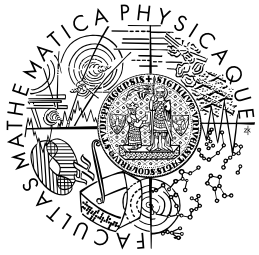


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Kateřina Nová

Analysis and visualization of OCR output

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis: doc. Mgr. Barbora Vidová Hladká, Ph.D.

Study programme: Computer Science

Study branch: General Computer Science

Prague 2022

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank you my supervisor, doc. Mgr. Barbora Vidová Hladká, Ph.D., for patient guidance and good advice.

Title: Analysis and visualization of OCR output

Author: Kateřina Nová

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. Mgr. Barbora Vidová Hladká, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Optical Character Recognition (OCR) is a process of converting text from images to a machine-readable text. We run three OCR systems (Tesseract, Ocrad and GOCR) on an original multilingual OCR dataset and perform statistical and linguistic analysis of the results in order to compare the tested systems and investigate typical OCR errors.

Keywords: Optical Character Recognition, golden data set, statistical analysis, Natural Language Processing

Contents

Introduction

Throughout human history, written text (or any other form of visualized speech) turned out to be an extremely successful and efficient tool to exchange, preserve and process information. In our times, sometimes called the Digital Age, its role is being more and more taken by digital or electronic text. Hand in hand with that, the need and importance of converting between these two types of text continues to grow. While all sorts of printers and displays make one way easy, the other one was and still is a major challenge of recently very dynamically developing field called *computer vision*.

Optical Character Recognition

For reasonably short texts the task of text digitizing can be done manually and it is in general still the most reliable method. On the other hand, this is extremely inefficient for larger data and another tool is needed. The automatic process of recognizing text in an image (which is mostly the only available electronic format of not yet digitized text) is called *Optical Character Recognition* (OCR). Its output is machine-readable text which allows for further processing. It is used in many real-life problems like handwritten text recognition, filled forms processing or text detection in photos. Here we focus on the recognition of printed plain text with not overly complicated formatting.

History and current methods

If we allow for slightly broader definitions of its task, OCR has quite long and rich history dating back to the 19$^{\text{th}}$ century and inventions of the retina scanner and Nipkow disk pioneering the path of scanning and automatic processing of images. During the 20$^{\text{th}}$ century the technology shifted from highly specialized tasks (text-to-telegraph conversion, reading devices for blind people, etc.) to greater generality and versatility. The general development of computing and communication technologies naturally influenced the field immensely allowing for unifying some of the tasks and employing powerful software and hardware.

Nowadays, OCR processing starts with the segmentation of images into lines, words or characters (cf. [6]). The next step usually consists of the individual characters recognition. One can classify the methods for doing so in a simplifying manner as

- comparison of the pixel matrix of an appropriate rectangle in the image with saved patterns,
- looking at the geometry and topology of the character, trying to decompose the character image in simpler shapes,
- methods based on neural networks.

More detailed version of this classification and the respective methods can be found in survey paper [17]. However, there is no general agreement on such a methods classification and even some related terms are used for different meanings by some authors. More about both the historical background and some modern features of OCR can be found e. g. in paper [3].

Goals and structure of the project

Besides relating our project briefly to other works in Chapter 1, the main goals of this thesis are building an original dataset for testing OCR systems, producing transcriptions using several selected ones, performing statistical and linguistic analysis of the output and summarizing and interpreting the results. The original plan (according to the thesis assignment) contained one more goal, namely to create a tool for visualizing the errors made the OCR process including their positions in the scanned image. Due to reasons related to our dataset (see 5.6 for details), we did not accomplish this goal.

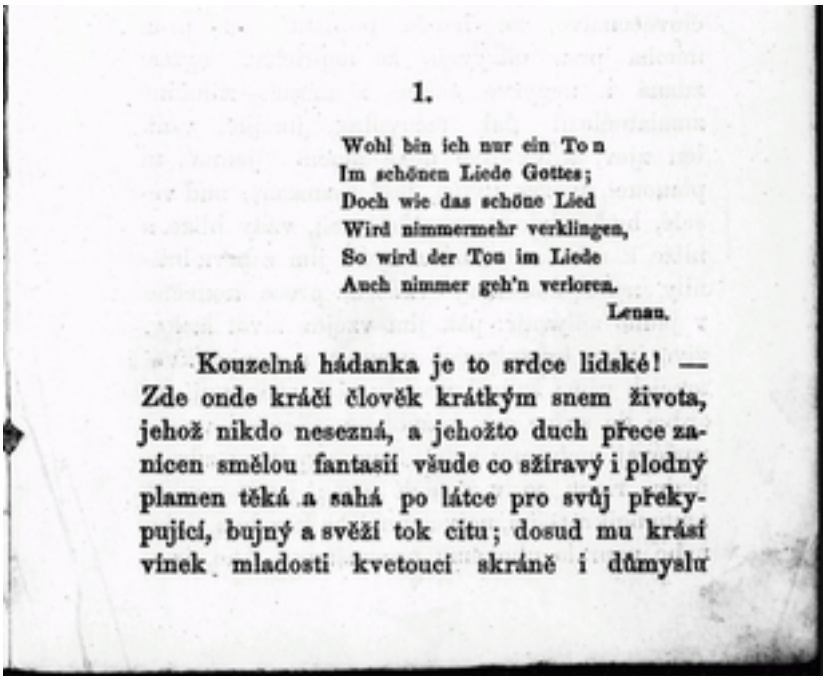
The dataset is decribed in detail in Chapter 2. It contains part of an existing OCR data collection B-MOD and several novels from the ELTeC project. The latter database is not meant for OCR processing and therefore the challenge here was adjusting the data for that. We can see a sample image from ELTeC in Figure 1 below.

Chapter 3 describes the used tools, mainly the three open-source OCR systems *Tesseract*, *GOCR* and *OCRad*. It also mentiones *UDPipe* – a program used for tokenization and further linguistic processing.

Chapter 4 then introduces the evaluation measures, particular statistical and linguistically analytical benchmarks used to quantify and compare the rate of success of the systems tested on our dataset.

Chapter 5 is devoted to the testing itself. The evaluation of its results uses original scripts which are presented as well.

Detailed results of the experiments are attached in Appendix A. Appendix B describes the content and structure of the electronic attachment consisting of all the data used for the experiments.


1.

Wohl bin ich nur ein Ton
Im schönen Liede Gottes;
Doch wie das schöne Lied
Wird nimmermehr verklingen,
So wird der Ton im Liede
Auch nimmer geh'n verloren.

Lenau.

Kouzelná hádanka je to srdce lidské! — Zde onde kráčí člověk krátkým snem života, jehož nikdo nesezná, a jehožto duch přece zanícen smělou fantasií všude co sžíravý i plodný plamen těká a sahá po látce pro svůj překypující, bujný a svěží tok citu; dosud mu krásí vínek mladosti kvetoucí skráně i důmyslu

Figure 1: Image for OCR

1. Related work

The report [12] formulates a list of tasks and recommendations for researchers and institutions involved in developing and applying OCR for digitizing paper manuscripts from archives, libraries, etc. The authors argue that following this suggested agenda would greatly influence the efficiency of this laudable effort. In this project, we partially address some of the points of that list.

The comparative study [5] contains an overview of a number of both proprietary and open-source OCR systems and a basic summary of results from other studies which compare the systems. Although they do not always employ the same evaluation measures as we do, the obtained success rate agrees in general with our results.

The thesis [16] compares all the systems we use and two other proprietary systems (FineReader and Cuneiform). The author compares them using basic character and word accuracy criteria. More interestingly from our point of view, this work studies the effect of image quality on OCR by adding several different noises and degrading the original images in ways that occur in practical applications.

A tool [1] computes statistics (both on word and character levels) of the difference between two texts. Besides statistics it also provides a graphical interface where users can see both compared texts next to each other with highlighted mutual differences. Hence, it can be useful for manual error checking.

Detailed study [10] compares OCR errors with human misspellings. It is focused on local changes of one word and suggests how it can be used in OCR post-processing. Following characteristics of OCR errors are examine: type and number of edit operations (substitution, insertions and deletions of characters), length of the word, the position of an error in a word, creation of existing or non-existing word and word boundary.

An example of creating an OCR dataset is contained in the study [7]. They used digitized historical Finnish newspapers and manually corrected the OCR output of the propriety system ABBYY FineReader 11. These data are used for testing the pre-trained Tesseract model.

In the experiments we investigate difference between POS tagged golden texts and OCR outputs (cf. Section 5.5). Paper [2] introduces a post-correction model which should improve the POS tagging and annotating with name entities on OCR output of German historical documents.

2. OCRData

For experimenting with OCR tools, one needs images containing text and the correct digital transcriptions of these texts. All the experiments described in this thesis use an original dataset called *OCRData*. Producing it was a substantial part of the project and it is provided as an attachment to the thesis (see Appendices A and B).

A smaller part (see Table 2.2) of the dataset comes from the *Brno Mobile OCR Dataset* (B-MOD) [8]. It is an established dataset adapted for OCR experiments. However, it contains solely English texts divided into lines and restricting our experiments to such a specific type of data would not reveal enough about the functionality of the tested systems.

The majority of OCRData (see Table 2.1) is made up of scanned books from the *European Literary Text Collection* (ELTeC). Unlike B-MOD, ELTeC is not an OCR dataset and therefore it was necessary to extract all the data we need and adjust them appropriately for our experiments.

In its final form, OCRData contains scanned images of all the used texts in both TIFF and PNM formats (we call them *OCR images*) and their machine-readable transcriptions as plain text (we call these *golden texts*). Tesseract, the main OCR system we use (see Chapter 3 for details), supports multi-page TIFF files (which are convenient to handle) but the other systems do not. Therefore we work also with OCR images in one-page PNM format.

2.1 ELTeC subset

The European Literary Text Collection[1] (ELTeC) is a part of Action Distant Reading for European Literary History – a broader project managed by EU via its funding organization European Cooperation in Science and Technology. It is a comprehensive collection of around 2500 full-text novels in more than 10 different languages published between 1840 and 1920. For more details see [11].

The digitized versions of the texts in the collection are in TEI encoding (cf. [15]), a representation of digital texts based on XML format. Besides plain text, it can contain e. g. linguistic features such as POS tags and lemmas. For marking these files, ELTeC uses three levels depending on the amount of the metadata available about the given novel:

- level 0 – basic TEI Encoding[2]
- level 1 – richer TEI Encoding[3]
- level 2 – TEI Encoding with tokenization and linguistic annotation.[4]

All novels which we use are from level 1. Files in that level contain information about the book such as title, author or the link to the source stated in the header and the main text in the body. We can see an example of a shortened header in Figure 2.1.

[1] `https://github.com/COST-ELTeC/ELTeC`
[2] `https://distantreading.github.io/Schema/eltec-0.html`
[3] `https://distantreading.github.io/Schema/eltec-1.html`
[4] `https://distantreading.github.io/Schema/eltec-2.html`

```
    <?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0" xml:id="CS0018" xml:lang="cs">
<teiHeader>
<fileDesc>
        <titleStmt>
                <title>Pravý přítel</title>
                <title xml:lang="en">The true friend</title>
                <author>Švestka, Josef (1816-1882)</author>
                <respStmt>
                        <resp>editor</resp>
                        <name>Institute of the Czech National Corpus:
                            diachronic section</name>
                </respStmt>
        </titleStmt>
        <extent>
                <measure unit="words">10665</measure>
        </extent>

        <sourceDesc>
        <bibl>
                <ref target="http://kramerius.nkp.cz/kramerius/
                    MShowMonograph.do?id=24911"></ref>
                <publisher>National Library of the Czech Republic</
                    publisher>
                <relatedItem type="printSource">
        <bibl>
                <title>Pravý přítel. Povídka ze života pro ušlechtění
                    srdce mládeže. Pochlebník. Krátká povídka k poučení a
                    k výstraze mládeže. Nezištné přátelství. Vyprávěnka
                    hodna následování.</title>
                <publisher>Jos. Mikuláš</publisher>
                <pubPlace>V Praze</pubPlace>
                <date>1881</date>
        </bibl>
                </relatedItem>
        </bibl>
        </sourceDesc>
</fileDesc>
<encodingDesc n="eltec-1">
        <p></p>
</encodingDesc>
<profileDesc>
        <langUsage><language ident="CS">Czech</language></langUsage>
</profileDesc>
</teiHeader>
```

Figure 2.1: TEI header example of novel CS0018

The sources vary from already digitized books (for example from Wikisource[5] or Project Gutenberg[6]) to physical paper books and scanned images of the respective books, whereas only the last type is suitable for us. In our experiments, we use the source images (appropriately reformatted, cf. Section 2.3) as OCR images and the main texts as golden texts.

[5] https://wikisource.org/wiki/Main_Page

[6] https://www.gutenberg.org/

Moreover, even for the scanned images, each language collection uses a different source (or multiple sources), usually a local library website. For instance, the book from the example 2.1 was downloaded from such a website[7]. This causes difficulties for building a large dataset in this way as the downloading process cannot be easily done automatically and some sources are not even publicly available (without registration or at all). However, most of the files can be downloaded directly and therefore we use only them.

In its final form, OCRData contains 64 twenty-page-long sections of novels in Czech, Portuguese, Slovenian and French. These languages were chosen because their novels' sources are mainly freely accessible scanned images. We take only the first 20 pages of each book because some novels are quite long and evaluating would be problematic due to technical limitations.

The number of novels and tokens for each language included in OCRData can be found in Table 2.1. Titles and authors of all the novels are listed in Appendix A.1.

Language	Number of novels	Number of tokens
Czech	19	71 850
Portuguese	16	70 179
Slovenian	18	218 216
French	10	44 998

Table 2.1: Numbers of novels from ELTeC in OCRData

2.2 B-MOD subset

Brno Mobile OCR Dataset (B-MOD), part of the PERO project[8], is a collection of photographs of scientific papers captured by different mobile devices. It contains about 20 000 photographs and more than 500 000 text lines with precise transcriptions (Figure 2.2). As the mentioned papers often come with nontrivially structured text while our evaluation scripts are suited primarily for plain text as input, we include only the cropped lines for the OCRData.

There are three levels of image quality – easy, medium, and hard – named according to the expected difficulty of their OCR processing. However, the quality of some images even in the easy part is much worse than of the images from ELTeC as we can see in an example in Figures 2.2 and 2.3.

