthography. In this period the new orthography was implemented; together with few other changes, it constitutes modern orthography.

In the analytical and practical part, the development of the tool was described. Its methodology involved researching sources, compiling conversion rules, and implementing them into the tool. For some words to avoid certain conversions, exception lists were created using Tēzaurs dataset. This tool approximates text to modern Latvian, which means that some words written correctly may be converted to a different or even incorrect words. The conversion is performed in character and word level, therefore the context is not considered. The word order in a sentence, lexicon, capitalization of words and punctuation use is also not considered and is not changed. The intended language for the tool is mainly Latvian standard language while the texts for conversion should be printed between the mid-18th and mid-20th century.

The designed tool is a website which as an input takes both typed text and an image of a text; while the output is the converted text. For image recognition, JavaScript library Tesseract.js was used, while for Python environment embedding into the website – PyScript.

In the practical part, historical corpus data collection and processing was described. It consists of 20 pages of text retrieved from two periodicals published from 1820s till 1930s. From these pages text was retrieved and the containing OCR errors were manually corrected.

After all 18 677 words were reviewed, they were also converted to modern Latvian by human and then the conversion tool and by three AI chatbots – ChatGPT, Google Gemini and Microsoft Copilot. All five texts were converted without taking context into consideration. Finally, all four-machine converted texts were compared with the human converted text. To measure the accuracy of the conversion each correctly converted word and each correctly converted character was counted. After all four texts were compared to a text converted by a human, the results were following:

- The conversion tool was able to convert 87.01% of words and 96.75% of characters correctly;
- Google Gemini was able to convert 65.84% of words and 78.91% of characters correctly;
- ChatGPT was able to convert 52.22% of words and 73.56% of characters correctly;
- Microsoft Copilot was able to convert 51.65% of words and 71.70% of characters correctly.

From this it can be concluded that the conversion tool is able to convert test corpus better than the three most popular AI chatbots. Furthermore, after reading each text, the most common mistakes made by the conversion tool did not interfere with reading and they were short instead of long vowels in suffixes and word endings; while the mistakes made by AI chatbots changed the word meaning, thus did interfere with reading.

Bibliography

affinda. (2023, 07 28). *6 Top Open-Source OCR Tools: An Honest Review*. Retrieved from affinda: https://www.affinda.com/tech-ai/6-top-open-source-ocr-tools-an-honest-review

Andronova, E., Frīdenberga, A., Pretkalniņa, L., Siliņa-Piņķe, R., Skrūzmane, E., Trumpa, A., & Vanags, P. (2022). The conversion of early written Latvian texts into modern spelling: previous experience and automatization attempts. *Aktuālas problēmas literatūras un kultūras pētniecībā*, 346-362.

Andronova, E., Frīdenberga, A., Pretkalniņa, L., Siliņa-Piņķe, R., Skrūzmane, E., Trumpa, A., & Vanags, P. (2022, 10). *User-friendly Search Possibilities for Early Latvian Texts: Challenges Posed by Automatic Conversion*. Retrieved from ResearchGate: https://ceur-ws.org/Vol-3232/paper13.pdf

Augstkalns, A. (1930). *Rīgas latviešu biedrība zinātņu komisijas 20. rakstu krājums; Veclatviešu rakstu apskats.* Riga: Valters & Rapa.

Balearica. (2023, 02 13). *API.* Retrieved from GitHub: https://github.com/naptha/tesseract.js/blob/a17e42c2127c97be34a8e540fe2b7210b35c6203/docs/api.md

Balearica. (2023, 11 14). *Local Installation.* Retrieved from GitHub: https://github.com/naptha/tesseract.js/blob/a17e42c2127c97be34a8e540fe2b7210b35c6203/docs/local-installation.md

Bergmane, A., & Blinkena, A. (1986). *Latviešu rakstu attīstība, Latviešu literārās valodas vēstures pētījumi.* Riga: Zinātne.

Bērziņa-Baltiņa, V. (1994). *Latviešu valodas gramatika.* Rīga: Latvju grāmata.

Cambridge Dictionary. (n.d.). *orthography*. Retrieved from Cambridge Dictionary: https://dictionary.cambridge.org/dictionary/english/orthography

Deksne, D. (2021). *Priedēkļverbu semantika un funkcionalitāte latviešu valodā.* Retrieved from E-resource repository of the University of Latvia: https://dspace.lu.lv/dspace/bitstream/handle/7/57128/298-84802-Deksne._Daiga_dd10051.pdf?sequence=1

Dimitriou, M. (2024, 02 13). *15 Best AI Chatbots To Use In 2024 [Free & Paid]*. Retrieved from Moosend: https://moosend.com/blog/best-ai-chatbots

Druviete, I. (1990). *Kārlis Mīlenbahs.* Riga: Zinātne.

Dunsdorf, E. (1979). *Pirmās latviešu bībeles vēsture.* Minneapolis: Latviešu ev.-lut. baznīca Amerikā.

frakturs.lnb.lv. (2019). *Labot*. Retrieved from https://frakturs.lnb.lv: https://frakturs.lnb.lv/start

Jansone, I. (2008, 01 01). Ceļš uz rakstības tradīcijas maiņu: no gotiskā uz latīnisko. *Akadēmiskā Dzīve*, p. 63.

Latīņu un slāvu alfabēts latviešu rakstībā. (1997, 02 06). *Latvijas Vēstnesis*, p. 15.

