Optical character recognition (OCR)

Code

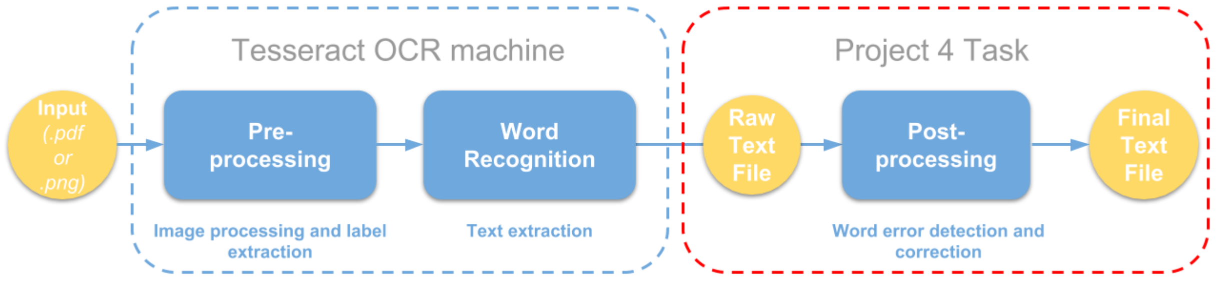
Group 7

GU4243/GR5243: Applied Data Science

Introduction

Optical character recognition (OCR) is the process of converting scanned images of machine printed or handwritten text (numerals, letters, and symbols), into machine readable character streams, plain (e.g. text files) or formatted (e.g. HTML files). As shown in Figure 1, the data *workflow* in a typical OCR system consists of three major stages:

- Pre-processing
- OCR character recognition
- Post-processing



We have processed raw scanned images through the first two steps are relying on the Tessearct OCR machine (https://en.wikipedia.org/wiki/Tesseract_(software)). R package tutorial can be found here (https://www.r-bloggers.com/the-new-tesseract-package-high-quality-ocr-in-r/).

BUT this is not the FOCUS of this project!!!

In this project, we are going to **focus on the third stage – post-processing**, which includes two tasks: *error detection* and *error correction*.

1 Step 1 - Load library and source code

Hide

```
if (!require("devtools")) install.packages("devtools")
#if (!require("pacman")) {
  ## devtools is required
#  library(devtools)
#  install_github("trinker/pacman")
#}
#pacman::p_load(knitr, readr, stringr, tesseract, vecsets)
source('../lib/ifCleanToken.R')
library(ngram)
library(NLP)
library(tm)
library(purrr)
#library(qdap)
library(knitr)
source("../lib/bigramize.R")
source("../lib/get_text.R")
```

2 Step 2 - read the files and conduct Tesseract OCR

Although we have processed the Tesseract OCR and save the output txt files in the `data` folder, we include this chunk of code in order to make clear the whole pipeline to you.

Hide

```
for(i in c(1:length(file_name_vec))){
  current_file_name <- sub(".txt","",file_name_vec[i])
  ## png folder is not provided on github (the code is only on demonstration purpose)
  current_tesseract_txt <- tesseract::ocr(paste("../data/png/",current_file_name,".png",sep=""
))

  ### clean the tessetact text (separate line by "\n", delete null string, transter to lower ca
se)
  clean_tesseract_txt <- strsplit(current_tesseract_txt,"\n")[[1]]
  clean_tesseract_txt <- clean_tesseract_txt[clean_tesseract_txt!=""]

  ### save tesseract text file
  writeLines(clean_tesseract_txt, paste("../data/tesseract/",current_file_name,".txt",sep=""))
}
```

3 Step 3 - Error detection

Now, we are ready to conduct post-processing, based on the Tessearct OCR output. First of all, we need to detect errors, or *incorrectly processed words* – check to see if an input string is a valid dictionary word or if its n-grams are all legal.

