GR5243 PROJ 4

Hengyang Lin, Zhibo Zhou, Peilin Li, Ghada Jerfel, Xiaoyi Li

GU4243/GR5243: Applied Data Science

step0 : Load libraries

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 3.4.4
```

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.4.4
```

```
## Loading required package: NLP
```

```
## Warning: package 'NLP' was built under R version 3.4.4
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 3.4.4
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Step1: Compute Confusion probability matrix

Preprocess

0. Set locations

```
file_loc <- "../data/"
truth_file_loc <- paste0(file_loc,"ground_truth/")
tesseract_file_loc <- paste0(file_loc, "tesseract/")
doc_name <- list.files(path = truth_file_loc, pattern = "txt")
truth_doc_loc <- paste0(truth_file_loc, doc_name)
tesseract_doc_loc <- paste0(tesseract_file_loc, doc_name)
```

1. Make sure that corresponding documents have same number of lines. If not, adjust them manually.

```
source("../lib/preprocess.R")
truth_lineNumber_vec <- sapply(truth_doc_loc, get_number_lines)
tesseract_lineNumber_vec <- sapply(tesseract_doc_loc, get_number_lines)

## Get the names of documents which have different lines in truth and tesseract
doc_diffLines_names <- doc_name[truth_lineNumber_vec != tesseract_lineNumber_vec]

truth_lineNumber_vec[truth_lineNumber_vec != tesseract_lineNumber_vec]
```

```
## named integer(0)
```

```
tesseract_lineNumber_vec[truth_lineNumber_vec != tesseract_lineNumber_vec]
```

```
## named integer(0)
```

There are 0 files with different number of lines.

After manually adjusting, now all the correspoding documents have same number of lines. (There are 13 files needed to be adjusted in total.)

2. After breaking every lines down, we need those lines with same number of tokens between ground truth and tesseract.

The reason to do this for estimating confusion matrix, is that if OCR machine reads non-space character to space or vice versa, we are not able to pair tokens one-to-one, which makes the estimation very complicated.

Let's compute the fraction of these lines, because we cannot abandon too many lines.

```
## compute the fraction of lines which have the same number of tokens in truth and tesseract
truth_tokenNum_list <- list()
for(i in 1:length(doc_name)){
  doc <- doc_name[i]
  docloc <- paste0(truth_file_loc, doc)
  truetext_vec <- readLines(docloc, warn = FALSE, encoding = "UTF-8")
  truth_tokenNum_list[[i]] <- get_number_tokens(truetext_vec)
}
names(truth_tokenNum_list) <- doc_name

tesseract_tokenNum_list <- list()
for(i in 1:length(doc_name)){
  doc <- doc_name[i]
  docloc <- paste0(tesseract_file_loc, doc)
  testext_vec <- readLines(docloc, warn = FALSE, encoding = "UTF-8")
  tesseract_tokenNum_list[[i]] <- get_number_tokens(testext_vec)
}
names(tesseract_tokenNum_list) <- doc_name

sum_line <- sum(sapply(truth_tokenNum_list, length)) ## total number of lines

sum_sametokenline <- 0
for(i in 1:100){
  sum_sametokenline <- sum_sametokenline + sum(truth_tokenNum_list[[i]] == tesseract_tokenNum_list
[[i]])
}

sum_sametokenline/sum_line
```

```
## [1] 0.8867473
```

About 88.7% of lines have same numbers of tokens, which are going to be used. It’s acceptable.

3. Write those 88.7% lines with same number of tokens in two output files.