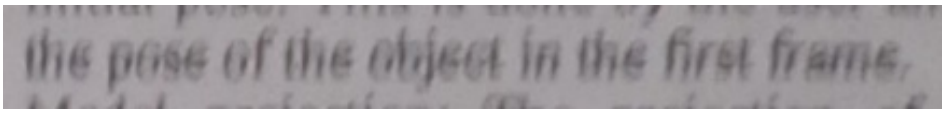


Figure 2.2: Image from B-MOD of easy quality

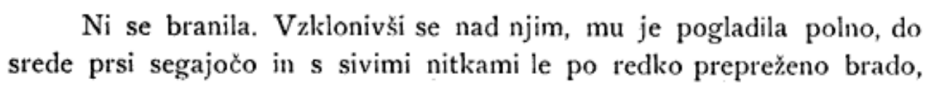


Figure 2.3: Image from ELTeC

[7] http://kramerius.nkp.cz/kramerius/MShowMonograph.do?id=24911
[8] https://pero.fit.vutbr.cz/

Difficulty set	Number of lines	Number of tokens
english.easy	1 000	11 108
english.medium	1 000	12 010
english.hard	1 000	10 118

Table 2.2: Size of data from B-MOD in OCRData

B-MOD divides the data into training, validation and testing subsets. For OCR-Data, we use the first 1 000 lines from the validation subset per each quality level (3 000 lines in total), the number of tokens can be found in Table 2.2. Originally, B-MOD contains a separate JPG file for every line (Figure 2.3). We merge all these files of each level into one multi-page TIFF file for consistency with the data from ELTeC.

For more details about B-MOD, see [8].

2.3 Pre-processing of ELTeC data

Different digital processing methods need different data sources. Recall that for OCR systems evaluation we need to provide text images and their exact golden transcriptions. Before recognising individual characters, OCR systems have to do some layout analysis (or segmentation) of the given image and this can be (in some systems) included in the output as well. Hence, for a detailed evaluation, the golden texts should in the best case contain not only words but also their positions. However, from the perspective of the ELTeC mission, only the content of the book matters and therefore the digitized texts do not have to look exactly the same as the scanned images or even not contain some information present in the scanned images.

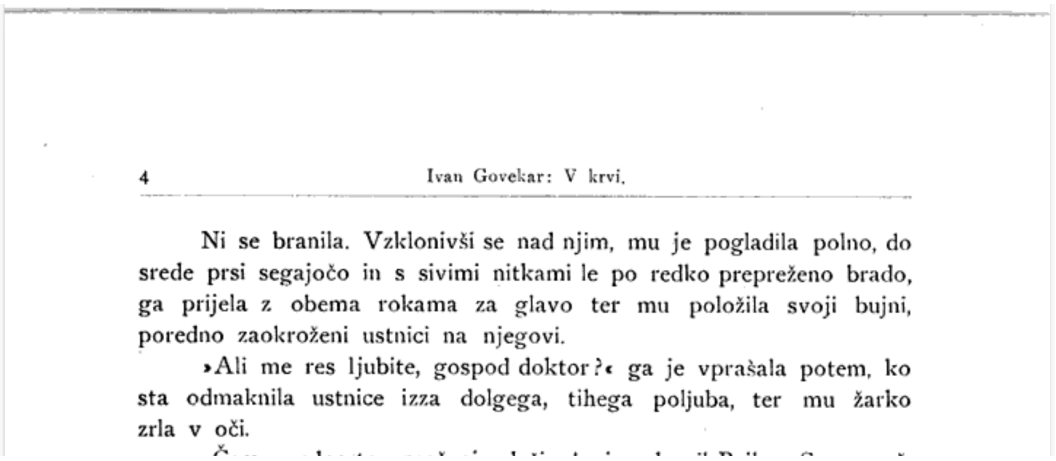
This section describes these differences and how we manage to remove them. Besides that, we need to transfer the original files to suitable formats for OCR processing and evaluation. The overview of the whole pipeline is depicted in Figure 2.6.

Paper books are not just the main texts, they also have covers and preface pages which contain some text as well. Generally, we can find here a title, name of the author(s) and publishing information. Most of the books are scanned including the introductory pages but some of golden texts only begin after them. That is one type of the image-transcription differences we usually encounter.

Another extra information in a scanned book occurs in headers as page numbers and sometimes the book title, author or chapter label. In Figure 2.4, we can see an example of a header from the Slovenian part of the OCRData.

The last major source of the image-transcription differences are words divided at the end of a line which occurs in scanned images (and therefore also in the OCR outputs) but in golden texts (see Figure 2.5). One solution would be to manually edit the golden texts according to the images and split the corresponding words. However, this would be quite tedious. For this reason, we use a different approach and merge all the divided words in OCR outputs before doing any experiments and evaluations.

After all these adjustments, golden data and OCR image should in theory have precisely the same text content but there is still the question about the structure of the data. In other words, it is reasonable to have some basic text unit and compare then the OCR input and output unit by unit. If we wish to do this on higher level, i. e. to divide the given book into several smaller units, we have basically two options.



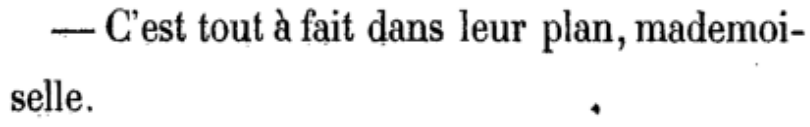
4 Ivan Govekar: V krvi.

Ni se branila. Vzklonivši se nad njim, mu je pogladila polno, do srede prsi segajočo in s sivimi nitkami le po redko prepreženo brado, ga prijela z obema rokama za glavo ter mu položila svoji bujni, poredno zaokroženi ustnici na njegovi.

»Ali me res ljubite, gospod doktor?« ga je vprašala potem, ko sta odmaknila ustnice izza dolgega, tihega poljuba, ter mu žarko zrla v oči.

Figure 2.4: Header in golden image of novel SLV00048

The first one presents a logical (or semantic) approach and uses division into chapters, paragraphs or sentences. Alternatively, we can divide the documents more visually into pages and lines. ELTeC uses only the first approach and hence it does not preserve what pages look like (with one one exception, namely the golden texts in Czech data are divided into pages according to the scanned books). On the other hand, a typical OCR output consists of lines corresponding to those from the source image and do not work with sentences at all. As neither of these approaches would not work for our entire dataset, we need to go to lower level and consider the texts just as a stream of words and forget about all other structure.



— C'est tout à fait dans leur plan, mademoi-

selle.

— C' est tout à fait dans leur plan , mademoiselle .

Figure 2.5: Divided word at the end of line and the corresponding transcription of golden data

2.3.1 Golden texts pre-processing

Recall that every text from ELTeC is in XML format. Its body contains a novel text divided into chapters and paragraphs which we do not use, therefore we extract only plain text out of it. From the header we need only the source link to the scanned book for downloading the corresponding OCR image.

The next step is to select only the text corresponding to the first 20 pages of the scanned novel. Except for the Czech collection, no XML files are structured into pages, hence this has to be done manually – we simply look at the end of the 20^{th} page in the image file and find the corresponding place in the text file.

2.3.2 OCR images and OCR output

As mentioned above, the files with scanned novels contain whole books including the covers and introductory pages which are not usable for OCR processing because they contain no text or have no corresponding transcription in the golden texts. Therefore all these pages are manually removed to achieve the highest possible similarity of the images and the golden texts. However, if there are transcription of at least some of these introductory pages in golden texts, we keep them as they give us more varied data than only simple text blocks of plain text.

All files with scanned images are in PDF format that is not supported by any of the used systems. Therefore we convert them to multi-page TIFF files. Then, as already mentioned, we take 20 pages from the first relevant page.

Similarly as for the introductory pages, the headers (containing novel titles, numbers of chapters, etc.) have no corresponding transcription and have to be removed. Therefore the next step consists of cropping the headers. The resulting TIFF file is then converted to 20 PNM images for processing by Ocrad and GOCR.

The last step is merging the divided words in the OCR output. We detect the last string of non-white-space characters at the end of the line ending with a hyphen (`'-'`) and merge it with the first string of the next line. As opposed to splitting words in golden texts according to the scanned images, it can be easily done automatically. We must keep in mind that we work with the OCR output here and hence the merging can be affected by the errors of the OCR process. Nevertheless, we make bad merging only in places where the OCR system made an error.

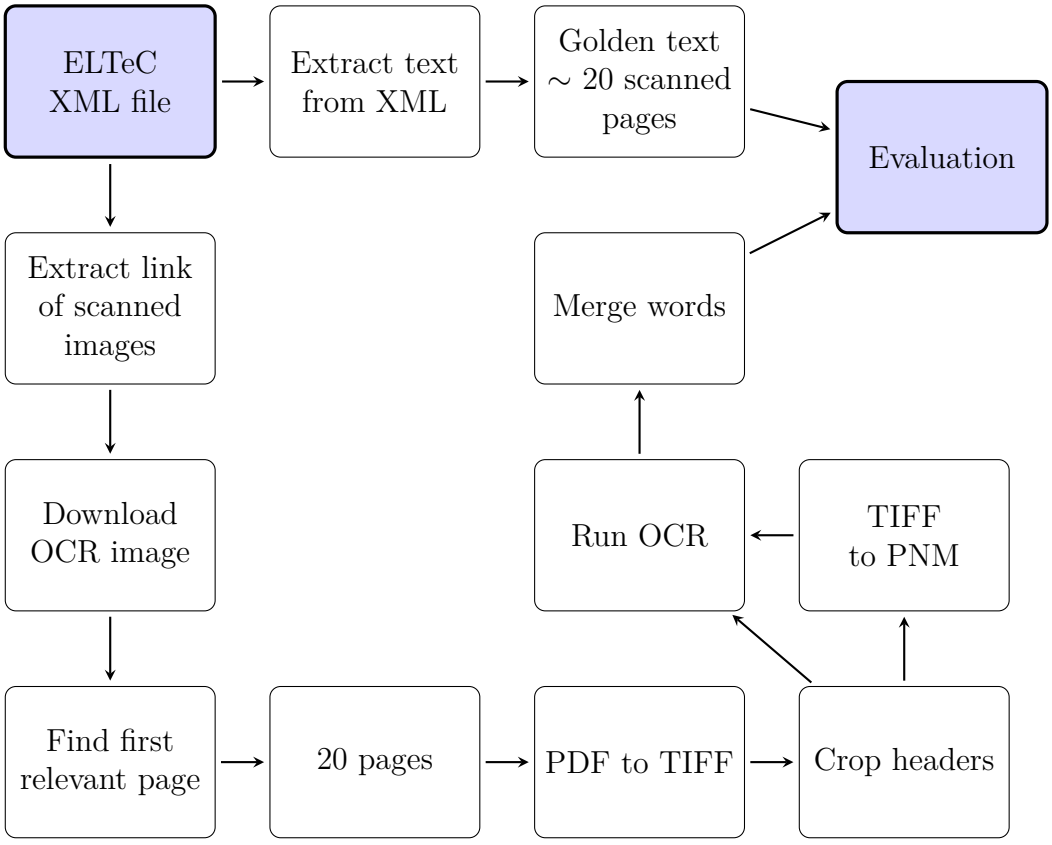


Figure 2.6: Pre-processing pipeline

3. Tools

This chapter describes all the compared OCR systems (Tesseract, GNU Ocrad and GOCR). There are many others available, but we decided to choose only open-source and command-line tools as a non-graphical interface allows for easier automatization of the evaluation. The last described tool is UDPipe which is used for *tokenization* and *POS tagging* which is done both on golden and OCR texts before the evaluation.

3.1 Tesseract

Tesseract is an open-source command-line program for OCR (see [13]). It was originally developed at Hewlett-Packard Laboratories between 1985 and 1994. The first open-source version was released in 2006. In the beginning, it supported only English. Five more languages (French, Italian, German, Spanish, Dutch) were added to its second version released in 2007 and now it is trained for more than 100 languages.

3.1.1 Setting and using

We use Tesseract version 4.1.1 equipped with a new engine based on Long Short-Term Memory ([4]) (LSTM). It can also be switched back to the old Legacy mode, though. The Legacy mode detects lines, splits them into words and characters and then recognizes every character according to the features based on the character shape.

One can specify the OCR Engine Mode by the *oem* option and use the old engine, LSTM neural network or a combination of both. For all experiments we compare all these three modes:

```
--oem N
    Specify OCR Engine mode. The options for N are:
    0 = Original Tesseract only.
    1 = Neural nets LSTM only.
    2 = Tesseract + LSTM.
```

When running Tesseract one has to specify a language (English by default). For multi-language data one can also use more languages. As every file from OCRData is in one language, we do not need this functionality.

Tesseract provides training data for 123 languages but it is also possible to train a model with user's data. We use only the pre-trained models for Czech, Slovenian, French, Portuguese and English.

Another option is changing the default *page segmentation mode* (psm, see Figure 3.1) which can be used when we know how the text is placed on the image. Some books (in ELTeC) have two text columns on one page and using this information should in principle give better results. However, Tesseract provides only single-column options (psm 4, 5, 6). Therefore we use only the default option which works fine even in the two-column cases.

```
—psm N
Set Tesseract to only run a subset of layout analysis
and assume a certain form of image. The options for N are:

    0 = Orientation and script detection (OSD) only.
    1 = Automatic page segmentation with OSD.
    2 = Automatic page segmentation, but no OSD, or OCR.
    (not implemented)
    3 = Fully automatic page segmentation, but no OSD.
    (Default)
    4 = Assume a single column of text of variable sizes.
    5 = Assume a single uniform block of vertically
    aligned text.
    6 = Assume a single uniform block of text.
    7 = Treat the image as a single text line.
    8 = Treat the image as a single word.
    9 = Treat the image as a single word in a circle.
    10 = Treat the image as a single character.
    11 = Sparse text. Find as much text as possible in
    no particular order.
    12 = Sparse text with OSD.
    13 = Raw line. Treat the image as a single text line,
    bypassing hacks that are Tesseract-specific.
```

Figure 3.1: Description of psm modes from manual page

The data from B-MOD consist of one-line images which is an available option in Tesseract (psm 7, 13). In this case, we compare default segmentation with the modes psm 7 and psm 13. The results can be found in Section 5.3.

3.2 GOCR

GOCR is open source program for optical character recognition.[1] It was developed by Joerg Schulenbur mostly between the years 2000 and 2010, the last version was released in 2018.

Its algorithm starts with line segmentation as well. Then it detects clusters of pixels which should corresponds to individual characters. For the recognition of these clusters, two engines are used. Original rule-based engine which describes shape of characters and a database engine which compares pixel clusters with images of characters and tries to find the one with the smallest distance. The database can be extended by the user to include e. g. language-specific characters.

The default setting is to take images in PNM format and write text on stdout. It is also possible to use stdin stream instead of an input file or with help of some external programs one can use more image formats (pnm.gz, pnm.bz2, png, jpg, jpeg, tiff, gif, bmp, ps and eps). However, only single page files are accepted and therefore we can not use our multi-pages TIFF files and we use PNM files instead.