Latviešu valodas ilgtspējas atbalsta partneri. (n.d.). *Darbības vārdu konjugācija*. Retrieved from Latviešu valodas ilgtspējas atbalsta partneri: https://www.lviap.lv/grammar/verbs/konjugacija/

Latviešu valodas resursi. (n.d.). *Vēsturiskais līdzskaņa zudums.* Retrieved from Latviešu valodas resursi: http://valoda.ailab.lv/latval/vidusskolai/fonetika/fon10.htm

Lingvistiskā karte. (n.d.). *Bībeles tulkošanas un izdošanas gaita (1682–1694)*. Retrieved from Lingvistiskā karte: http://www.lingvistiskakarte.lv/tag/bibeles_tulkosana_izdosana

literatura.lv. (2023, 02 22). *Juris Bārs*. Retrieved from literatura.lv: https://www.literatura.lv/personas/juris-bars

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/ 732509

National Library of Latvia. (2022, 12 15). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/1654367

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/731749

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/731999

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/731146

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/731037

National Library of Latvia. (2021, 05 06). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/1394369

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/731447

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/730786

National Library of Latvia. (2021, 05 06). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/1394994

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/730490

National Library of Latvia. (2021, 05 06). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/1395251

National Library of Latvia. (2018, 11 02). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/730043

National Library of Latvia. (2018, 11 01). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/728921

National Library of Latvia. (2018, 11 01). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/728825

National Library of Latvia. (2018, 11 01). *Latviešu Avīzes.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/727442

National Library of Latvia. (2017, 12 28). *Atpūta.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/238251

National Library of Latvia. (2018, 01 17). *Atpūta.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/351977

National Library of Latvia. (2018, 02 02). *Atpūta.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/420741

National Library of Latvia. (2018, 01 23). *Atpūta.* Retrieved from LNB Digitālā Bibliotēka: https://proc.dom.lndb.lv/DigitalObjects/DigitalObjectOverview/381774

Nefedova, M. (2023, 06). *Source image recommendations*. Retrieved from ABBYY Help Center: support.abbyy.com/hc/en-us/articles/360017326979

OCMA. (2023, 06 30). *Personvārdu datu bāze*. Retrieved from Personvārdu datu bāze: https://personvardi.pmlp.gov.lv/index.php?name=Johanna

OCMA. (2023, 06 30). *Personvārdu datu bāze*. Retrieved from Personvārdu datu bāze: https://personvardi.pmlp.gov.lv/index.php?name=Joana

OCMA. (2023, 06 30). *Personvārdu datu bāze*. Retrieved from Personvārdu datu bāze: https://personvardi.pmlp.gov.lv/index.php?name=Joanna

OCMA. (2023, 06 30). *Personvārdu datu bāze*. Retrieved from Personvārdu datu bāze: https://personvardi.pmlp.gov.lv/index.php?name=Johana

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

PyScript. (n.d.). *Architecture, Lifecycle & Interpreters*. Retrieved from PyScript: https://pyscript.github.io/docs/2023.11.1/user-guide/architecture/#micropython

PyScript. (n.d.). *Say Hello to PyScript*. Retrieved from PyScript: https://pyscript.net

Pretkalniņa, L., Paikens, P., Grūzītis, N., Rituma, L., & Spektors, A. (2012, 05). *Making Historical Latvian Texts More Intelligible to Contemporary Readers*. Retrieved from researchgate: https://www.researchgate.net/publication/230800163_Making_Historical_Latvian_Texts_More_Intelligible_to_Contemporary_Readers

Rainis, J. (n.d.). *The History of Blackletter Calligraphy*. Retrieved from Jake Rainis: https://jakerainis.com/blog/the-history-of-blackletter-calligraphy/

Rīgas Latviešu biedrības Zinātnības komisija. (1876). *Rakstu krājums 1. krājums.* Riga.

Rīgas Latviešu biedrības Zinātnības komisija. (1885). *Rakstu krājums 3. sējums.* Rīga.

Rīkojums par ortogrāfijas reformu. (1921, 02 01). *Izglītības Ministrijas Mēnešraksts*, p. 218.

Saulespurēns, V. (2022). Training Tesseract OCR Models for Latvian Fraktur Scripts Via Crowdsourcing. *ELAG 2022 conference.* Riga.

Spektors, A., Pretkalniņa, L., Grūzītis, N., Paikens, P., Rituma, L., Saulīte, B., . . . Darģis, R. (2023, 09). *Tēzaurs.lv, 2023, Autumn edition.* Retrieved from Clarin: http://hdl.handle.net/20.500.12574/92

Stefanowicz, B. (2024, 03 21). *22 Best AI Chatbots for 2024: ChatGPT & Alternatives*. Retrieved from Tidio: https://www.tidio.com/blog/ai-chatbot/

valoda.ailab.lv. (n.d.). *Literārās valodas jēdziens*. Retrieved from Latviešu valodas resursi: http://valoda.ailab.lv/latval/vidusskolai/literval/lit1.htm

valoda.ailab.lv. (n.d.). *Veclatviešu rakstības vidējais posms, 1631. - 1739. gads*. Retrieved from Latviešu valodas resursi: http://valoda.ailab.lv/latval/vidusskolai/literval/lit13.htm

Vanags, P. (2019). Latviešu valoda pirms Latvijas valsts. *Valsts valodas komisijas raksti, 10*(Valoda un valsts), 55-86.

Vanags, P. (2023. gada 21. 02). *latviešu valoda*. Ielādēts no Nacionālā Enciklopedija: https://enciklopedija.lv/skirklis/9891-latviešu-valoda

Vilcāne, A. (2018). *Latvia and Latvians : collection of scholarly articles. Volume II [2]*. Riga: Latvijas Zinātņu akadēmija.