```r
## Get the logic which have the same number of tokens in truth and tesseract
log_tokNumEq_list <- list()
for(i in 1:length(doc_name)){
  log_tokNumEq_list[[i]] <- truth_tokenNum_list[[i]] == tesseract_tokenNum_list[[i]]
}
names(log_tokNumEq_list) <- doc_name

## Write those 88.7% lines with same number of tokens in output files.
output_loc <- "../output/"
for(i in 1:length(doc_name)){
  doc <- paste0(truth_file_loc, doc_name[i])
  text_vec_all <- readLines(doc, warn = FALSE, encoding = "UTF-8")
  ind_sameTKnum <- log_tokNumEq_list[[i]]
  text_vec <- text_vec_all[ind_sameTKnum]
  writeLines(text_vec, paste0(output_loc,"ground_truth/",doc_name[i]), useBytes = TRUE)
}

for(i in 1:length(doc_name)){
  doc <- paste0(tesseract_file_loc, doc_name[i])
  text_vec_all <- readLines(doc, warn = FALSE, encoding = "UTF-8")
  ind_sameTKnum <- log_tokNumEq_list[[i]]
  text_vec <- text_vec_all[ind_sameTKnum]
  writeLines(text_vec, paste0(output_loc,"tesseract/",doc_name[i]), useBytes = TRUE)
}
```

4. Write those corresponding tokens with same number of characters down in two output files.

After tokens were paired, we have to abandon those corresponding tokens with different length. The reason to this is we didn’t consider reading one letter to two or vice-versa in our confusion matrix. (The calculation of estimating error involving character alignment is very complicated.)

We need these non-space to non-space converting characters to estimate our confusion probability.

```r
## Write those tokens with same length down in output files.
truthline_loc <- paste0(output_loc, "ground_truth/")
tessline_loc <- paste0(output_loc, "tesseract/")

for(i in 1:length(doc_name)){
  truth_text <- readLines(paste0(truthline_loc, doc_name[i]), warn = FALSE, encoding = "UTF-8")
  tess_text <- readLines(paste0(tessline_loc, doc_name[i]), warn = FALSE, encoding = "UTF-8")
  truth_strsplit_list <- strsplit(truth_text, " ")
  tess_strsplit_list <- strsplit(tess_text, " ")
  truth_letternum_list <- lapply(truth_strsplit_list, nchar)
  tess_letternum_list <- lapply(tess_strsplit_list, nchar)

  truetk_text <- c()
  tesstk_text <- c()
  for(j in 1:length(truth_letternum_list)){
    ind_letternumEq <- truth_letternum_list[[j]] == tess_letternum_list[[j]]
    truetk_text <- c(truetk_text, paste0(truth_strsplit_list[[j]][ind_letternumEq], collapse = ""))
    tesstk_text <- c(tesstk_text, paste0(tess_strsplit_list[[j]][ind_letternumEq], collapse = ""))
  }
  writeLines(truetk_text, paste0(output_loc,"ground_truth_tk/",doc_name[i]), useBytes = TRUE)
  writeLines(tesstk_text, paste0(output_loc,"tesseract_tk/",doc_name[i]), useBytes = TRUE)
}
```

Computating and Saving confusion matrix

```
source("../lib/confusion_prob_matrix.R")
```

1. We care about A-Z, a-z, 0-9, and some common punctuations in our letterlist.

```
lowerletters <- c("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u",
"v","w","x","y","z")
upperletters <- toupper(lowerletters)
numberletters <- c("0","1","2","3","4","5","6","7","8","9")
punctuation <- c("-", ":", "\\*", "\\(", "\\)", "'", ",", "\\$")
letterlist <- c(lowerletters, upperletters, numberletters, punctuation)
d <- length(letterlist)
```

2. compute and save confusion number matrix. Rownames are printed letters, colnames are true letters.

```
ground_true_loc <- "../output/ground_truth_tk/"
tesseract_loc <- "../output/tesseract_tk/"
```

```
source("../lib/confusion_prob_matrix.R")
confusion_num_mat <- confusion_num_matrix(ground_true_loc,tesseract_loc)
rownames(confusion_num_mat) <- letterlist
colnames(confusion_num_mat) <- letterlist
save(confusion_num_mat, file = "../output/confusion_number_matrix.Rdata")
```

3. compute and save the numbers of true letters in letterlist.