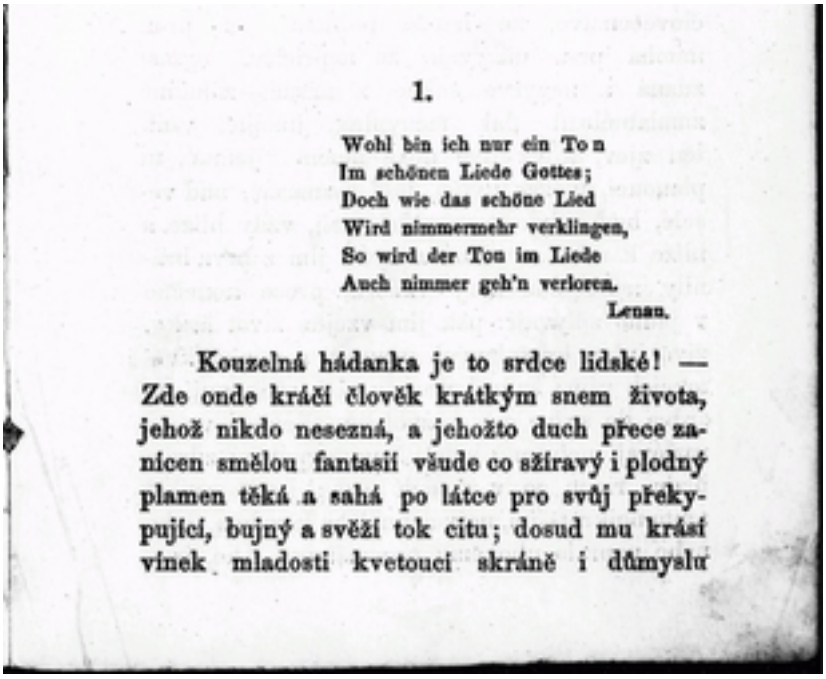
[1] `http://jocr.sourceforge.net/`

During the recognition, GOCR estimates for every character the probability that it has been correctly recognized. When this value is below an (adjustable) treshold, the character is considered unrecognized and GOCR writes special character on the output. The default option is '_' but it can be also changed by the user. We keep default setting for both these settings.

The user's next option is setting a filter function which allows only a limited set of characters on output:

```
–C string
      only recognise characters from string, this is
      a filter function in cases where the interest is
      only to a part of the character alphabet, you can
      use 0–9 or a–z to specify ranges, use — to detect
      the minus sign
```

This can be useful if a dataset is somehow limited and we know it contains e. g. only numbers. Unfortunately, this is not our case, our data are quite varied. The only way for us to make use of this functionality would be creating a list of characters depending on the language. However, it can be quite tricky and needs some manual control of the texts to ensure the list includes all the characters. For example, there are some quotes in German in this Czech book (see Figure 3.2).



1.

Wohl bin ich nur ein Ton
Im schönen Liede Gottes;
Doch wie das schöne Lied
Wird nimmermehr verklingen,
So wird der Ton im Liede
Auch nimmer geh'n verloren.

Lenau.

Kouzelná hádanka je to srdce lidské! — Zde onde kráčí člověk krátkým snem života, jehož nikdo neseznná, a jehožto duch přece zanícen smělou fantasií všude co sžíravý i plodný plamen těká a sahá po látce pro svůj překypující, bujný a svěží tok citu; dosud mu krásí vínek mladosti kvetoucí skráně i důmyslu

Figure 3.2: German quotes in Czech book

3.3 GNU Ocrad

Ocrad is console program and library for OCR.[2] It can do some layout analysis to find text locations and remove pictures and frames. Then it detects characters on each line and finds the best match to every character. Afterwards, some post-processing can be

[2] https://www.gnu.org/software/ocrad/

done, but only on character level and therefore there are no language-specific modes as in Tesseract.

It can read only PNM images and produces text in byte or UTF-8 format. The default setting for output format is byte but we used UTF-8 for easier post-processing.

Similarly as GOCR, Ocrad also has a filter function (see Figure 3.3 below). There are several built-in filters and one can also use a customized set of characters. The build-in filters are certainly too restrictive for our dataset and as mentioned in Section 3.2, creating our own filter is tricky as well.

```
—filter=letters
Forces every character that resembles a letter to be
   recognized as a letter. Other characters will be output
   without change.
—filter=letters_only
Same as '--filter=letters', but other characters will be
   discarded.
—filter=numbers
Forces every character that resembles a number to be
   recognized as a number. Other characters will be output
   without change.
—filter=numbers_only
Same as '--filter=numbers' but other characters will be
   discarded.
—filter=same_height
Discards any character (or noise) whose height differs in more
    than 10 percent from the median height of the characters
   in the line.
—filter=text_block
Discards any character (or noise) outside of a rectangular
   block of text lines.
—filter=upper_num
Forces every character that resembles a uppercase letter or a
   number to be recognized as such. Other characters will be
   output without change.
—filter=upper_num_mark
Same as '--filter=upper_num', but other characters will be
   marked as unrecognized.
—filter=upper_num_only
Same as '--filter=upper_num', but other characters will be
   discarded.
```

Figure 3.3: Description of filter function from Ocrad manual

3.4 UDPipe

UDPipe is a tool for tokenization, tagging, lemmatization and syntactic analysis (see [14]). We use tokenization and tagging.

Tokenization is a process of parsing text into a sequence of tokens. A token can be either a word or punctuation. Tokenization is an essential step as all the used evaluation measures work only with token post-processing, not with raw text.

The *POS tagging* is a process of annotating words in sentences by *parts of speech* tags. Words with the same POS tags play similar roles in sentences and they can be sometimes substitutes to each other without loosing grammatical correctness. We use only basic set of tags such as noun, verb, adjective, etc.

UDPipe can be used directly from command line (Linux, Windows, OS X), as a library for multiple languages (C++, Python, Perl, Java, C#) and as a web service using LINDAT/CLARIN-CZ infrastructure[3].

3.4.1 Usage

UDPipe provides pre-trained models for around 50 languages.

We did the tokenization from Linux command line. All our evaluation metrics work only with a sequence of tokens, no sentences or paragraphs, therefore we used vertical output format (one token per line). The input format is plain text.

The POS tagger needs either already done tokenized input or it is possible to use plain text and perform the tokenization together with tagging, we use the second option. We could also extract tokens from CoNLL-U files (described below) similarly as we do it for tags and do the tokenization only once. However, our way allows to do experiments more independently.

The default output format for tagger is CoNLL-U v2. It divides a text into paragraphs, sentences and tokens. Each token in a sentence have ten morphological and syntactic features. The list of all features and their description we can see in Table 3.1.

The output of the UDPipe tagger contains only the first six features (ID, FORM, LEMMA, UPOS, XPOS, FEATS). The Figure 3.4 shows a sample sentence from the dataset and its tokenization and tagging in CoNLL-U format. However, for our purpose we need only UPOS for each token and, as mentioned above, only as a sequence of tokens and tags. The POS tag analysis script takes the CoNLLU format and extracts the POS tags themselves.

[3] `http://lindat.mff.cuni.cz/services/udpipe/`

Feature	Description
ID	Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes (decimal numbers can be lower than 1 but must be greater than 0).
FORM	Word form or punctuation symbol.
LEMMA	Lemma or stem of word form.
UPOS	Universal part-of-speech tag.
XPOS	Language-specific part-of-speech tag; underscore if not available.
FEATS	List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.
HEAD	Head of the current word, which is either a value of ID or zero (0).
DEPREL	Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.
DEPS	Enhanced dependency graph in the form of a list of head-deprel pairs.
MISC	Any other annotation.

Table 3.1: Features in CoNLLU format

```
    # sent_id = 6
# text = Que os corações humanos tanto Obriga,
1       Que     que     SCONJ   _       _       _       _       _
        _
2       os      o       DET     _       Definite=Def|Gender=Masc|
    Number=Plur|PronType=Art        _       _       _       _
3       corações        corações        NOUN    _       Gender=Masc
    |Number=Plur _       _       _       _
4       humanos humano  ADJ     _       Gender=Masc|Number=Plur _
        _       _       _
5       tanto   tanto   ADV     _       _       _       _       _
        _
6       Obriga  obriga  PROPN   _       Gender=Masc|Number=Sing _
        _       _       SpaceAfter=No
7       ,       ,       PUNCT   _       _       _       _       _
        SpacesAfter=\n\n
```

Figure 3.4: UDPipe example of tokenized and tagged sentence in CoNLL-U format

4. Evaluation measures

In this chapter we describe two main measures which we use to compare quality of OCR processing – Word error rate and Token accuracy. For detailed analysis we might be interested in how hard is concrete data for OCR processing. We can look at images and say if they seems to be well scanned or if they contain some graphical items which can negatively influence the results. However all these things are difficulty to quantify. Instead that we decide to look at number of Out of vocabulary words in golden text.

All used evaluation measures work with tokens which we get from UDPipe (cf. Section 3.4). Since line segmentation appears natural in OCR recognition, it could also be interesting to use an evaluation measure working on the line level. However, it does not suit our dataset. Indeed, the golden data from ELTeC are not divided into lines. On the other hand, while the B-MOD data do consist of single lines, every line is in separate picture hence line accuracy would be only picture accuracy.

4.1 Word error rate and Token accuracy

The basic idea of comparing two strings of tokens is to go through both strings and count in how many tokens they agree. However, if we compare only tokens in the same position, a single missing or remaining token in two otherwise identical strings can make all the following token pairs to be different. As such a situation is in the context of OCR clearly much better than e. g. completely wrong second half of the string, this approach turns out to be too naive.

To get more interesting evaluation measures we need to work with insertions and deletions. In particular, we introduce the *word error rate* measure (WER), a normalized variant of the classical *Levenshtein distance* ([9]). Let us briefly explain how it works. Consider three editing operations on a given token string S:

- *insertion:* inserting new token t at one (arbitrary) position in S,
- *deletion:* deleting one occurrence of token t from S,
- *substitution:* replacing one occurrence of token t in S by token t'.

Given two strings of tokens G and R (not necessarily of the same length) we compute $\mathrm{LEV}(R, G)$, the *Levenshtein distance* of G and R, as the smallest non-negative integer n such that it is possible to obtain R by performing n editing operations on G. Finally, denoting by N_G the number of tokens in G, we define

$$\mathrm{WER}(G, R) = \frac{\mathrm{LEV}(G, R)}{N_G},$$

the *word error rate* of G and R. Note that while LEV is a metric in the mathematical sense, WER is not (it is not symmetrical in its two arguments). If R is longer than G, it can happen that $\mathrm{WER} > 1$. Nevertheless, it still makes sense to interpret this quantity as an average number of errors made by an OCR system given the golden data G. For application in OCR evaluation, we think of G as the golden text, of R as the recognized text (OCR output) and we interpret the editing operations as errors appeared along the OCR process (namely deletion $\sim$ omitting a token; substitution $\sim$

altering a token and insertion $\sim$ adding a wrong token). In this context, we will use the terms *editing operation* and *editing error* interchangeably.

Let us look at an example with two string of tokens separated by spaces. To obtain

$$R = \texttt{'This is sentence from the same recognized text .'}$$

from

$$G = \texttt{'This is a sentence from the golden text'}$$

we need at least four editing operations, e. g. one substit. `'golden'`$\rightarrow$ `'recognized'`, one deletion (`'a'`) and two insertions (`'same'`,`'.'`). Therefore

$$\mathrm{WER}(G, R) = \frac{4}{8} = \frac{1}{2}.$$

It is possible in general to assign different weights/penalizations to the three operations/errors, we set all of them to one.

For computing WER we use the Wagner-Fisher algorithm (see [18]) and in addition to the current number of operations we keep also the type of editing operation. Although the optimal number of operations is unique (it is the lowest possible), the optimal sequence of editing operations (or even the numbers of their types) are not. For example to obtain $R =$ `'B A'` from $G =$ `'A B'`, we can make two substitutions (`'A'` $\rightarrow$ `'B'` and `'B'` $\rightarrow$ `'A'` at the end) or one deletion (`'A'`) and one insertion (`'A'` at the end). We solve this issue by defining the following priority order for choosing the appropriate operation type within the Wagner-Fisher algorithm: substitution $>$ insertion $>$ deletion. The reason for the first "inequality" is that interpreting OCR processing errors as substitutions seems more natural than as a combination of deletions and insertions. The rest of the order is not essential.

Another useful evaluation measure stems from – using the terminology from above – disregarding the insertions (i. e. wrong tokens additions) as errors and, essentially, only count how many tokens in the OCR output survived from the golden text. More precisely, when computing $\mathrm{LEV}(R, G)$, it can be proved that although the optimal sequences of editing operations are in general not unique, the numbers I, S, and D, denoting respectively the number of insertions, substitutions and deletions used, are well-defined. Hence we can define the number of correctly recognized tokens C as $C = N_G - S - D$ and the token accuracy

$$\mathrm{ACC}(G, R) = \frac{C}{N_G} = 1 - \mathrm{WER}(G, R) + \frac{I}{N_G} \leq 1.$$

Note that unlike the word error rate, token accuracy is a "positive" evaluation benchmark in the sense that its value close to one means a successful OCR.

4.2 Out of vocabulary

Typical images for OCR processing consist of meaningful text. Hence OCR systems can benefit from using some kind of vocabulary to improve their performance. Such training vocabularies for Tesseract are available (see links in Table B.4) and we can investigate how the number of unknown words encountered in the experiments affects the results. Note that the source vocabulary for Tesseract training came from modern

texts while most of the OCRData are novels from the 19$^{\text{th}}$ century. Let OOV (*Out of vocabulary*) denote the number of unknown tokens that appear in the golden text but not in the Tesseract vocabulary. For comparison with other texts, we use OOV per token defined as $\frac{OOV}{N_G}$, where N_G is again the number of tokens in the golden text.

5. Experiments and results

In this chapter, we introduce our experiments. First, in Section 5.2, we use the evaluation measures WER and ACC described in Section 4.1 to compare the performance of the three tested systems (five including the different modes of Tesseract, cf. Section 3) on OCRData and analyse OCR error tendencies in terms of the most frequent types of the editing errors (substitutions, deletions and insertions).

Next, in the following several sections, we look at the obtained results from the linguistic point of view and investigate whether and how OOV (see Section 4.2) can influence the accuracy of the OCR processing and how the POS tags of our OCR output look like.

All these experiments work with tokens as basic units. In the last section of this chapter, we work on the character level instead. Namely, we look closer at character changes during the OCR processing.

The overview of the whole process can be found in Figure 5.1. The first step is tokenization and tagging of both golden and OCR texts (already merged, see section 2.3) by UDPipe. Tokens are then passed to the main script which computes WER, and ACC and give us a complete list of all (editing) errors. The substitutions are used to compute the number of character substitutions. By comparing the golden texts with the Tesseract vocabulary, we estimate OOV. The last step consists of computing the distribution of POS tags. All experiments are done by original scripts (see Section B.2 in Appendix B).