The referenced papers are:

1. Rule-based techniques (http://webpages.ursinus.edu/akontostathis/KulpKontostathisFinal.pdf)

- rules are in the section 2.2

2. Letter n-gram (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1672564)

- focus on positional binary digram in section 3-a.error detection

3. Probabilistic techniques – SVM garbage detection (https://dl.acm.org/citation.cfm?doid=2034617.2034626)

- features are in section 5 (you can choose not to implement â□□Levenshtein distanceâ□□ feature)

We implement the letter n-gram in our project in paper 2.
The detection method is positional binary n-gram and correction method is topic model. Error detection is performed by comparing all possible bigrams of all words in Tesseract with positional binary matrix constructed from Ground Truth. If any bigrams of a word do not appear in the corresponding positional binary matrix then the word is classified as error.

Hide

```
# We saved the results and load it to avoid running all the codes repeatedly.
load("../output/ground_truth_by_length.Rdata")
load("../output/tesseract_words_cleaned.Rdata")
load("../output/tesseract_labels.Rdata")
load("../output/ground_truth_list.Rdata")
load("../output/detected_words.Rdata")
load("../output/tesseract_labels.Rdata")
```

Hide

```r
#### ground truth ##################################################################
### read files
ground_truth_dir <-"../data/ground_truth"
ground_truth_file_name <- list.files(ground_truth_dir)
ground_truth_file_path <- paste0(ground_truth_dir,'/',ground_truth_file_name)
ground_truth_onelines <- lapply(ground_truth_file_path,get_text)
### clean the texts and sort the words by letter length
ground_truth_words_cleaned <- unlist(lapply(ground_truth_onelines,clean_text))
ground_truth_by_length <- sort_by_length(ground_truth_words_cleaned)
save(ground_truth_by_length, file = "../output/ground_truth_by_length.Rdata")
### bigramize the word from word length >= 3
num_len <- length(ground_truth_by_length)
ground_truth_bigram_from3 <- lapply(ground_truth_by_length[3:num_len],bigramize)
save(ground_truth_bigram_from3,file="../output/ground_truth_bigram_from3.Rdata")
# load("../output/ground_truth_bigram_from3.Rdata")
### include word length <= 2
ground_truth_bigram_from3$l_1$PD_1_1 <- letters
ground_truth_bigram_from3$l_2$PD_1_2 <- unique(ground_truth_by_length[[2]])
ground_truth_bigram_from3 <- ground_truth_bigram_from3[c(29,30,1:18)]
### romove duplicate bigrams
final_dictionary <- lapply(ground_truth_bigram_from3,unique_bigram)
save(final_dictionary,file="../output/dictionary.Rdata") ### this is the final dictionary for e
rror detection
#load("../output/dictionary.Rdata")
#### tesseract ####################################################################
### read files
tesseract_dir <- "../data/tesseract"
tesseract_file_name <- list.files(tesseract_dir)
tesseract_file_path <- paste0(tesseract_dir,'/',tesseract_file_name)
tesseract_onelines <- lapply(tesseract_file_path,get_text)
### clean the texts
tesseract_words_cleaned <- lapply(tesseract_onelines,clean_text)
save(tesseract_words_cleaned,file="../output/tesseract_words_cleaned.Rdata")
# load("../output/ground_truth_words_cleaned.Rdata")

#### error detection ##############################################################
tesseract_labels <- list()
for (i in 1:length(tesseract_words_cleaned)) {
  cat("current file number =",i,"\n")
  cur_tesseract <- tesseract_words_cleaned[[i]]
  # cur_file <- tesseract_words_cleaned[[i]]
  cur_labels <- c()
  for (j in 1:length(cur_tesseract)) {
    cur_word <- cur_tesseract[j]
    cur_length <- nchar(cur_word)
    cur_bigram <- bigramize_word(cur_word)
    cur_dic <- final_dictionary[[paste("l_",cur_length,sep="")]]
    tmp_lable <- c()
    # browser()
    for (k in 1:length(cur_bigram)){
      tmp_lable[k] <- cur_bigram[[k]]%in%cur_dic[[k]]
    }
    cur_tmp_lable <- as.numeric(sum(tmp_lable)==length(cur_bigram))
    cur_labels[j] <- cur_tmp_lable
  }
  tesseract_labels[[i]] <- cur_labels
}
error_rate <- 1-(sapply(tesseract_labels, sum)/sapply(tesseract_words_cleaned, length))
summary(error_rate)
save(tesseract_labels,file="../output/tesseract_labels.Rdata")
### save the error words
detected_words <- list()
for (i in 1:100){
  detected_words[[i]] <- tesseract_words_cleaned[[i]][!as.logical(tesseract_labels[[i]])]
}
save(detected_words,file="../output/detected_words.Rdata")
```