```
trueletter_num_vec <- rep(0, d)
for(k in 1:d){
  trueletter_num_vec[k] <- compute_num_trueletter(letterlist[k], ground_true_loc)
}
names(trueletter_num_vec) <- letterlist
save(trueletter_num_vec, file = "../output/trueletter_num_vec.Rdata")
```

4. see how many letters we are going to use to estimate the confusion probability.

We compute the ratio of letters used in estimation over total true letters in our selected tokens. As a result, we can see almost all alphabets are used. However, some punctuations are not fully used.

```
load("../output/confusion_number_matrix.Rdata")
load("../output/trueletter_num_vec.Rdata")
apply(confusion_num_mat, 2, sum)/trueletter_num_vec
```

```
##          a          b          c          d          e          f 
## 0.99502834 0.99887320 0.99958372 0.99923554 0.99944395 0.99377757 
##          g          h          i          j          k          l 
## 0.99945455 0.99988264 0.99997821 1.00000000 0.99886673 0.99961665 
##          m          n          o          p          q          r 
## 0.99975735 0.99953390 0.99927586 0.99963862 1.00000000 0.99942142 
##          s          t          u          v          w          x 
## 0.99835614 0.99937916 0.99985259 0.99985652 0.99916369 0.99923567 
##          y          z          A          B          C          D 
## 0.99809469 0.99937578 0.99849089 0.99702204 0.99904681 0.99806051 
##          E          F          G          H          I          J 
## 0.99981329 0.99941418 0.99745061 0.98798558 1.00000000 0.99689441 
##          K          L          M          N          O          P 
## 0.99348534 0.99814011 0.99934426 0.99923810 0.99966865 0.99297789 
##          Q          R          S          T          U          V 
## 1.00000000 0.99430789 0.99480563 0.99961499 0.99065421 1.00000000 
##          W          X          Y          Z          0          1 
## 0.99846626 0.99590164 1.00000000 1.00000000 0.99694283 0.99847251 
##          2          3          4          5          6          7 
## 0.99720358 0.99544419 0.99493243 0.99773414 0.99050633 0.99831555 
##          8          9          -          :        \\*        \\( 
## 0.99320017 0.99702381 0.90243902 0.99391172 0.05769231 0.22343921 
##        \\)          '          ,        \\$ 
## 0.75675676 0.99493813 0.86837897 0.37461300
```

5. correction: Add 0.5 at zero entries of the confusion number matrix as correction.

```
confusion_numcor_mat <- confusion_num_mat
for(i in 1:d){
  for(j in 1:d){
    if(confusion_num_mat[i,j] == 0){
      confusion_numcor_mat[i,j] <- 0.5
    }
  }
}
```

6. Use corrected confusion number matrix and numbers of true letters to compute confusion probability matrix and save it.

```
confusion_prob_mat <- confusion_numcor_mat
for(i in 1:d){
  for(j in 1:d){
    confusion_prob_mat[i,j] <- confusion_numcor_mat[i,j]/trueletter_num_vec[j]
  }
}
save(confusion_prob_mat, file = "../output/confusion_probability_matrix.Rdata")
```

Step2: Detect Typo word

we use a rule-based detection. The ouput is a list with 100 elements which are vectors.

Each element represents a document, while the values of the vector are TRUE or FALSE (Garbage -> True; Not Garbage -> False), and the names of the vector are tokens of the document.

```
source("../lib/detection.R")

detection_list <- list()
for(i in 1:length(doc_name)){
  ocr_fileloc <- paste0(tesseract_file_loc, doc_name[i])
  detection_list[[i]] <- detect_file(ocr_fileloc)
}
names(detection_list) <- doc_name
save(detection_list, file = "../output/detection_list.Rdata")
```

Let’s have a view at the second element of our ouput, which is the detection result of file “group1_00000010.txt”

(TRUE indicates that the word is garbage; FALSE indicates that the word is clean.)