5.1 General observations

In Table A.5, we can see detailed results containing the number of tokens in golden texts, ACC, WER, OOV, OOV per token and the number of insertions, deletions and substitutions from WER computation for all data. Since the language segments of OCRData are not of equal sizes, insertions (I), deletions (D) and substitutions (S) are normalized by the respective number of tokens. The results are averaged over each language (for English over each difficulty level).

As we can see, in the most cases we have much fewer deletions than other errors. The reason is that systems can recognize some noise as letters or punctuation which gives us insertion errors but it happens rarely that they completely skip a word. However, it happens that there is missing space in the OCR output which gives us some deletion errors.

Different behaviour can be observed at B-MOD data proccesed by Tesseract and GOCR where the number of deletions increases with lower image quality. The english.hard images are usually very blurred (see Figure 5.2) and systems sometimes consider them as images with no text.

5.2 Systems comparison

Figure 5.3 shows one page from the Czech data and the corresponding output from Tesseract with oem0 (Figure 5.4) and oem1 (Figure 5.5), Ocrad (Figure 5.6 and GOCR (Figure 5.7). The texts produced by Tesseract contain some mistakes but we can in

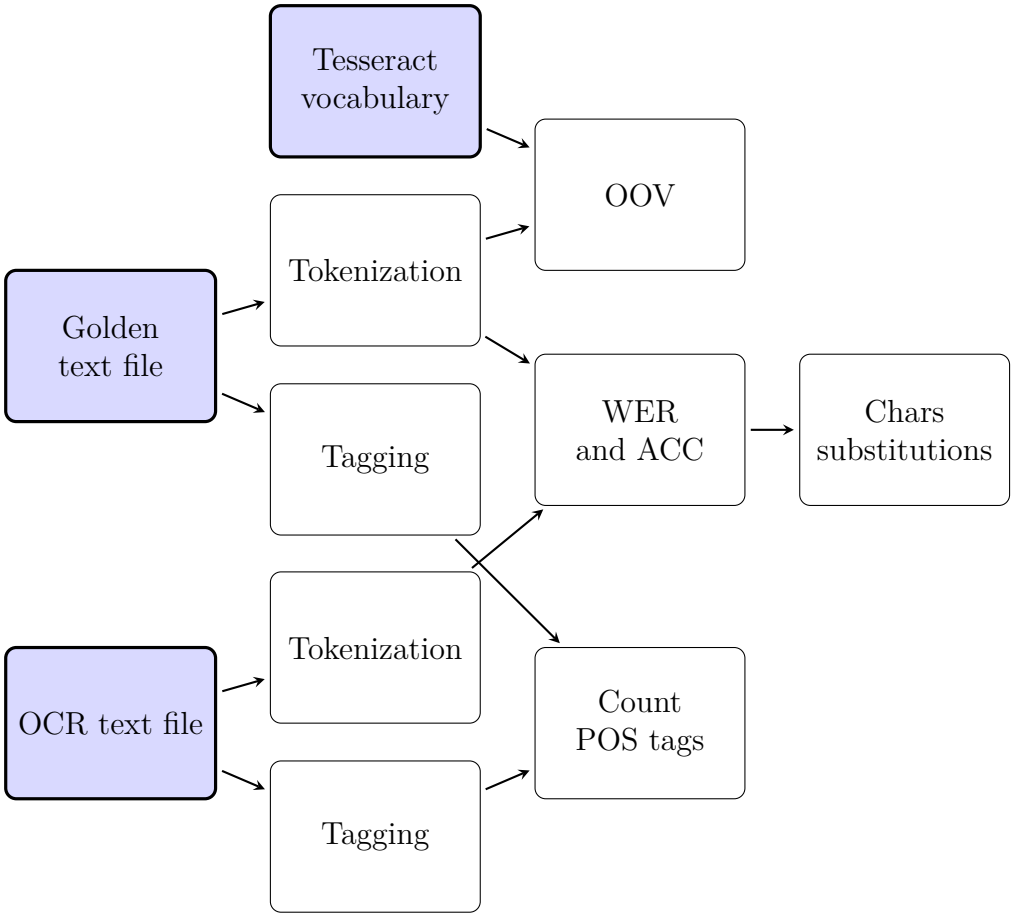


Figure 5.1: Experiments

Figure 5.2: Example of english.hard image (B-MOD)

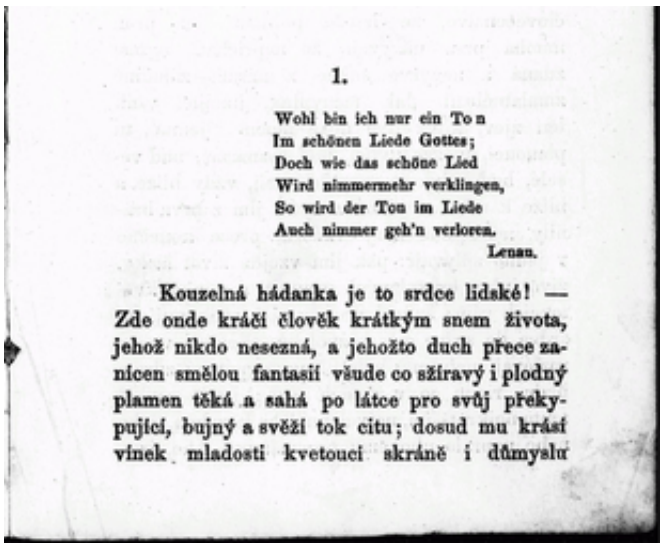
general read them and we are mostly able to fix the mistakes without looking at the scanned images. Ocrad and GOCR transcriptions contain a lot of unknown characters and the texts are not readable.

In Figure 5.8 we can see the average ACC for each language and system, Figure 5.9 shows analogous results for WER. All Tesseract modes give significantly better results than Ocrad and GOCR which corresponds to our sample page below. The ELTeC data has generally better ACC than B-MOD which corresponds to our observation that ELTeC has better quality than B-MOD. However, the situation is different for WER and Ocrad and GOCR systems. High values of WER in these cases are caused by a higher number of insertions, see Table A.5. Unlike cropped lines from B-MOD where most of the space is covered by text, novels from ELTeC contain more graphical non-text areas such as frames and sometimes even images. As we investigate closer in Section 5.5, these areas are often recognized as punctuation marks and Ocrad and GOCR tend to do this error more often.

When we look at individual languages of the ELTeC part and order them by ACC or WER, we get different orders for each system. Therefore we can not say that some

language parts are easier in general.

Recall that the first Tesseract mode oem0 uses the original module, oem1 use LSTM neural network and oem2 combines both. As we can expect, oem1 works generally better than older oem0. The mode oem2 produces very similar results on ELTeC data as oem1 and slightly worse than oem0. The largest difference can be seen at english.easy data where oem1 gives significantly better result than the other systems. It indicates that oem1 works reasonably even on lower quality data. However, english.hard data is probably too difficult and there is no big difference between the systems performances.



1.

Wohl bin ich nur ein To n
Im schönen Liede Gottes;
Doch wie das schöne Lied
Wird nimmermehr verklingen,
So wird der Ton im Liede
Auch nimmer geh'n verloren.
Lenau.

Kouzelná hádanka je to srdce lidské! — Zde onde kráčí člověk krátkým snem života, jehož nikdo nesezná, a jehožto duch přece zanícen smělou fantasií všude co sžíravý i plodný plamen těká a sahá po látce pro svůj překypující, bujný a svěží tok citu; dosud mu krásí vínek mladosti kvetoucí skráně i důmyslu

Figure 5.3: OCR image from CS0002

```
1.

Wohl bin ich nur ein To n

lm schónen Liede Gottes;

Doch wie das schóne Lied

Wird nimmermehr verklingen,

So wird der Ton im Liede

Auch nimmer geh'n verloren.
Lenan.

—Kouzelná hádanka je to srdce lidské! --
Zde onde kráčí člověk krátkým snem života,
jehož nikdo nesezná, a jehožtc duch přece
zanícen smělou fantasií všude co sžíravý i plodný
plamen těká .a. sahá po látce pro svůj
překypujíci, bujný a. svěží tok citu; dosud mu krásí
vínek _ mladosti kvetoucí skráně i důmysltr
```

Figure 5.4: Tesseract oem0 – sample output

```
1.

Wohl bin ich nur em Ton

Im schěnen Liede Gottes;

Doch wie das schěne Lied

Wird nimmermehr verklingen,

Bo wird der Ton im Liede

Auch nimmer geh'n verloren.
Lenan.

—Kouzelná hádanka je to srdce lidské! –
Zde onde kráčí člověk krátkým snem života,
jehbož nikdo nesezná, a jehožto duch přece
zanícen smělou fantasií všude co sžíravý i plodný
plamen těká a sahá po látce pro svůj
překypující, bujný a svěží tok citu; dosud mu krásí
vínek mladosti kvetoucí: skráně i důmyslw
```

Figure 5.5: Tesseract oem1 – sample output (CS0002)

```
l.

Wohl bin icb nor ein Ton
lm acbÓoen Lieda Gottea;
lDocb wie daa acb_oe Lied
Wird nim_ermebr _erBingBo,
r So wird aer Too i_ Liada
Aucb ñ_mer geb'n _eTlore_
IRn_o. '

Bou_elna bad_Ba je to ardce lidBBé! –
_Zde ond6 _ra_L _lo_a_ Brat__m an6m _i_ota,
j6boE niBdo neaazna, a jebo_to ducb p*ec6
_enlcen amélou fant&aií ___ude co B_Lra_p i ploanQ
plamen ta_a ,a _aba po latca pro a_aj
p*6Bppujlcl_ bujnp a a_é_L tok citu; doBud mu gr&B_
_6k. mladoBti _vetouci g_rané i d_mpBl_r . !'_
```

Figure 5.6: Ocrad – sample output (CS0002)

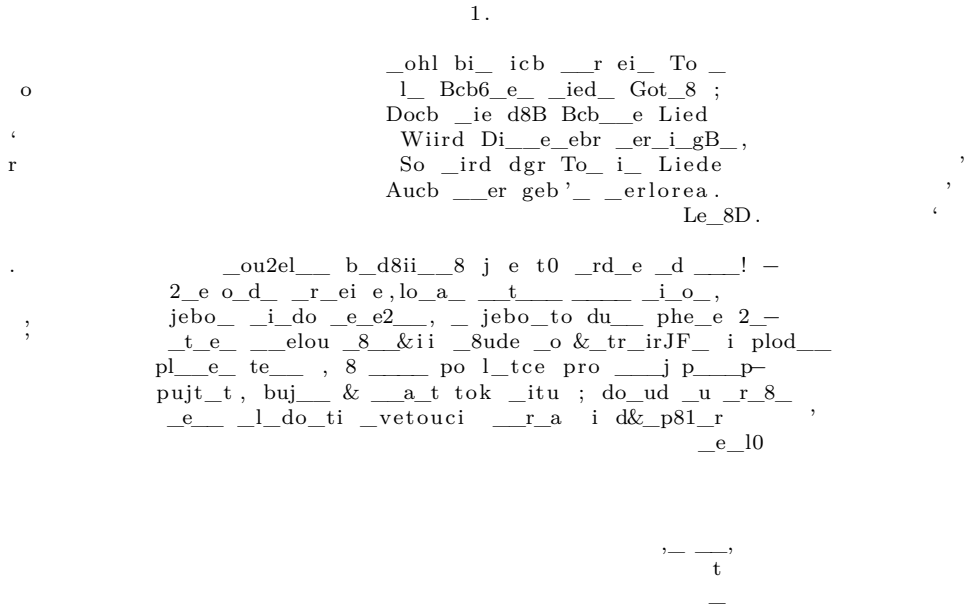


Figure 5.7: GOCR – sample output (CS0002)