4 Step 4 - Error correction

Given the detected word error, in order to find the best correction, we need to generating the candidate corrections: a dictionary or a database of legal n-grams to locate one or more potential correction terms. Then we need invoke some lexical-similarity measure between the misspelled string and the candidates or a probabilistic estimate of the likelihood of the correction to rank order the candidates.

The referenced papers are:

1. Letter n-gram (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1672564%7D%7Bpositional%20binary%20digram)

- focus on section 3-b.error correction

2. Supervised model – correction regressor (https://arxiv.org/pdf/1611.06950.pdf)
3. probability scoring without context (https://link.springer.com/content/pdf/10.1007%2FBF01889984.pdf)

- focus on section 3

4. probability scoring with contextual constraints (https://link.springer.com/content/pdf/10.1007%2FBF01889984.pdf)

- focus on section 5

5. topic models (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4377099)

Here, in our project, we use the topic models. The error correction algorithm consists of a topic model that provides information about word probabilities. We list all the candidates of an error word and calculate the probability of each candidate based on the topic model. The candidate with the highest probability will be chosen as the correct word.

Hide

```
library(tm)
files <- lapply(ground_truth_file_path,readLines)
docs <- Corpus(VectorSource(files))
load("../output/ground_truth_by_length.Rdata")
dic <- ground_truth_by_length
source("../lib/sub_prob.R")
prob <- sub_prob()
source("../lib/LDAmodel.R")
words_pro <- LDA_MODEL(docs)
words_topic <- names(words_pro)

load("../output/detected_words.Rdata")
detected_words_vec <- lapply(detected_words,unique)
source("../lib/Candidate.R")


results <- list()
for (text in 1:100){
  len <- length(detected_words_vec[[text]])
  result <- rep(0, len)
  for (i in 1:len){
    result[i] <- Candidate_best(detected_words_vec[[text]][i])$best
  }
  results[[text]] <- result
}
```

Hide

```
load("../output/ground_truth_by_length.Rdata")
load("../output/results.Rdata")
```

5 Step 5 - Performance measure

The two most common OCR accuracy measures are precision and recall. Both are relative measures of the OCR accuracy because they are computed as ratios of the correct output to the total output (precision) or input (recall). More formally defined,

$$\text{precision} = \frac{\text{number of correct items}}{\text{number of items in OCR output}}$$
$$\text{recall} = \frac{\text{number of correct items}}{\text{number of items in ground truth}}$$

where *items* refer to either characters or words, and ground truth is the original text stored in the plain text file.

Both *precision* and *recall* are mathematically convenient measures because their numeric values are some decimal fractions in the range between 0.0 and 1.0, and thus can be written as percentages. For instance, recall is the percentage of words in the original text correctly found by the OCR engine, whereas precision is the percentage of correctly found words with respect to the total word count of the OCR output. Note that in the OCR-related literature, the term OCR accuracy often refers to recall.