```
load("../output/detection_list.Rdata")
head(detection_list$group1_00000010.txt, 50)
```

```
##     Proposed      esearch      Program            0         Met:
##        FALSE        FALSE        FALSE        FALSE        FALSE
##        nlsms           of     Phosgene       Injury           to
##         TRUE        FALSE        FALSE        FALSE        FALSE
##          the        Lungs         EXHI            T            A
##        FALSE        FALSE        FALSE        FALSE        FALSE
##        Puzpo          The           pu          ass           of
##        FALSE        FALSE        FALSE        FALSE        FALSE
##         thls     program,           to           be collectlvely
##         TRUE        FALSE        FALSE        FALSE         TRUE
##       funded           by    Voluntary         sub?      scrlptl
##        FALSE         TRUE        FALSE        FALSE         TRUE
##           n5         from   1nterested    companles,           15
##        FALSE        FALSE         TRUE         TRUE        FALSE
##           to      provlde   1nfomatlon           on          the
##        FALSE         TRUE         TRUE        FALSE        FALSE
##        baslc      mechanl            s     Involved           1n
##         TRUE         TRUE        FALSE        FALSE         TRUE
##     phosgene       damage           to          the       lungs.
##        FALSE        FALSE        FALSE        FALSE        FALSE
```

Step3: Correction Process

Preparation

0. Get and save a dictionary for us to generate correct candidate. Here we use ground truth files as a dictionary.

```
source("../lib/lexicon.R")
dictionary_loc <- truth_file_loc
#lexicon <- get_dictionary(dictionary_loc)
lexicon <- get_trueitems(dictionary_loc)
save(lexicon, file = "../output/lexicon.Rdata")
```

1. Get and save a model list consists of 100 LDA models. Set number of topics to 5.

Each LDA model is trained by ground truth documents other than a specific document doc_i. Then this model is used to compute scores when we are correcting the corresponding tesseract of the specific document doc_i.

```
n_topics <- 5
source("../lib/LDAlist.R")
lda_list <- get_ldamodels(truth_file_loc, n_topics)
save(lda_list, file = "../output/lda_list.Rdata")
```

2. Load confusion probability matrix, detection result list lexicon and LDA model list as well as letterlist.

```
load("../output/confusion_probability_matrix.Rdata")
load("../output/detection_list.Rdata")
load("../output/lexicon.Rdata")
load("../output/lda_list.Rdata")

lowerletters <- c("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u",
"v","w","x","y","z")
upperletters <- toupper(lowerletters)
numberletters <- c("0","1","2","3","4","5","6","7","8","9")
punctuation <- c("-", ":", "\\*", "\\(", "\\)", "'", ",", "\\$")
letterlist <- c(lowerletters, upperletters, numberletters, punctuation)
d <- length(letterlist)
```

Correction

```
source("../lib/lowercandidate_generate.R")
source("../lib/correction.R")
```

1. First, let’s see how good it is when the function corrects a single typo word.

Try to correct typoword “posltlon” in file “group1_00000005.txt”.

Candidates are:

```
lowercandidate_generate("posltlon", lexicon)
```

```
## [1] "posltlon" "position"
```

Correction is:

```
correct_word("posltlon", "group1_00000005.txt")
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## [1] "position"
```

This function is able to correct upper letter. Try to correct typoword “WIllIam” in file “group1_00000005.txt”.

Candidates are: (here candidates are all lowercase letters because we need lowercase words to compute topicmodel score.)

```
lowercandidate_generate("Wllllam", lexicon)
```

```
## [1] "wllllam" "william"
```

Correction is: (here correction contains an uppercase letter W)

```
correct_word("Wllllam", "group1_00000005.txt")
```

```
## [1] "William"
```

Try to correct typoword “prohlblt:”, which contains a punctuation, in file “group1_00000005.txt”. The true token is “prohibit:”.

Candidates are: (here candidates are with right punctuation. The punctuations would be removed during computing topicmodel score.)

```
lowercandidate_generate("prohlblt:", lexicon)
```

```
## [1] "prohlblt:" "prohibit:"
```

Correction is: (here correction contains right punctuation “:”)

```
correct_word("prohlblt:", "group1_00000005.txt")
```

```
## [1] "prohibit:"
```

2. Seconed, let’s try to correct a single tesseract file. We try to correct file “group1_00000010.txt” since it’s the shortest to be presented here.

Correct text “group1_00000010.txt” and save the processed text.