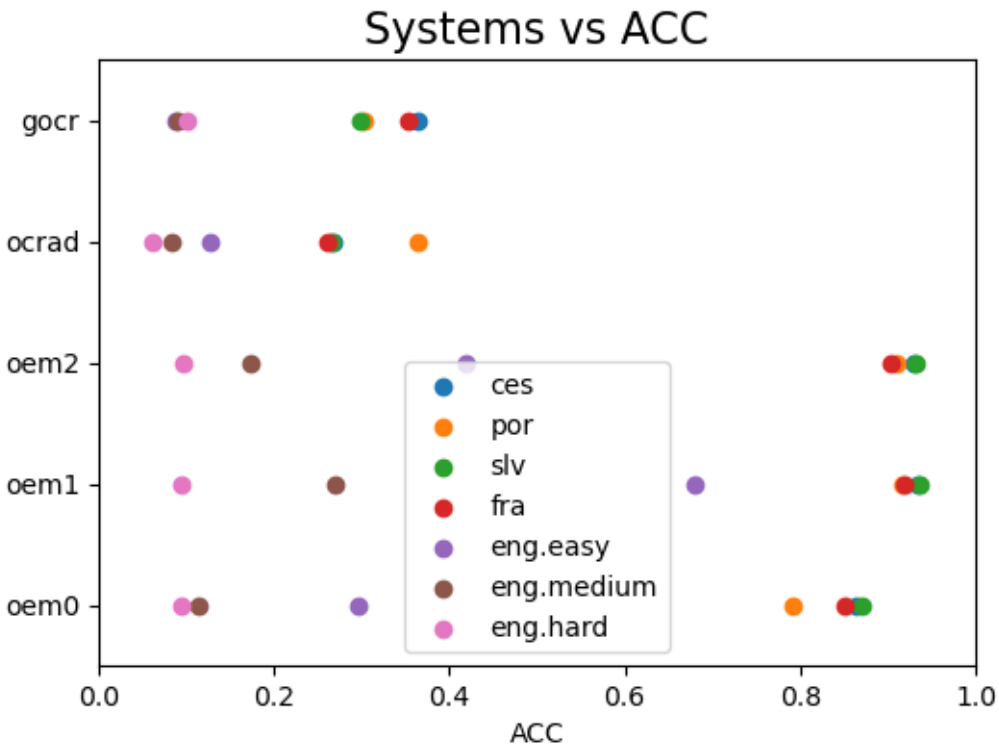


Figure 5.8: ACC for each language and system

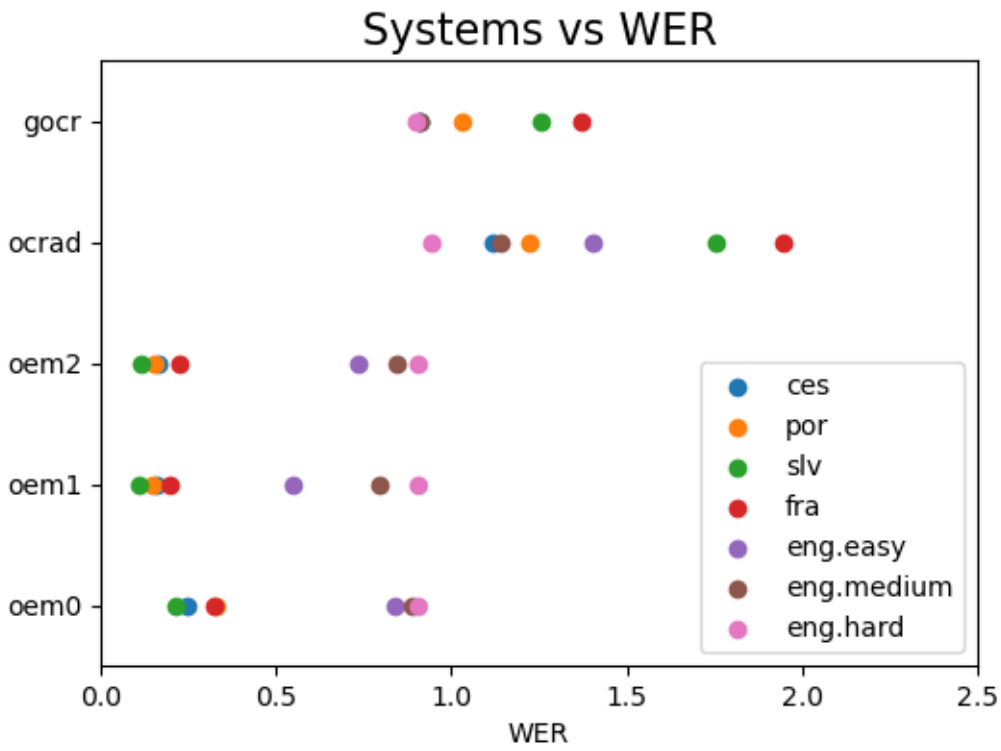


Figure 5.9: WER for each language and system

5.3 Comparing Tesseract psm modes

Segmentation is important for OCR processing and as we desribe in Section 3.1, Tesseract provides setting for some specific types of text placement. For one-line data from B-MOD, we compare two one-line modes psm7 and psm13 and the default mode. By setting psm mode we provide Tesseract with more information about the image, and hence we expect better results from it. However, as one can see in Table A.6, for our data it is more complicated.

For oem0 we get worse results we get much worse results with psm7 and psm13 for english.easy data, for english.medium still worse but with smaller difference and for english.hard even little better. For easier data psm7 looks better. When we look closer at results, we can see that psm7 and psm13 have less insertions and deletions but more substitutions which causes higher WER and worse ACC. Deletions errros are more common in worse quality data and their reduction play bigger role than increasing number of substitutions here.

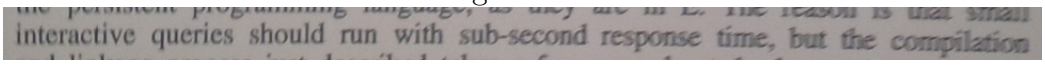
For oem1 both non-default psm modes improve the results, more for psm13. Again we can see big reduction of insertions and deletions errors but unlike oem0 numbers of substitutions are smaller too. For oem2 there is big improvement with psm7 but psm13 has worse results.

It seems that psm7 and psm13 somehow worsen the process of recognition of character itself for oem0 but it is not clear to the author, why exactly. The results of oem2 then corresponds to the fact that it is a combination of both previous modes.

Table A.6 contains also information about the number of tokens and lines in OCR outputs. The psm13 has the correct number of lines in all cases, psm7 has less and default segmentation has more than twice. It is expected behaviour as the images are

usually single lines from a longer text and some are not perfectly cropped and contain parts of lines above or below. In the default mode, Tesseract tries to recognize that as a next lines and it gives us more insertions and lines. Figure 5.10 shows the image and its transcription (all done in oem1).

OCR image from B-MOD:



Default psm mode:

```
eapgyry FU Twwey CNR. SEE CEE SR AI

interactive queries 'should run. with sub-second response time, but the conpitilin
```

Mode psm7:

```
interactive queries should run with sub-second response time, but the compilation
```

Mode psm13:

```
interactive queries should run with sub-second response time, but the compilation
```

Figure 5.10: Bad cropped line

Difference between number of tokens are smaller than between lines because, as we mentioned earlier, default segmentation mode has also smaller number of deletions and therefore the resulting number of tokens for easier data are similar for all psm modes. For english.hard data numbers of tokens with default segmentation are much smaller than in the golden text and it increase for psm7 and psm13. Tesseract sometimes recognizes bad quality image as an empty page with no text but with information there is one line of text probably tries to find at least something at the image.

5.4 OOV

We examine only correlation with ACC (number of correctly recognized tokens) not with WER because we assume deletions and insertions are usually cost by image noise and it can not be improved by a better dictionary.

The vocabularies used to compute OOV are for Tesseract and Ocrad and GOCR does not even have language-specific models but we include them too for comparison.

In Figure 5.11 we can see all files and their OOV and corr. The OOV is similar for novels of one language and therefore different accuracy of novels of one language is more likely caused by the image quality. However the scanned images of one language often come from the same source, and the image quality differs more across languages than across files of one language.

For examination of influence OOV to accuracy, we should compare similar data to reduce other causes, B-MOD data has the worse quality and they look different therefore we compute correlation only for ELTeC data (Czech, Slovenian, French, Portuguese).

In order to compute the correlation as a statistical quantity, we interpret our experimental data in a stochastic manner. We consider both ACC and OOV as random variables on probability space Ω consisting of all the novels from ELTeC. For any novel $x \in \Omega$ we denote by $N(x)$ its number of tokens and we set

$$N = \sum_{x \in \Omega} N(x)$$

to be the total number of tokens in the ELTeC golden data. Next, we introduce the probability P on Ω simply by defining its value on its each element (i. e. the elementary events in the stochastic terminology), namely

$$P(x) = \frac{N(x)}{N} \quad \text{for any } x \in \Omega.$$

This formula ensures that every token has the same weight in our statictics. The random variables $ACC, OOV : \Omega \to \mathbb{R}$ are defined in the obvious way. The sought correlation is then given by the formula

$$\operatorname{corr}(ACC, OOV) = \frac{\mathbb{E}\left(ACC \cdot OOV\right) - \mathbb{E}\left(ACC\right)\mathbb{E}\left(OOV\right)}{\sqrt{\mathbb{E}\left(ACC^2\right) - \mathbb{E}\left(ACC\right)^2} \cdot \sqrt{\mathbb{E}\left(OOV^2\right) - \mathbb{E}\left(OOV\right)^2}}.$$

The values for every system are listed in Table 5.1. Surprisingly, we see no initially anticipated negative correlation. On contrary, all the correlations are positive, some of them even quite high, which could suggest that higher OOV actually improves the accuracy. Since this seems to us far-fetched, we should provide a plausible explanation of this apparent inconsistency. We conjecture that OOV actually does influence ACC in a negative manner, but our data are not capable of demonstrating that on the level of correlation. More precisely, we believe that there are factors that influence ACC much stronger than OOV (such as the OCR image quality, font type, etc.) and which are independent of OOV. However, our dataset is too small to unavoidably show this independence. Consequently, OOV can be in our data on average accidentally higher for e. g. the files of better quality implying the positive correlation. One could argue that this is actually the case by observing that there is more tokens in the Slovenian novels than in the other three languages combined (see Table 2.1), they have simultaneously higher OOV and comparable ACC to the other languages (see Figure 5.11 below). At the same time, a glimpse on the data suggests that the Slovenian novels have relatively good quality, in the other words, their text is well-readable. However, since quantifying the last property in a rigorous way is beyond the scope of this thesis, we do not claim that this reasoning is more than an educated guess.

system	$\operatorname{corr}(ACC, OOV)$
oem0	0.340
oem1	0.397
oem2	0.409
ocrad	0.015
gocr	0.034

Table 5.1: Correlation between OOV and ACC

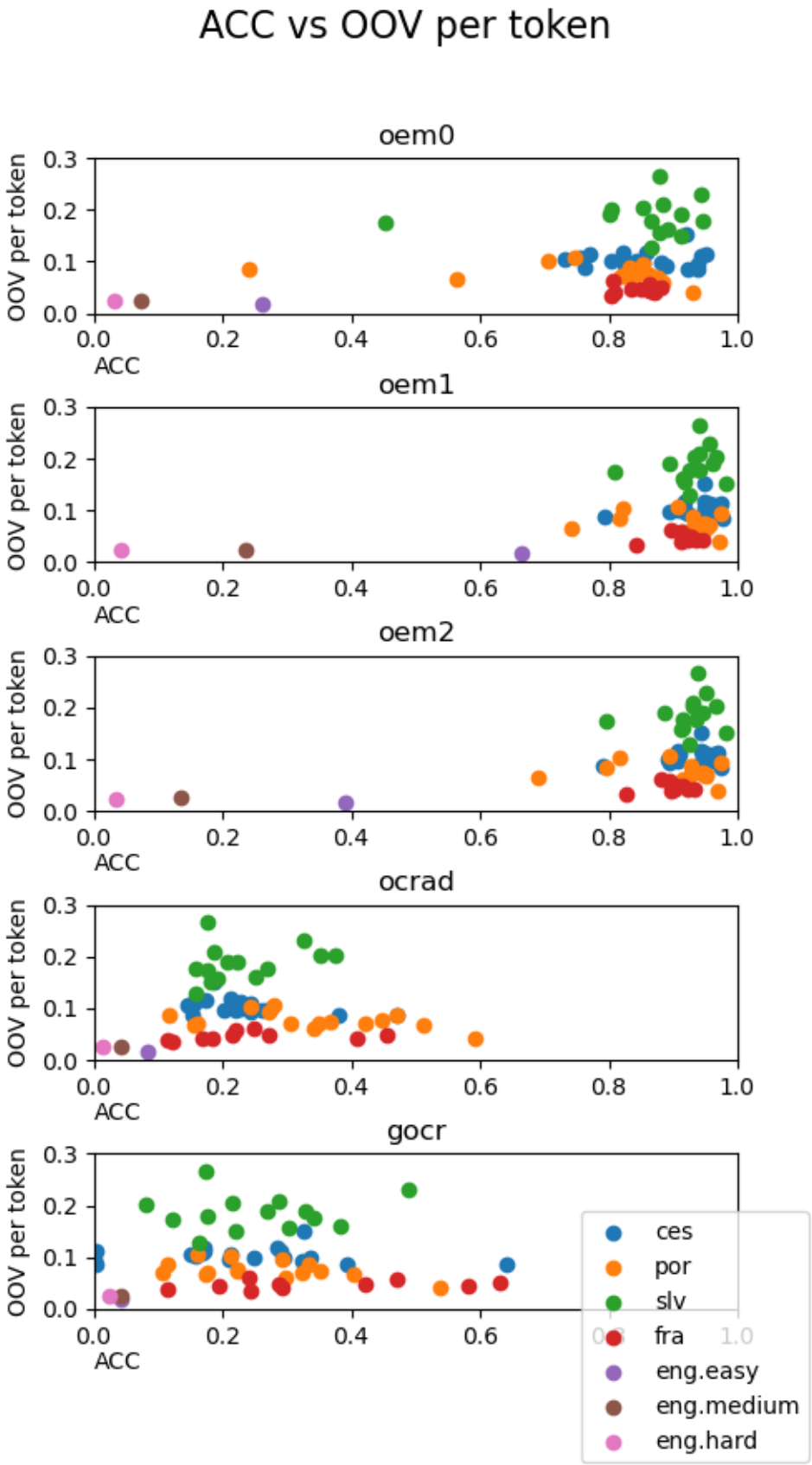


Figure 5.11: Relation between ACC and OOV per token for each system

5.5 POS tags

We do not work with random sequences of characters, therefore, it could make sense to look at the errors from the linguistic point of view. One way is looking at the distribution of POS tags of golden and OCR texts. Recall that for POS tagging we used UDPipe (see Section 3.4).

The Figure 5.12 shows the relations between the number of occurences of each tag in the golden and OCR texts averaged over all data. The most significant difference is a larger number of PUNCT (punctuation) in the OCR texts by GOCR, Ocrad and Tesseract oem0 compared to the golden texts. From the previous metrics, we know these systems are generally worse and they often recognize noise, graphical items, and other non-letter elements in texts, e. g. punctuation.

The following example shows how the systems deal with a frame around the page 5.13. First, we have plain text recognition of the highlighted part of the text (yellow). In the bottom part of the frame (red line) the mode oem0 recognizes it as some punctuation (Listing 5.14 and the oem1 as random letters5.15. Although there is the same line on the right (blue) as on the bottom, this Tesseract correctly ignores it in all cases.

The number of POS tags comes from UDPipe, let's look at what the tagging output of parts between both parts of yellow looks like. The oem0 5.16 is the only sequence of PUNCT as we expected and the sequences of letters from oem1 5.17 are mostly tagged as PROPN, probably because of the capital letter at the beginning. It corresponds to our graph, although the difference is smaller than PUNCT, the number of PROPN in OCR texts is bigger than in the gold texts.

GOCR and Ocrad are even worse and try to interpret not only the red bottom line but even the blue left and right ones. The difference between these systems is that Ocrad interprets vertical lines as next columns and therefore the output contains the left line, block of text and the right line and GOCR adds extra character to the beginning and end of the line of words.

The next reason for the large number of PUNCT in GOCR and Ocrad is both systems use `'_'` instead of unrecognized characters. When the `'_'` substitute only a few letters in a word, UDPipe try tagged it as a regular word but for shorter words, it can happen that no letter remains and then it is tagged as PROPN. The Figure 5.18 shows an example of one such that sentence of GOCR output.

The difference between other tags is smaller, except for the NOUN and SYM, all tags are more common in the golden text. Similar to the PUNCT situation the distributions of GOCR and Ocrad differ more than Tesseract. Of course, there is a question if the results of GOCR and Ocrad tagging are relevant because when the output does not make any sense automatic tagging can produce any kind of results.