Hide

```
### delete error words in tesseract
load("../output/tesseract_words_cleaned.Rdata")
load("../output/tesseract_labels.Rdata")
load("../output/ground_truth_list.Rdata")

split_char <- function(vec){length(unlist(strsplit(vec, "")))}
tesseract_delete_error <- list()
for (i in 1:100){
  tesseract_delete_error[[i]] <- tesseract_words_cleaned[[i]][as.logical(tesseract_labels
[[i]])]
}
# save(tesseract_delete_error,file="../output/tesseract_delete_error.Rdata")
### performance table -- mean of 100 documents
### wordwise
#ground_truth_list <- lapply(ground_truth_onelines,clean_text)
tesseract_list <- tesseract_words_cleaned
tesseract_delete_error_list <- tesseract_delete_error
old_intersect_list <- list()
for (i in 1:100){
  old_intersect_list[[i]] <- vecsets::vintersect(tolower(ground_truth_list[[i]]), tolower(tesse
ract_list[[i]]))
}
new_intersect_list <- list()
for (i in 1:100){
  new_intersect_list[[i]] <- vecsets::vintersect(tolower(ground_truth_list[[i]]), tolower(tesse
ract_delete_error_list[[i]]))
}
#correction
tesseract_vec_all_add_correct <- list()
correct_intersect_list <- list()
for (i in 1:100){
  tesseract_vec_all_add_correct[[i]] <- append(tolower(tesseract_delete_error_list[[i]]), resul
ts[[i]])
  correct_intersect_list[[i]] <- vecsets::vintersect(tolower(ground_truth_list[[i]]), tolower(t
esseract_vec_all_add_correct[[i]]))
}
#
OCR_performance_table <- data.frame("Tesseract" = rep(NA,4),
                                    "Tesseract_with_postprocessing" = rep(NA,4),
                                    "Tesseract_with_correction" = rep(NA,4))
row.names(OCR_performance_table) <- c("word_wise_recall","word_wise_precision",
                                      "character_wise_recall","character_wise_precision")
word_wise_recall_tesseract <- rep(0,100)
word_wise_precision_tesseract <- rep(0,100)
word_wise_recall_postprocessing <- rep(0,100)
word_wise_precision_postprocessing <- rep(0,100)
character_wise_recall_tesseract<-rep(0,100)
character_wise_precision_tesseract<-rep(0,100)
character_wise_recall_postprocessing<-rep(0,100)
character_wise_precision_postprocessing<-rep(0,100)
for (i in 1:100){
  word_wise_recall_tesseract[i] <- length(old_intersect_list[[i]])/length(ground_truth_list
[[i]])
```

```
  word_wise_precision_tesseract[i] <- length(old_intersect_list[[i]])/length(tesseract_list
[[i]])
  word_wise_recall_postprocessing[i] <- length(new_intersect_list[[i]])/length(ground_truth_lis
t[[i]])
  word_wise_precision_postprocessing[i] <- length(new_intersect_list[[i]])/length(tesseract_del
ete_error_list[[i]])

  character_wise_recall_tesseract[i]<-split_char(old_intersect_list[[i]])/split_char(ground_tru
th_list[[i]])
  character_wise_precision_tesseract<-split_char(old_intersect_list[[i]])/split_char(tesseract_
list[[i]])
  character_wise_recall_postprocessing<-split_char(new_intersect_list[[i]])/split_char(ground_t
ruth_list[[i]])
  character_wise_precision_postprocessing<-split_char(new_intersect_list[[i]])/split_char(tesse
ract_delete_error_list[[i]])
}
#correction
word_wise_recall_correction <- rep(0,100)
word_wise_precision_correction <- rep(0,100)
character_wise_recall_correction <- rep(0,100)
character_wise_precision_correction <- rep(0,100)
for (i in 1:100){
  word_wise_recall_correction[i] <- length(correct_intersect_list[[i]])/length(ground_truth_lis
t[[i]])
  word_wise_precision_correction[i] <- length(correct_intersect_list[[i]])/length(tesseract_vec
_all_add_correct[[i]])
  character_wise_recall_correction[i] <- split_char(correct_intersect_list[[i]])/split_char(gro
und_truth_list[[i]])
  character_wise_precision_correction[i] <- split_char(correct_intersect_list[[i]])/split_char
(tesseract_vec_all_add_correct[[i]])
}
#
OCR_performance_table["word_wise_recall","Tesseract"] <- mean(word_wise_recall_tesseract)
OCR_performance_table["word_wise_precision","Tesseract"] <- mean(word_wise_precision_tesseract)
OCR_performance_table["word_wise_recall","Tesseract_with_postprocessing"] <- mean(word_wise_rec
all_postprocessing)
OCR_performance_table["word_wise_precision","Tesseract_with_postprocessing"] <- mean(word_wise_
precision_postprocessing)
OCR_performance_table["character_wise_recall","Tesseract"] <- mean(character_wise_recall_tesser
act)
OCR_performance_table["character_wise_precision","Tesseract"] <- mean(character_wise_precision_
tesseract)
OCR_performance_table["character_wise_recall","Tesseract_with_postprocessing"] <- mean(characte
r_wise_recall_postprocessing)
OCR_performance_table["character_wise_precision","Tesseract_with_postprocessing"] <- mean(chara
cter_wise_precision_postprocessing)
#correction
OCR_performance_table["word_wise_recall","Tesseract_with_correction"] <- mean(word_wise_recall_
correction)
OCR_performance_table["word_wise_precision","Tesseract_with_correction"] <- mean(word_wise_prec
ision_correction)
OCR_performance_table["character_wise_recall","Tesseract_with_correction"] <- mean(character_wi
se_recall_correction)
OCR_performance_table["character_wise_precision","Tesseract_with_correction"] <- mean(character
_wise_precision_correction)
#
kable(OCR_performance_table, caption="Summary of OCR performance")
```