```
# correct tesseract file : "group1_00000010.txt"
doc2 <- "group1_00000010.txt"
correction_2 <- correct_file(doc2, detection_list)
writeLines(correction_2, paste0(file_loc,"prediction/",doc2), useBytes = TRUE)
```

Show processed text:

```
doc2 <- "group1_00000010.txt"
text <- readLines(paste0(file_loc,"prediction/",doc2), warn = FALSE, encoding = "UTF-8")
cat(text)
```

```
## Proposed esearch Program 0 Met: nlsms of Phosgene Injury to the Lungs EXHI T A Puzpo The pu ass of
this program, to be collectively funded by Voluntary sub? scrlptl n5 from interested companies, 15 to
provide 1nfomatlon on the basic mechanl s Involved in phosgene damage to the lungs. Th1: information 1
5 a prerequl itS for the detemlnatlon of a recommended method of medical treat? ment of 1 dustrlal exp
osures to phosgene. Initial p oject As recomme dad by the Ad Hoc Task Group on Phosgene Safety compose
d of represents was of interested member companies, the lnMediate project 15 that defined in he propos
al made by Dr. M OF Frosolono of the.Mt. SUm Hospital of Clevelan in 3 letter to MCA dated August 23,
1972 Th1: project Is to be conducted un er Contracts (37 between MCA and Mt. SUm Hospital of Clevelan
d, not to excee $30,00 and (by between MCA and Dr. Frosolono for consulting fees regarding the researc
h referred to in 157, not to exceed $3,00 Supervision Th1: research 5 m be supervised on MCA'S behalf
by a Technical Task Group composed of on technical representative from each subscribing company plus a
n MCA staff repr tentative to serve as secretary. Drganlzatlonall Within MCA, the Task Group 15 to fun
ction under the Jurlsi diction of the afety and Fire Protection Committee Funding Consnltments hav bee
n made by thirteen interested companies in support of this research, u to $3,50 per company. Funds are
to be collected in advance of commitments d disbursements subject to MCA staff clearance and approval.
Stlpulatlons The sclentlflc in rmation developed from this program 15 m be fully d15* closed without r
e trlctlon For purposes of consistency and control, however, all releases are t be made through MCA on
ly, notably based on Task Force recomi mendations as m n ture and timing Th1: requlrement 15 not to pr
eclude pay vats sclentlflc In erahange of non'competltlve information which Will be useful, nor releva
nt prlvat cosnunlcatlon among subscribing companies and organizations the staff represent tive m be 1n
fomed of all such interchanges and communications Operation The Task Group will a governed by MCA'S "G
eneral Principles Applicable to the Structure and op Iations of Committees and the Rules of Drganlzatl
on and Procedure of the 556 ty and Fire Protection Committee MCA CMA 035207
```

Show tesseract text:

```
text <- readLines(paste0(file_loc,"tesseract/",doc2), warn = FALSE, encoding = "UTF-8")
cat(text)
```