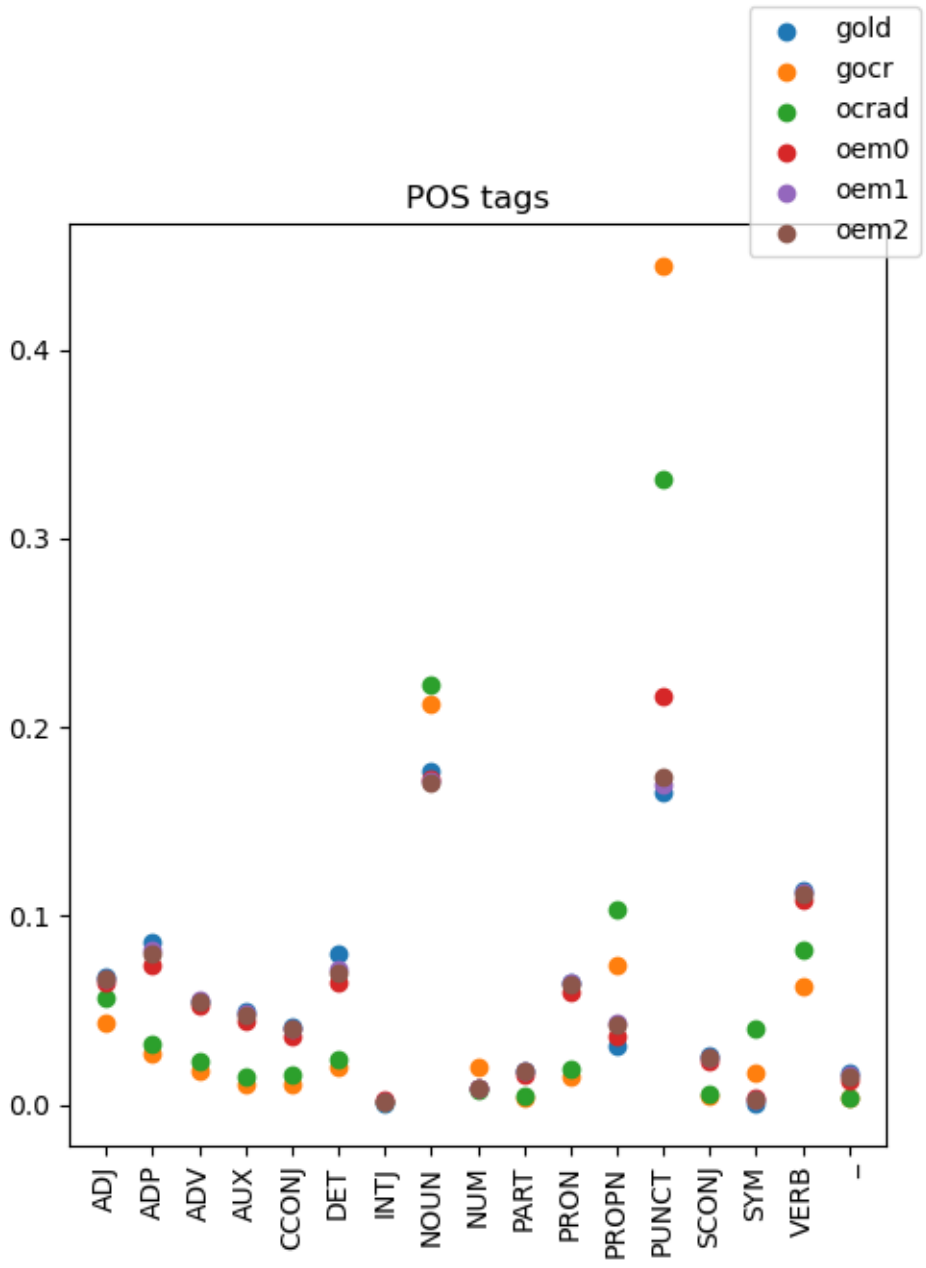
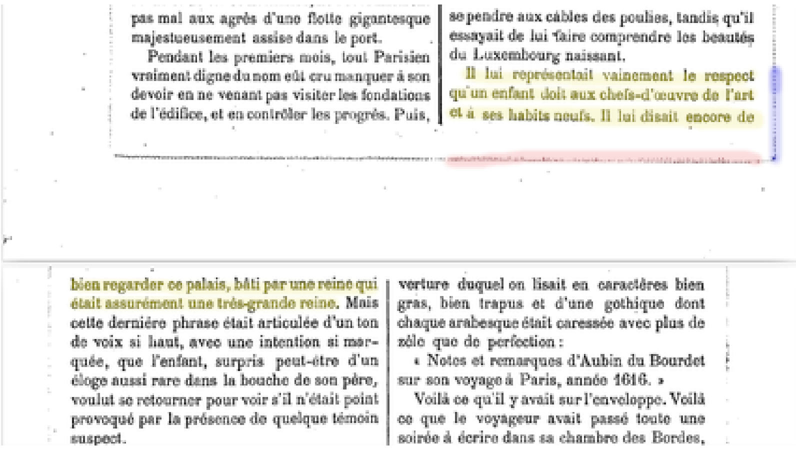


Figure 5.12: POS tags



pas mal aux agrès d'une flotte gigantesque majestueusement assise dans le port.

Pendant les premiers mois, tout Parisien vraiment digne du nom eût cru manquer à son devoir en ne venant pas visiter les fondations de l'édifice, et en contrôler les progrès. Puis, se pendre aux câbles des poulies, tandis qu'il essayait de lui faire comprendre les beautés du Luxembourg naissant.

Il lui représentait vainement le respect qu'un enfant doit aux chefs-d'œuvre de l'art et à ses habits neufs. Il lui disait encore de

bien regarder ce palais, bâti par une reine qui était assurément une très-grande reine. Mais cette dernière phrase était articulée d'un ton de voix si haut, avec une intention si marquée, que l'enfant, surpris peut-être d'un éloge aussi rare dans la bouche de son père, voulut se retourner pour voir s'il n'était point provoqué par la présence de quelque témoin suspect.

verture duquel on lisait en caractères bien gras, bien trapus et d'une gothique dont chaque arabesque était caressée avec plus de zèle que de perfection :

« Notes et remarques d'Aubin du Bourdet sur son voyage à Paris, année 1616. »

Voilà ce qu'il y avait sur l'enveloppe. Voilà ce que le voyageur avait passé toute une soirée à écrire dans sa chambre des Bordes,

Figure 5.13: FRA05702 – noise example

Il lui représentait vainement le respect
qu" un enfant doit aux chefs–d'œuvre de l'art

et à ses habits neufs. Il lui disait encore de

um,.C:üy...u.m..:.fià.t...au' .:... ....» ,...»... .. ..… . ..... '.

'x'......_....

…...w.…....….…....«va

bien regarder ce palais, bâti par une reine qui
était assurément une très–grande reine.

Figure 5.14: FRA05702 – oem0

Il lui représentait vainement le respect
qu'un enfant doit aux chels—d'œuvre de l'art

et à ses habits neufs. Il lui disait encore de

ISA D SI LT Lie A LÉO DGA EE 6 à dr000 ca

were
—
ruine esse 00 100 05 VAN

bien regarder ce palais, bâti par une reine qui
était assurément une très—grande reine.

Figure 5.15: FRA05702 – oem1

```
# sent_id = 934
# text = um,.‘:üy...u.m..:. fià.t...au’ .:... ..... .... .» ,...»... .. ........ .
    ..... ’.
1       um      um      NOUN    _       Gender=Masc|Number=Plur _       _       _
        SpaceAfter=No
2       ,       ,       PUNCT   _       _       _       _       _       SpaceAfter=No
3       .‘:üy...u.m..:. fià.t...au’     .‘:üy...u.m..:. fià.t...aue     SCONJ   _
        _       _       _       _       _
4       .:      .:      PUNCT   _       _       _       _       _       SpaceAfter=No
5       ...     ...     PUNCT   _       _       _       _       _       _
6       .....   .....   PUNCT   _       _       _       _       _       _
7       ....    ....    PUNCT   _       _       _       _       _       _
8       .»      .»      PUNCT   _       _       _       _       _       _
9       ,       ,       PUNCT   _       _       _       _       _       SpaceAfter=No
10      ...»    ...»    PUNCT   _       _       _       _       _       SpaceAfter=No
11      ...     ...     PUNCT   _       _       _       _       _       _
12      ..      ..      PUNCT   _       _       _       _       _       _
13      .....   .....   PUNCT   _       _       _       _       _       _
        _
14      .       .       PUNCT   _       _       _       _       _       _
15      .....   .....   PUNCT   _       _       _       _       _       _
16      ’       ’       PUNCT   _       _       _       _       _       SpaceAfter=No
17      .       .       PUNCT   _       _       _       _       _       SpacesAfter=\n
    \n\s\n\n

# newpar
# sent_id = 935
# text = ‘x’....._....
1       ‘x’....._....   ‘x’....._....   NOUN    _       Gender=Masc|Number=Sing _
        _       _
SpacesAfter=\n\n

# newpar
# sent_id = 936
# text = ....w.....«va


1       ....w.....«va   ....w.....«ver  VERB    _
Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin   _       _       _
    SpacesAfter=\n
2


        PROPN   _       _       _       _       _       SpacesAfter=\s\n\n
```

Figure 5.16: FRA05702 – oem0 POS tags

```
# sent_id = 821
# text = ISA D SI LT Lie A LÉO DGA EE 6 à dr000 ca
1       ISA     Ison    DET     _       Gender=Fem|Number=Sing|Poss=Yes|PronType=Prs
        _       _       _       _
2       D       D       PROPN   _       _       _       _       _       _
3       SI      si      X       _       _       _       _       _       _
4       LT      LT      PROPN   _       _       _       _       _       _
5       Lie     Lie     PROPN   _       _       _       _       _       _
6       A       a       PROPN   _       _       _       _       _       _
7       LÉO     Léo     PROPN   _       _       _       _       _       _
8       DGA     DGA     PROPN   _       _       _       _       _       _
9       EE      EE      PROPN   _       _       _       _       _       _
10      6       6       NUM     _       _       _       _       _       _
11      à       à       ADP     _       _       _       _       _       _
12      dr000   dr000   NUM     _       _       _       _       _       _
13      ca      ca      PRON    _       PronType=Dem    _       _       _
    SpacesAfter=\n\n\s\n\n

# newpar
# sent_id = 822
# text = were – ruine esse 00 100 05 VAN


1       were    were    NOUN    _       Gender=Fem|Number=Sing  _       _       _
           SpacesAfter=\n
2       –       –       PUNCT   _       _       _       _       _       SpacesAfter=\n
3       ruine   ruine   ADJ     _       Gender=Fem|Number=Sing  _       _       _
          _
4       esse    esser   NOUN    _       Gender=Fem|Number=Sing  _       _       _
          _
5       00 100 05       00 100 05       X       _       _       _       _       _
          _
6       VAN     Van     PROPN   _       _       _       _       _       SpacesAfter=\n
7


                PROPN   _       _       _       _       _       SpacesAfter=\s\n\n
```

Figure 5.17: FRA05702 – oem1 POS tags

```
# sent_id = 31
# text = Antonin js_ ___ _ite_ nej_t8r8i , byl n&hle hǐDn_otem ,_ pr_.
1	Antonin	Antonin_:H	NOUN	NNIS1-----A-----	Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing|Polarity=Pos	_	_	_	_
2	js_	js_	NOUN	NNFS2-----A-----	Case=Gen|Gender=Fem|Number=Sing|Polarity=Pos	_	_	_	_
3	___	___	PUNCT	Z:----------	_	_	_	_	_
4	_ite_	_ite_	NOUN	NNFS2-----A-----	Case=Gen|Gender=Fem|Number=Sing|Polarity=Pos	_	_	_	_
5	nej_t8r8i	nej_t8r8i	NOUN	NNFS2-----A-----	Case=Gen|Gender=Fem|Number=Sing|Polarity=Pos	_	_	_	SpaceAfter=No
6	,	,	PUNCT	Z:----------	_	_	_	_	SpacesAfter=\s\s
7	byl	být	AUX	VpYS----XR-AA----	Gender=Masc|Number=Sing|Polarity=Pos|Tense=Past|VerbForm=Part|Voice=Act	_	_	_	_
8	n	n	ADJ	VsYS----XX-AP----	Gender=Masc|Number=Sing|Polarity=Pos|Variant=Short|VerbForm=Part|Voice=Pass	_	_	_	SpaceAfter=No
9	&	&	PUNCT	Z:----------	_	_	_	_	SpaceAfter=No
10	hle	hle	ADV	Dg----------1A----	Degree=Pos|Polarity=Pos	_	_	_	SpacesAfter=\n\s\s
11	hǐDn_otem	hǐDn_ot	NOUN	NNIS7-----A-----	Animacy=Inan|Case=Ins|Gender=Masc|Number=Sing|Polarity=Pos	_	_	_	_
12	,	,	PUNCT	Z:----------	_	_	_	_	SpaceAfter=No
13	_	_	PUNCT	Z:----------	_	_	_	_	_
14	pr_	pr_	NOUN	NNFS2-----A-----	Case=Gen|Gender=Fem|Number=Sing|Polarity=Pos	_	_	_	SpaceAfter=No
15	.	.	PUNCT	Z:----------	_	_	_	_	SpaceAfter=No
```

Figure 5.18: GOCR - POS tags

5.6 Character substitutions

The errors discussed in the previous section are mostly insertions and deletions, let us now have a closer look at substitutions. The algorithm for counting WER gives us pairs of tokens that are marked as substitutes for each other. We did not distinguish substitutions according to how "bad" they are, so e. g. completely different words counted the same as a only one letter changed. In this section, we focus on the minor character changes. We look at every substitution pair of tokens of equal length (let us call these *substitution pairs)*, compare them character by character and count the most common character substitution. We expect that the most common ones will be similar-looking pairs of characters such as `'0'` and `'O'`.

Nevertheless, the assumption that substitution pairs correspond exactly to minor changes on the character level is a simplification. It can happen that one character changes to multiple and visa versa within the same token or one or several spaces can be added or deleted by the OCR process splitting one token into multiple or merging multiple in one. Such an error can still be interpreted as a minor character change, yet it most likely produces tokens of different lengths.

Let n be the total number of char substitutions within the substitution pairs and denote by n_s the number of occurrences of substitution s. Analogously to the previously used evaluation measures, we use the normalized quantity $\frac{n_s}{n}$ for comparison. Since our languages use different character sets, we count the most common substitution for each language part separately.

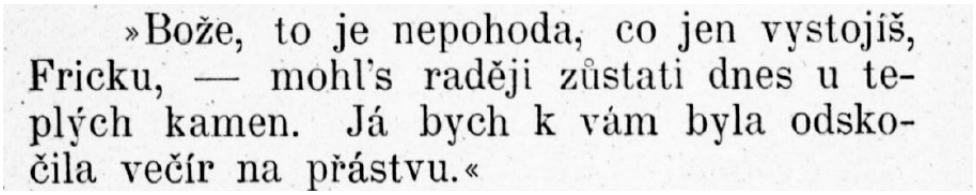
GOCR and Ocrad use the underline symbol `'_'` to mark unrecognized characters and it is thus no surprise that most of the substitutions with this system are *charac-*

ter → ’_’. Some characters are apparently harder to recognize than others because their substitutions for an underline appear with different frequency from their occurrence frequency in the golden texts.