Summary of OCR performance

	Tesseract	Tesseract_with_postprocessing	Tesseract_with_correction
word_wise_recall	0.6412554	0.6412255	0.6547511
word_wise_precision	0.6466806	0.7875722	0.7005428
character_wise_recall	0.5227591	0.4916339	0.5402035
character_wise_precision	0.5169737	0.7428246	0.6159443

We can see that after error detection, the ratio of correct words in all ground truth words or tesseract words is larger, which means we do detect the true error words. But the topic model doesn't do so well in the correction step. Since the ratios are not as high as the ratios after error detection, but they are larger than OCR. We think that is because LDA automatically process the words to its oringinal form and we can't correct the words to their other forms. For exapmle, when we detect "tltles" as an error word, we can't correct it to "titles" because in topic model we only have word "title" in the dictionary.

Besides the above required measurement, you are encouraged the explore more evaluation measurements. Here are some related references:

1. Karpinski, R., Lohani, D., & BelaÃ¯d, A. *Metrics for Complete Evaluation of OCR Performance*. pdf (https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/IPC3481.pdf)

- section 2.1 Text-to-Text evaluation

2. Mei, J., Islam, A., Wu, Y., Moh'd, A., & Milios, E. E. (2016). *Statistical learning for OCR text correction*. arXiv preprint arXiv:1611.06950. pdf (https://arxiv.org/pdf/1611.06950.pdf)

- section 5, separate the error detection and correction criterions

3. Belaid, A., & Pierron, L. (2001, December). *Generic approach for OCR performance evaluation*. In Document Recognition and Retrieval IX (Vol. 4670, pp. 203-216). International Society for Optics and Photonics. pdf (https://members.loria.fr/ABelaid/publis/spie02-belaid-pierron.pdf)

- section 3.2, consider the text alignment

References

1. Borovikov, E. (2014). *A survey of modern optical character recognition techniques*. arXiv preprint arXiv:1412.4183.pdf (https://pdfs.semanticscholar.org/79c9/cc90b8c2e2c9c54c3862935ea00df7dd56ed.pdf) (This paper is the source of our evaluation criterion)

2. Kukich, K. (1992). *Techniques for automatically correcting words in text*. Acm Computing Surveys (CSUR), 24(4), 377-439. pdf (http://www.unige.ch/eti/ptt/docs/kukich-92.pdf) (This paper is the benchmark review paper)