```
## Proposed esearch Program 0 Met: nlsms of Phosgene Injury to the Lungs EXHI T A Puzpo The pu ass of
thls program, to be collectlvely funded by Voluntary sub? scrlptl n5 from 1nterested companles, 15 to
provlde 1nfomatlon on the baslc mechanl s Involved 1n phosgene damage to the lungs. Th1: 1nformatlon 1
5 a prerequl 1te for the detemlnatlon of a recommended method of medlcal treat? ment of 1 dustrlal exp
osures to phosgene. Inltlal p oject As recomme dad by the Ad Hoc Task Group on Phosgene Safety compose
d of represents was of 1nterested member companles, the lnaedlate project 15 that deflned 1n he propos
al made by Dr. M. F, Frosolono of the.Mt. 5mm Hospltal of Clevelan 1n 3 letter to MCA dated August 23,
1972. Th1: project Is to be conducted un er Contracts (37 between MCA and Mt. 5mm Hospltal of Clevelan
d, not to excee $30,000 and (by between MCA and Dr. Frosolono for consultlng fees regardlng the resear
ch referred to 1n 157, not to exceed $3,000. Supervlslon Th1: research 5 m be supervlsed on MCA'S beha
lf by a Technlcal Task Group composed of on technlcal representatlve from each subscrlblng company plu
s an MCA staff repr sentatlve to serve as secretary. Drganlzatlonall Within MCA, the Task Group 15 to
functlon under the Jurlsi diction of the afety and Fire Protectlon Comnlttee. Fundlng Consnltments hav
been made by thlrteen 1nterested companles 1n support of thls research, u to $3,500 per company. Funds
are to be collected 1n advance of commltments. d dlsbursements subject to MCA staff clearance and appr
oval. Stlpulatlons The sclentlflc 1n omatlon developed from thls program 15 m be fully d15* closed wlt
hout re trlctlon. For purposes of conslstency and control, however, all releases are t be made through
MCA only, nomally based on Task Force recomi mendatlons as m n ture and tlmlng. Th1: requlrement 15 no
t to preclude pry vats sclentlflc In erahange of non'competltlve 1nformatlon whlch Will be useful, nor
relevant prlvat cosnunlcatlon among subscrlblng companles and organlzatlons; the staff represent tlve
m be 1nfomed of all such 1nterchanges and communlcatlons. Operatlon The Task Group Hill a governed by
MCA'S "General Prlnclples Appllcable to the Structure and op ratlons of Conmlttees" and the Rules of D
rganlzatlon and Procedure of the 556 ty and Fire Protectlon Commlttee. MCA CMA 035207
```

Show ground truth text:

```
text <- readLines(paste0(file_loc,"ground_truth/",doc2), warn = FALSE, encoding = "UTF-8")
cat(text)
```

```
## Proposed Research Program on Mechanisms of Phosgene Injury to the Lungs EXHIBIT A Purpose The purpo
se of this program, to be collectively funded by voluntary sub- scriptions from interested companies,
is to provide information on the basic mechanisms Involved in phosgene damage to the lungs. This infor
mation is a prerequisite for the determination of a reconmended method of medical treat- ment of indus
trial exposures to phosgene. Initial Project As recommended by the Ad Hoc Task Group on Phosgene Safet
y composed of representatives of interested member companies, the lnaediate project is that defined in
the proposal made by Or. M. F, Frosolono of the.Mt. Sinai Hospital of Cleveland in a letter to MCA dat
ed August 23, 1972. This project Is to be conducted under contracts (a) between MCA and Mt. Sinai Hosp
ital of Cleveland, not to exceed $30,000 and (b) between MCA and Dr. Frosolono for consulting fees reg
arding the research referred to in (a), not to exceed $3,000. Supervision This research is to be super
vised on MCA's behalf by a Technical Task Group composed of one technical representative from each sub
scribing company plus an MCA staff representative to serve as secretary. Organizationally within MCA,
the Task Group is to function under the Juris- diction of the Safety and Fire Protection Comnittee. Fu
nding Consnitments have been made by thirteen interested companies in support of this research, up to
$3,500 per company. Funds are to be collected in advance of commitments, and disbursements subject to
MCA staff clearance and approval. Stipulations The scientific information developed from this program
is to be fully dis- closed without restriction. For purposes of consistency and control, however, all
releases are to be made through MCA only, normally based on Task Force recom- mendations as to nature
and timing. This requirement is not to preclude pri- vate scientific Interchange of non'competitive in
formation which will be useful, nor relevant private cosnunlcation among subscribing companies and org
anizations; the staff representative to be informed of all such interchanges and communications. Opera
tion The Task Group will be governed by MCA's "General Principles Applicable to the Structure and Oper
ations of Conmittees" and the Rules of Organization and Procedure of the Safety and Fire Protection Co
mmittee. MCA CMA 036207
```

As we can see in the above, the first correction happened at second line ,where “1nterested” and “companles” were corrected as “interested” and “companies”.

We can also find out that this method helps nothing at many typo words, since it could not correct tokens when number of characters changes nor three or more letters change.

Step4: Evaluate

To save time, we use first 10 files to measure our correction performance.

word-wise evaluation

Since our algorithm takes upper case or lower case into consideration, and is able to correct them, we didn’t convert text to lower case when we calculating recall or precision.

So our recall and precision are more “strict” than those in starter codes.

1. Process first 10 files. Write down the prediction text in “../data/prediction” file.