ELTeC

Languages that use letters with diacritics (Czech, Slovenian, Portuguese and French) have a lot of character substitutions where the corresponding letters differ only in a diacritic mark. Generally, omitting diacritics by the OCR process is more frequent than adding it. (Tables 5.2, 5.3, 5.4, 5.5)

In Tesseract columns in Czech, French and Slovenian character substitutions we can see many punctuation changes (’ → ’, - → —, „ → », etc.). They look similar and usually do not change meaning of a text and therefore they can be consider as expected and not very serious errors. However, in some cases, Tesseract makes a better job and recognizes the right symbol, unlike the golden transcription. In Figure 5.19 we can see an example of the image, corresponding golden transcription and OCR output. As we work with quite old books, they sometimes use (» «) as a quotations marks which Tesseract correctly recognize but in golden text we find nowadays more common („ “) instead.


»Bože, to je nepohoda, co jen vystojíš, Fricku, — mohl's raději zůstati dnes u teplých kamen. Já bych k vám byla odskočila večír na přástvu.«

Golden text:

„Bože, to je nepohoda, co jen vystojíš, Fricku, – mohls raději zůstati dnes u teplých kamen. Já bych k vám byla odskočila večír na přástvu.“

OCR text (oem1, before merging):

»Bože, to je nepohoda,; co jen vystojíš,
Fricku, – mohl's raději zůstati dnes u te–
plých kamen. Já bych k vám byla odsko–
čila večír na přástvu.«

Figure 5.19: Bad quotations marks (CSOO22)

B-MOD (English)

The ratio of the most common substitution is smaller than in the previous languages (Table 5.6) which can be caused by multiple things. First, there are no letters with diacritics which are common errors in other languages. Second, there is no problem with bad punctuation marks in golden texts.

As it was mentioned in the section 2.2, the B-MOD data is of poorer quality than ELTeC. Therefore OCR text often does not agree with the golden texts at all and it consists more of a random sequence of characters.

The mode oem0 tends to use punctuation marks as we said in the previous section and it is probably the reason for a large number of substitution of the type “letters

to a punctuation mark" in the first column. In oem1 and oem2 there is an interesting phenomenon of a frequent usage of letters `'e'` and `'a'` in OCR text. It can be caused by the fact that there are some of the most common non-white character in golden texts (with frequencies 0.119 and 0.071 frequency) and therefore probably in training data too and Tesseract uses them more if the correct letter is unclear.

oem 0				oem 1				oem 2				gocr				ocrad			
subs.			ratio	subs.			ratio	subs.			ratio	subs.			ratio	subs.			ratio
í	→	i	0.079	"	→	“	0.131	"	→	“	0.126	a	→	_	0.073	a	→	_	0.071
a	→	n	0.046	–	→	—	0.126	–	→	—	0.114	n	→	_	0.046	v	→	_	0.055
–	→	—	0.04	ť	→	t	0.053	ť	→	t	0.039	k	→	_	0.041	h	→	b	0.038
á	→	a	0.039	í	→	i	0.035	í	→	i	0.035	v	→	_	0.039	n	→	o	0.035
"	→	“	0.037	i	→	í	0.033	i	→	í	0.03	a	→	8	0.038	ě	→	é	0.032
ť	→	t	0.036	„	→	»	0.025	„	→	»	0.024	m	→	_	0.038	y	→	p	0.029
i	→	í	0.033	.	→	,	0.021	v	→	y	0.02	s	→	_	0.032	e	→	_	0.029
a	→	&	0.026	“	→	«	0.02	“	→	«	0.019	á	→	_	0.029	s	→	a	0.028
a	→	e	0.024	,	→	;	0.016	.	→	,	0.018	í	→	i	0.021	s	→	_	0.028
a	→	u	0.021	a	→	á	0.015	a	→	á	0.016	y	→	p	0.019	k	→	_	0.027

Table 5.2: Most common char substitution – Czech

oem 0				oem 1				oem 2				gocr				ocrad			
subs.			ratio	subs.			ratio	subs.			ratio	subs.			ratio	subs.			ratio
’	→	’	0.486	’	→	’	0.552	’	→	’	0.536	a	→	_	0.105	a	→	_	0.055
à	→	a	0.058	–	→	—	0.056	–	→	—	0.045	e	→	_	0.041	e	→	c	0.038
–	→	—	0.035	t	→	l	0.024	t	→	l	0.031	n	→	_	0.037	d	→	_	0.027
	→	‘	0.033	î	→	i	0.019	î	→	i	0.015	s	→	_	0.031	e	→	_	0.026
-	→	—	0.024	e	→	é	0.014	e	→	é	0.014	d	→	_	0.024	t	→	L	0.021
ê	→	é	0.011	ê	→	é	0.01	i	→	1	0.014	l	→	_	0.023	s	→	_	0.02
a	→	e	0.011	c	→	e	0.009	c	→	e	0.012	t	→	_	0.022	e	→	6	0.016
u	→	n	0.01	.	→	,	0.008	ê	→	è	0.01	é	→	_	0.021	l	→	_	0.014
R	→	B	0.009	ê	→	è	0.008	ê	→	é	0.008	u	→	_	0.018	s	→	g	0.012
c	→	e	0.009	i	→	I	0.008	e	→	c	0.007	v	→	_	0.015	s	→	a	0.011

Table 5.3: Most common character substitution – French

oem 0			oem 1			oem 2			gocr			ocrad		
subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio
„	→ ,	0.1	„	→ »	0.13	„	→ »	0.123	a	→ _	0.067	a	→ _	0.065
,	→ .	0.076	“	→ "	0.118	“	→ "	0.091	e	→ g	0.037	e	→ c	0.039
“	→ "	0.075	“	→ «	0.046	“	→ «	0.06	e	→ _	0.018	e	→ _	0.029
e	→ o	0.031	„	→ ,	0.031	„	→ ,	0.032	e	→ c	0.017	a	→ h	0.024
“	→ ,	0.029	ó	→ o	0.028	ó	→ o	0.022	s	→ g	0.016	s	→ _	0.022
o	→ e	0.023	é	→ e	0.019	é	→ č	0.021	s	→ _	0.016	k	→ _	0.02
ó	→ o	0.019	é	→ č	0.019	c	→ e	0.018	a	→ 8	0.015	i	→ l	0.019
.	→ ,	0.016	á	→ a	0.015	e	→ č	0.015	n	→ _	0.015	č	→ _	0.018
e	→ c	0.015	.	→ ,	0.014	.	→ ,	0.015	k	→ _	0.014	v	→ _	0.017
á	→ a	0.011	e	→ č	0.013	é	→ e	0.013	č	→ e	0.013	d	→ _	0.016

Table 5.4: Most common character substitution – Slovenian

oem 0			oem 1			oem 2			gocr			ocrad		
subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio
á	→ a	0.048	á	→ a	0.091	á	→ a	0.076	e	→ g	0.054	s	→ _	0.11
,	→ .	0.04	é	→ e	0.052	é	→ e	0.049	a	→ _	0.049	a	→ _	0.093
é	→ e	0.031	í	→ i	0.046	í	→ i	0.044	a	→ 8	0.046	e	→ c	0.069
c	→ e	0.029	ó	→ o	0.032	c	→ e	0.031	m	→ _	0.021	t	→ l	0.052
e	→ c	0.025	c	→ e	0.025	ó	→ o	0.03	s	→ _	0.02	e	→ _	0.031
i	→ í	0.021	ê	→ e	0.025	ê	→ e	0.023	e	→ c	0.02	i	→ l	0.027
í	→ i	0.018	.	→ ,	0.023	a	→ á	0.022	e	→ _	0.019	d	→ _	0.024
t	→ l	0.018	a	→ á	0.022	t	→ l	0.021	s	→ g	0.015	ã	→ á	0.012
.	→ ,	0.017	ú	→ u	0.021	.	→ ,	0.021	t	→ l	0.015	m	→ _	0.012
a	→ n	0.016	,	→ .	0.019	ú	→ u	0.02	e	→ Æ	0.015	,	→ .	0.011

Table 5.5: Most common character substitution – Portuguese

oem 0			oem 1			oem 2			gocr			ocrad		
subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio	subs.		ratio
,	→ .	0.014	o	→ e	0.012	t	→ l	0.015	a	→ _	0.043	e	→ _	0.041
t	→ l	0.011	o	→ a	0.01	e	→ c	0.013	o	→ _	0.039	t	→ _	0.034
e	→ m	0.01	s	→ e	0.01	e	→ o	0.012	e	→ _	0.036	a	→ _	0.03
e	→ c	0.01	n	→ e	0.009	i	→ l	0.01	t	→ _	0.034	o	→ _	0.026
e	→ .	0.01	e	→ s	0.009	,	→ e	0.008	.	→ _	0.028	i	→ _	0.025
t	→ .	0.009	,	→ .	0.009	.	→ ,	0.008	i	→ _	0.024	n	→ _	0.022
,	→ -	0.009	.	→ ,	0.008	a	→ n	0.008	n	→ _	0.023	s	→ _	0.022
a	→ .	0.007	t	→ i	0.007	o	→ e	0.007	,	→ _	0.023	h	→ _	0.017
t	→ m	0.007	s	→ n	0.007	.	→ e	0.006	s	→ _	0.021	t	→ l	0.014
o	→ m	0.007	e	→ o	0.007	,	→ .	0.006	h	→ _	0.019	r	→ _	0.013

Table 5.6: Most common character substitution – English

Conclusion

We created our OCRData dataset by adjusting the primary non-OCR dataset ELTeC for OCR experiments. OCRData does not contain 100 percent perfect transcriptions but we tried to choose the best compromise between automation and therefore a larger amount of data and manual checking.

The best-observed system was Tesseract with oem1, other Tesseract modes oem0 and oem2 give a little worse results. Ocrad and GOCR are much worse and not usable for practical usage.

The assumption that OOV influences the accuracy of OCR processing was not confirmed. We find out the OCR text often contains much more punctuation marks than golden data, mostly because of visual noise in images. Lastly, we make statistics of most common character substitutions. The results are influenced by inaccurate golden transcriptions but besides that our assumptions were confirmed.

Besides statistical analysis, we wanted to visualizing output and its errors. However for that we need more visually structured data that we have. Therefore we implement our WER script to print all editing errors to have at least some option to manually examine the errors.

Bibliography

[1] Rafael C. Carrasco. An open-source ocr evaluation tool. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, page 179–184, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325882. doi: 10.1145/2595188.2595221. URL `https://doi.org/10.1145/2595188.2595221`.

[2] Michel Généreux and Diego Spano. Nlp challenges in dealing with ocr-ed documents of derogated quality. In *Workshop on Replicability and Reproducibility in Natural Language Processing: adaptive methods, resources and software at IJCAI*, pages 25–27, 2015.

[3] Karez Hamad and Mehmet Kaya. A detailed analysis of optical character recognition technology. *International Journal of Applied Mathematics, Electronics and Computers*, 4:244–244, 12 2016. doi: 10.18100/ijamec.270374.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] P. Jain, K. Taneja, and H. Taneja. Which OCR toolset is good and why : A comparative study. In *Kuwait Jurnal of Science*, volume 48, 2021. doi: 10.48129/kjs.v48i2.9589.

[6] Sukhvir Kaur, PS Mann, and Shivani Khurana. Page segmentation in ocr system-a review. *International Journal of Computer Science and Information Technologies*, 4(3):420–2, 2013.

[7] Kimmo Tapio Kettunen, Jukka Kervinen, and Jani Mika Olavi Koistinen. Creating and Using Ground Truth OCR Sample Data for Finnish Historical Newspapers and Journals. In *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference*, volume 2084, pages 162–169, 2018. doi: 10138/312778.

[8] M. Kišš, M. Hradiš, and O. Kodym. Brno Mobile OCR Dataset. *International Conference on Document Analysis and Recognition (ICDAR)*, September 2019: 1352–1357, 2019. ISSN 1520-5363.

[9] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[10] Thi-Tuyet-Hai Nguyen, Adam Jatowt, Mickael Coustaty, Nhu-Van Nguyen, and Antoine Doucet. Deep Statistical Analysis of OCR Errors for Effective Post-OCR Processing. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 29–38, 2019. doi: 10.1109/JCDL.2019.00015.

[11] Carolin Odebrecht, Lou Burnard, Borja Navarro Colorado, and Christof Schöch. European Literary Text Collection (ELTeC): Release with 10 collections of at least 50 novels., November 2019. URL `https://doi.org/10.5281/zenodo.4271467`.

[12] David A Smith and Ryan Cordell. A research agenda for historical and multilingual optical character recognition. *NUlab, Northeastern University.@ https://ocr. northeastern. edu/report*, page 36, 2018.

[13] R. Smith. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, 2007. doi: 10.1109/ICDAR.2007.4376991.

[14] Milan Straka. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/K18-2020. URL `https://www.aclweb.org/anthology/K18-2020`.

[15] eds. TEI Consortium. TEI P5: Guidelines for Electronic Text Encoding and Interchange, April 2022. URL `https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html`.

[16] Martin Tomaschek. Evaluation of off-the-shelf ocr technologies. *Bachelor thesis Masaryk University, Brno, Czech Republic*, 2018.

[17] Rohit Verma and Jahid Ali. A-survey of feature extraction and classification techniques in ocr systems. *International Journal of Computer Applications & Information Technology*, 1(3):1–3, 2012.

[18] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, jan 1974. ISSN 0004-5411. doi: 10.1145/321796.321811. URL `https://doi.org/10.1145/321796.321811`.