```
load("../output/detection_list.Rdata")
prediction_loc <- "../data/prediction/"

docname_toeval <- doc_name[1:10]
for(i in 1:length(docname_toeval)){
  processed_text <- correct_file(docname_toeval[i], detection_list)
  writeLines(processed_text, paste0(prediction_loc, docname_toeval[i]), useBytes = TRUE)
}
```

2. Visualization

```
source("../lib/evaluation.R")
word_ground_truth_loc <- "../data/ground_truth/"
word_prediction_loc <- "../data/prediction/"
word_tesseract_loc <- "../data/tesseract/"

word_performance_table <- data.frame("Before_Processing" = rep(NA,2),
                                     "After_Processing" = rep(NA,2))
row.names(word_performance_table) <- c("word_wise_recall", "word_wise_precision")

word_performance_table["word_wise_recall", "Before_Processing"] <- recall_words(word_ground_truth_loc,
 word_tesseract_loc)
word_performance_table["word_wise_recall", "After_Processing"] <- recall_words(word_ground_truth_loc,
 word_prediction_loc)
word_performance_table["word_wise_precision", "Before_Processing"] <-
  precision_words(word_ground_truth_loc, word_tesseract_loc, word_tesseract_loc)
word_performance_table["word_wise_precision", "After_Processing"] <-
  precision_words(word_ground_truth_loc, word_prediction_loc, word_tesseract_loc)

knitr::kable(word_performance_table, caption="OCR performance: word-wise evaluation")
```

OCR performance: word-wise evaluation

	Before_Processing	After_Processing
word_wise_recall	0.6137132	0.8101513
word_wise_precision	0.6056989	0.7950108

Character-wise evaluation

Our correction function cannot help if OCR machine “breaks” one token as two or “combines” two tokens as one during reading. Though word-wise evaluation could still be easily implemented, **because we don’t want to ignore the order of character, character-wise evaluation is very complicated if breaking or combining happens.** Under this situation, we need to use lines with same number of tokens as tesseract data to evaluate our correction character-wisely.

Lines with same number of tokens have been written down at “../output/tesseract” file in Step1 – preprocess.

0. Run our detection algorithm on tesseract files at output file. Save the result of detection.

```
tesseract_loc_eval <- "../output/tesseract/"

source("../lib/detection.R")

detection_list_eval <- list()
for(i in 1:length(doc_name)){
  ocr_fileloc_eval <- paste0(tesseract_loc_eval, doc_name[i])
  detection_list_eval[[i]] <- detect_file(ocr_fileloc_eval)
}
names(detection_list_eval) <- doc_name
save(detection_list_eval, file = "../output/detection_list_eval.Rdata")
```

1. Run correction algorithm on first 10 tesseract files of output file. Write down the corrected text in prediction doc of output.

```
prediction_loc_eval <- "../output/prediction/"
load("../output/detection_list_eval.Rdata")

docname_toeval <- doc_name[1:10]
for(i in 1:length(docname_toeval)){
  processed_text <- correct_file(docname_toeval[i], detection_list_eval)
  writeLines(processed_text, paste0(prediction_loc_eval, docname_toeval[i]), useBytes = TRUE)
}
```

2. Visualization

```
source("../lib/evaluation.R")

char_ground_truth_loc <- "../output/ground_truth/"
char_prediction_loc <- "../output/prediction/"
char_tesseract_loc <- "../output/tesseract/"

char_performance_table <- data.frame("Before_Processing" = rep(NA,2),
                                     "After_Processing" = rep(NA,2))
row.names(char_performance_table) <- c("character_wise_recall", "character_wise_precision")

char_performance_table["character_wise_recall", "Before_Processing"] <- recall_chars(char_ground_truth
_loc, char_tesseract_loc)
char_performance_table["character_wise_recall", "After_Processing"] <- recall_chars(char_ground_truth_
loc, char_prediction_loc)
char_performance_table["character_wise_precision", "Before_Processing"] <-
  precision_chars(char_ground_truth_loc, char_tesseract_loc, char_tesseract_loc)
char_performance_table["character_wise_precision", "After_Processing"] <-
  precision_chars(char_ground_truth_loc, char_prediction_loc, char_tesseract_loc)

knitr::kable(char_performance_table, caption="OCR performance: character-wise evaluation")
```

OCR performance: character-wise evaluation

	Before_Processing	After_Processing
character_wise_recall	0.8875897	0.9321851
character_wise_precision	0.8986381	0.9453177

Remember that we omitted those lines in which OCR machine breaking or combining tokens during reading happened, in another word we didn’t consider the error that OCR machine reads a non-space character to space or vice-versa. **So here character-wise precision and recall are both overestimated.**

Evaluation

```
performance_table <- rbind(word_performance_table,char_performance_table)
knitr::kable(performance_table, caption="OCR performance evaluation")
```

OCR performance evaluation

	Before_Processing	After_Processing
word_wise_recall	0.6137132	0.8101513

	Before_Processing	After_Processing
word_wise_precision	0.6056989	0.7950108
character_wise_recall	0.8875897	0.9321851
character_wise_precision	0.8986381	0.9453177

Step5: Use iteration to improve

As stated in the paper, we could use iteration to improve prediction. If a file was processed, then the processed file contains more correct words, which help to improve the topic model part, which means we could perform a new round of detection and correction on the processed file.

This could be easily implemented in our project if we just regard file prediction as file tesseract.

To save time, we perform both word-wise and character-wise evaluation on data of lines with same number of tokens, which are contained in outpu file. **Under this situation, the word-wise recall would be equal to word-wise precision.**

We would not run this part because it takes too much time. Besides, according to experiments on several files, iteration here helps quite little.

```
load("../output/confusion_probability_matrix.Rdata")
load("../output/lexicon.Rdata")

extra <- 1 #One extra iterations
pred_file_loc <- "../output/prediction/"

pred_file_name <- list.files(pred_file_loc)

performance_mat <- matrix(NA, nrow = 4, ncol = 2 + extra)
col_names <- paste0("After_",0:(extra+1),"_times_processing")

performance_table <- data.frame(performance_mat)
colnames(performance_table) <- col_names
row.names(performance_table) <- c("word_wise_recall", "word_wise_precision",
                                  "character_wise_recall", "character_wise_precision")

performance_table[1:2, 1:2] <- word_performance_table
performance_table[3:4, 1:2] <- char_performance_table

for(I in 1:extra){
  #detection
  cur_detection_list <- list()
  for(i in 1:length(pred_file_name)){
    ocr_fileloc <- paste0(pred_file_loc, pred_file_name[i])
    cur_detection_list[[i]] <- detect_file(ocr_fileloc)
  }
  names(cur_detection_list) <- pred_file_name
  #correction
  for(i in 1:length(pred_file_name)){
    processed_text <- correct_file(pred_file_name[i], cur_detection_list)
    writeLines(processed_text, paste0(pred_file_loc, pred_file_name[i]), useBytes = TRUE)
  }
  #evaluation
  performance_table["word_wise_recall",I + 2] <- recall_words("../output/ground_truth/", pred_file_lo
c)
  performance_table["word_wise_precision",I + 2] <- precision_words("../output/ground_truth/", pred_fi
le_loc, "../output/tesseract/")
  performance_table["character_wise_recall", I + 2] <- recall_chars("../output/ground_truth/", pred_fi
le_loc)
  performance_table["character_wise_precision", I + 2] <- precision_chars("../output/ground_truth/", p
red_file_loc, "../output/tesseract/")
}
knitr::kable(performance_table, caption="Evaluation in interations")
```