List of Figures

List of Tables

A. Appendix

A.1 List of novels from ELTeC

Code	Author	Title
CS0001	Stroupežnická, Marie	Bouře aneb oučinky zlého svědomí
CS0002	Janda Cidlinský, Bohumil	Jaroslav
CS0003	Žížala-Donovský, Václav	Karbaník a rodina jeho
CS0005	Křičenský, Josef Jaroslav	Lichvář a pokoutník
CS0006	Pravda, František	Krejčí Fortunat
CS0007	Prokop, F. F.	Jeden rok z mladého života
CS0008	Formánek-Činoveský, Jan	Bitva u Nýrska
CS0009	Formánek-Činoveský, Jan	Smířená vina
CS0010	Sabina, Karel	Jen tři léta!
CS0011	Švestka, Josef	Poslední večer v roce
CS0013	Mollenda, Václav	Večer svatojanský
CS0014	Vlček, Václav	Paní lichnická
CS0015	Vávra, Jan	Ratmír
CS0018	Švestka, Josef	Pravý přítel
CS0019	Vítek, Jan	Dva nestejní bratři
CS0020	Karas, Matěj	Český plavec na severu
CS0021	Unger, Ludvík	Mlada, kněžna abatyše
CS0022	Kuchař, Josef	Pašerova Anežka
CS0023	Viková-Kunětická, Božena	Silhouetty mužů

Table A.1: List of Czech novels

Code	Author	Title
FRA00801	Competesse Dash	Le château de Pinon I
FRA02001	Gilbert Jehan	Vers le pôle en aéroplane
FRA03201	Stella Blandy	Rouzétou
FRA05702	Auguste Maquet	La Maison du baigneur
FRA05801	Pierre Mille	Louise et Barnavaux
FRA06001	Hugues Rebell	Les Nuits chaudes du Cap français
FRA06101	Jules Sandeau	Mademoiselle de la Seiglière
FRA06201	Horace de Viel-Castel	Archambaud de Comborn
FRA06401	Claire de Chandeneux	Les Ménages militaires
FRA06501	Paul Sescau Gyp	Totote

Table A.2: List of French novels

Code	Author	Title
POR0044	Maria Amália Vaz de Carvalho	Serões no campo
POR0045	Antonio Augusto Teixeira de Vasconcellos	A ermida de Castromino
POR0046	Francisco Gomes de Amorim	Os selvagens
POR0047	José da Silva Mendes Leal	Infaustas Aventuras de Mestre Marçal Estouro
POR0049	António Francisco Barata	O último cartuxo da Scala Caeli de Évora: Romance histórico
POR0050	Carlos Pinto de Almeida	A conquista de Lisboa
POR0067	J.P. Oliveira Martins	Phebus Moniz
POR0071	J. A. d'Oliveira Mascarenhas	O Trovador da Infanta
POR0073	Raul Azevedo	Tríplice Aliança
POR0075	Cosme Velho	Miss Kate
POR0078	Eduardo de Noronha	O agonizar de uma dinastia
POR0080	D. João da Câmara	O Conde de Castel Melhor
POR0081	A.M. da Cunha e Sá	Da parte d'el-rei
POR0082	Júlio César Machado	A vida em Lisboa
POR0083	Caïel	Retalhos de verdade
POR0085	Augusto Sarmento	Providência

Table A.3: List of Portuguese novels

Code	Author	Title
SLV00024	Bedenek, Jakob	Od pluga do krone
SLV00048	Govekar, Fran	V krvi
SLV00058	Kersnik, Janko	Ciklamen
SLV00072	Meško, Ksaver	Kam plovemo
SLV00090	Štrukelj, Ivan	Spletke
SLV00092	Štrukelj, Ivan	Zmota in povrat
SLV00094	Bartel, Anton	Pomladanski vetrovi
SLV00103	Koder, Anton	Zvezdana
SLV00111	Detela, Fran	Pegam in Lambergar
SLV00112	Detela, Fran	Veliki grof
SLV00126	Jaklič, Fran	Ljudska osveta
SLV00132	Maselj Podlimbarski, Fran	Gorski potoki
SLV00135	Janežič-Kraljev, Ivan	Gospa s pristave
SLV00136	Tavčar, Ivan	Času primerna povest iz prihodnjih dob.
SLV00174	Mencinger, Janez	Abadon
SLV00187	Kersnik, Janko	Agitator
SLV00194	Kersnik, Janko	Na Žerinjah
SLV00216	Jurčič, Josip	Rokovnjači

Table A.4: List of Slovenian novels

A.2 Results

	POR	FRA	CES	SLV	eng.e	eng.m	eng.h
Golden text							
tokens	70 179	44 998	66 997	188 802	11 108	12 010	10 118
OOV	5 511	2 096	6 951	44 227	201	306	249
OOV per token	0.079	0.047	0.104	0.234	0.018	0.025	0.025
Tesseract oem0							
ACC	0.791	0.852	0.865	0.871	0.296	0.115	0.094
WER	0.331	0.322	0.249	0.215	0.838	0.888	0.906
S	0.188	0.134	0.104	0.113	0.614	0.651	0.323
I	0.122	0.174	0.114	0.085	0.134	0.002	0
D	0.021	0.014	0.031	0.016	0.09	0.234	0.583
Tesseract oem1							
ACC	0.918	0.919	0.934	0.937	0.681	0.269	0.094
WER	0.147	0.198	0.157	0.11	0.547	0.793	0.906
S	0.073	0.07	0.042	0.051	0.247	0.616	0.397
I	0.065	0.117	0.091	0.048	0.228	0.062	0
D	0.009	0.011	0.023	0.012	0.073	0.115	0.508
Tesseract oem2							
ACC	0.91	0.903	0.929	0.933	0.418	0.174	0.097
WER	0.155	0.222	0.166	0.117	0.733	0.846	0.903
S	0.08	0.085	0.048	0.055	0.495	0.683	0.413
I	0.065	0.125	0.095	0.05	0.151	0.02	0
D	0.01	0.012	0.023	0.012	0.087	0.143	0.49
Ocrad							
ACC	0.364	0.26	0.267	0.265	0.126	0.083	0.062
WER	1.224	1.949	1.118	1.755	1.402	1.138	0.943
S	0.623	0.735	0.73	0.733	0.874	0.917	0.935
I	0.589	1.209	0.386	1.02	0.528	0.22	0.004
D	0.013	0.005	0.003	0.001	0	0	0.003
GOCR							
ACC	0.302	0.353	0.365	0.297	0.088	0.091	0.102
WER	1.032	1.371	0.909	1.255	0.912	0.909	0.898
S	0.678	0.633	0.54	0.7	0.68	0.378	0.227
I	0.334	0.724	0.273	0.552	0.001	0	0
D	0.02	0.014	0.096	0.003	0.232	0.531	0.671

Table A.5: Complete results

File and psm	easy default	easy 7	easy 13	med. default	med. 7	med. 13	hard default	hard 7	hard 13
Golden text									
Tokens		11098			12010			10118	
Lines		1000			1000			1000	
Tesseract oem0									
Tokens	11 089	11 387	10 532	8 682	9 445	10 246	3 585	6 226	6 235
Lines	2 396	968	1 000	1 843	887	1 000	974	819	1 000
ACC	0.262	0.074	0.013	0.073	0.04	0.028	0.032	0.036	0.071
WER	0.878	1.008	0.989	0.93	0.962	0.972	0.968	0.964	0.931
S	6 820	9 743	10 378	7 817	8 977	9 927	3 273	5 880	5 938
I	1 487	864	19	29	13	1	0	0	14
D	1 001	80	87	2 816	2 060	1 242	5 895	3 342	3 464
Tesseract oem1									
Tokens	12 324	10 295	10 608	10 868	9 425	10 371	4 436	5 768	6 775
Lines	2 397	968	1 000	1 839	886	1 000	966	819	999
ACC	0.665	0.718	0.757	0.237	0.307	0.342	0.043	0.049	0.086
WER	0.573	0.297	0.264	0.829	0.703	0.669	0.957	0.95	0.918
S	2 739	2 526	2 360	7 399	5 788	6 311	4 021	5 292	6 920
I	2 531	158	222	749	111	126	1	0	45
D	810	466	217	1 380	2 177	1 254	5 143	3 777	2 323
Tesseract oem2									
Tokens	11 317	10 928	14 125	10 007	9 511	12 842	4 519	6 071	7 678
Lines	2 397	968	1 000	1 843	887	1 000	974	819	1 000
ACC	0.39	0.621	0.131	0.136	0.251	0.074	0.036	0.013	0.069
WER	0.768	0.46	1.203	0.885	0.765	1.046	0.964	0.992	0.933
S	5 499	3 478	9 188	8 200	6 430	10 603	4 175	9 462	7 353
I	1 678	864	3 543	243	189	1 384	1	45	20
D	964	539	21	1 721	2 163	41	4 961	32	2 067

Table A.6: Psm and oem mode (English)

B. Electronic attachments

The main directory attached to the text contains a directory *src* with pre-processing and evaluation scripts described closer in section B.2 and *data* directory with input images, golden texts and results of all experiments. A detailed description of the files in it can be found in section B.4.

B.1 Requirements

- Tesseract version 4.1.1 with Czech, English, Portuguese, Slovenian and French language mode. We need models which support both old legacy and new LSTM engine[1].
- GOCR version 0.52
- GNU Ocrad version 0.27
- UDPipe version 2.0 saved in the main directory, language models in *udpipemodel* directory, which is again saved in the main directory. Placement can be changed in Makefile. All used models for our experiments we can see in Table B.1

Language	UDPipe model
Czech	czech-cac-ud-2.3-181115.udpipe
Portuguese	portuguese-bosque-ud-2.3-181115.udpipe
Slovenian	slovenian-ssj-ud-2.3-181115.udpipe
French	french-gsd-ud-2.3-181115.udpipe
English	english-ewt-ud-2.3-181115.udpipe

Table B.1: UDPipe models

B.2 Developer documentation

For pre-processing, experiments and visualisation of the results, we use external programs (mostly OCR systems) and several original scripts (all run on Linux). The whole process includes many steps and we want to have the option to do them independently, therefore we use a larger number of simple scripts and for their execution we use Makefile.

Most of the scripts are written in Python3, two in Bash. Then we use external tools described in chapter 3 and command-line program *pdftopnm* for converting images.

List of the scripts and their descriptions

- **wer.py** Computes WER (including the number substitutions, insertions and deletions) and ACC. Also prints a list of all errors with the nearest context. WER is computed by the Wagner-Fisher algorithm. ([18])

[1] `https://github.com/tesseract-ocr/tessdata`

- **extract_links.py** Extracts link to scanned image from ELTeC XML file header.

- **merge_word.py** Goes through all file and merge divided word at the end of the line.

- **xmltotext.py** Extracts all text content from a body of ELTeC XML file.

- **char_sub.py** Counts character substitution from complete list of errors.

- **count_tag.py** Counts distribution of POS of all input file and draw a graph.

- **oov.py** Prints all out-of-vocabulary words and their number.

- **statistics.py** Reads results and averaged them over language. Makes graph of the relation between WER and systems, ACC and systems and OOV per token and ACC for each system. Computes correlation between OOV per token and ACC for each system.

- **print_result.sh** Prints WER, ACC, OOV, OOV per token and number of insertions, deletions and substitutions for the input file.

- **tifftopnmpages.sh** Converts multi-page TIFF files to individual PNM images.

B.3 Usage

For executing OCR systems and scripts described in the previous section we use Makefile. Most of the command requires external variables to specify language, system, code of the file and other parameters. Table B.2 shows a list of all possible commands and their required variables, Table B.3 shows possible values for each variable.

For easier usage, most of the command has also variant which ends with *_lang* which executes commands for all novels of one language without passing individual novels' codes. For example, following command merges divided words at end of the lines in Tesseract oem0 output for all Czech novels.

```
make merge_lang lang=ces system=oem0
```

command	variables	description
xmltotext	lang, code	execute xmltotext.py
pdftotiff20	lang, code, bot_even, bot_odd, up_even, up_odd, start	convert PDF to TIFF using *pdftoppm* tool, take 20 pages from page *start*, crop image by *bot_even*, *bot_odd*, *top_even*, *top_odd* pixels on bottom and top of even and odd pages
tifftopnm	lang, code	execute tifftopnmpages.sh
gocr	lang, code	execute GOCR
ocrad	lang, code	execute Ocrad
tesseract_all	lang, code	execute all three Tesseract modes
tesseract_psm	lang, code, psm	execute all three Tesseract modes with non-default psm mode
merge	lang, code, system	execute merge.py
nomerge	lang, code, system	if no merge is needed, copy files to right directory
tokenize	lang, code, system	execute UDPipe tokenization for OCR output and golden text
tags_lang	lang, system	execute UDPipe tokenization for OCR outputs and golden texts of one language
tags_count	-	execute count_tag.py for all languages
wer	lang, code, system	execute wer.py
oov	lang, code	execute oov.py for golden text
char_sub_lang	lang, system	execute char_sub.py for all files of one language
print	lang, code, [psm]	execute print_result.sh
print_all	lang	execute print_result.sh for all files of one language
graphs	-	execute statistics.py for all languages

Table B.2: Makefile basic commands

variable	possible values
lang	ces, fra, por, slv, eng.easy, eng.medium, eng.hard
code	for each language codes from data/{lang}/{lang}-code.txt
system	oem0, oem1, oem2, ocrad, gocr
psm	number of psm mode (0 – 13), we use 7 and 13
bot_even, bot_odd, up_even, up_odd	number of pixels
start	page number

Table B.3: Possible values of variables for Makefile commands

B.4 Data

The data directory contains a directory for each of the four languages from ELTeC and three directories for B-MOD for every difficulty setting. Graphs of results are by default saved here too.

Content of the data directory

- ces/
- eng.easy/
- eng.medium/
- eng.hard/
- fra/
- por/
- slv/
- graphs images
- eng-{system}.chars files

Every language directory contains directories with source files and detailed results of experiments. Every tested system has its own set of directories, {system} means oem0, oem1, oem2, gocr or ocrad. There are only a few differences between ELTeC and B-MOD directories, hence we listed them together. In the B-MOD context novel means difficulty set (english.easy, english.medium, english.hard) and page means a one-line image.

Note: For reducing the size of the attachment, gold.pnm directories are empty because they can be easily generate by

```
make tifftopnm_lang lang={lang}
```

Content of a language directory

- **gold.xml/** Original ELTeC file in XML (only ELTeC directories)
- **gold.txt/** Golden text extracted from ELTeC XML and manually shorted, for B-MOD lines transcriptions
- **gold.token/** Tokenized golden text, one token on one line.
- **gold.conllu/** Tokenized and tagged golden text in CoNLL-U format
- **gold.oov/** All out of vocabulary words and their number
- **gold.pdf/** Full scanned novel in PDF (only ELTeC)
- **gold.tiff/** 20-pages TIFF file of each novel, manually cropped (ELTeC), 1000-pages TIFF file for every difficulty set (B-MOD)

- **gold.pnm/** Directories for every novel, each PNM images for every page, converted from TIFF files
- **ocr.{system}/** OCR output
- **ocr.{system}/.txt** merged OCR output for ELTeC, for B-MOD same files as in ocr.{system}
- **ocr.{system}.token/** Tokenized OCR output
- **ocr.{system}.conllu/** Tokenized and tagged OCR output in CoNLL-U format
- **ocr.{system}.result/** For every novel complete list of editing errors and their numbers, WER and ACC
- **ocr.{system}.chars/** Character substitutions and their number
- **{lang}.wordlist** Tesseract vocabulary B.4
- **{lang}-results.csv** Complete results (WER, ACC, number of editing errors, OOV) for each novel
- **{lang}-code.txt** List of all novels' codes
- **{system}-all.chars** List of character substitutions of language for the system.

`https//github.com/tesseract-ocr/langdata/blob/master/ces/ces.wordlist`
`https//github.com/tesseract-ocr/langdata/blob/master/slv/slv.wordlist`
`https//github.com/tesseract-ocr/langdata/blob/master/por/por.wordlist`
`https//github.com/tesseract-ocr/langdata/blob/master/fra/fra.wordlist`
`https//github.com/tesseract-ocr/langdata/blob/master/eng/eng.wordlist`

Table B.4: Links to Tesseract train vocabularies