STAT210 (Mining the History of Holyoke) Project Jin: Reverse Directory

Kevin Jin

2023-05-17

Table of contents

Background

Importance of Holyoke's History

The history of Holyoke, MA is very rich and deep; starting as a water town, Green (1939) chronicles the history of Holyoke from its beginnings as a farming district to its transformation "into a mill town by the exploitation of its water power". More succinctly, Green writes that

> "The history of Holyoke is the history of Massachusetts industrialism and all its social consequences."

As such, learning and studying the history of Holyoke is greatly impactful in learning more about not only the history of Massachusetts but also the history of the United States. Green mentions that much of the data to write her book was obtained from living persons and getting access to records and manuscripts. From this, we can see that in order to continue gleaning insights from the history of Holyoke, we need to make these historical archives more accessible by digitizing them for future preservation. Much work has been conducted towards this goal, one of which was the archiving of the public and commercial directories of Holyoke, more of which will be explained in the next section

Holyoke Directory

Directories and registries are typically used as an authoritative list of information, whether government or commercially made. One example of a directory is the yellow pages and white pages which contain business and people information in geographical areas and their addresses. Similarly to this, cities and municipals also have directories that contain information on where people live. Typically, these directories contain information on individuals who live in a city, with other miscellaneous information such as marital status (and with whom), occupation, and most importantly, their addresses. However, this information was typically stored in sorting based on the person's name. As such, reverse directories were created to help sort this information based on the address of the individual.

These reverse directories have many beneficial properties; not only can they help with gaining more genealogical insights regarding families and individuals who want to learn about their family's history, but they can also help with understanding how the architecture of a location changes over time.

Holyoke first started maintaining their reverse directory in 1915 to help with this task. However, Eileen from the Holyoke Public Library History Room mentioned that individuals trying to locate their family's history typically cannot progress further before 1915. As such, there was a demand to help assist Holyoke's history by creating such a reverse directory.

Toward making this free and publicly accessible directory, we will be using the 1885 Holyoke Directory archived by the Internet Archive in 2013 (link here for the 1885 directory). More specifically, we will be using the OCR converted `.txt` file of the scanned pdf, which introduced some errors and mistakes in formatting.

As such, we list below the goals that we are hoping to achieve with the creation of the reverse directory:

- We want to first clean up the text file as there are many formatting issues and text corruptions in many places. Some examples of these issues will be shown later.
- We then want to create an automated process that will parse through each line of the text to collect the address information of individuals.

- Then, we will want to output it and create a `CSV` file that we will give access to Eileen and the Holyoke Public Library room as an important resource.

Creating the 1885 Holyoke Reverse Directory

Initial File Information and Example Errors

The text file that we will be using had 40793 lines. However, the directory information contained was from lines 4376 to 21928 (which we manually found which is in total 17553 lines). From this directory information, there were many errors. For example, in line 1674 (of the new text file we created of just the directory information), there was a long text of errors and corrupted characters that lasted until line 2133. A piece of the corruption is printed below (lines 2102 to 2111).

```
[1] "09"    ""     "O"    ""     "D"    ""     "CD"   "CO"   ""     "â\200¢a"
```

From looking at this section in the actual directory, we see that it corresponded with an advertisement in the original archived directory, an example shown below in Figure 1.

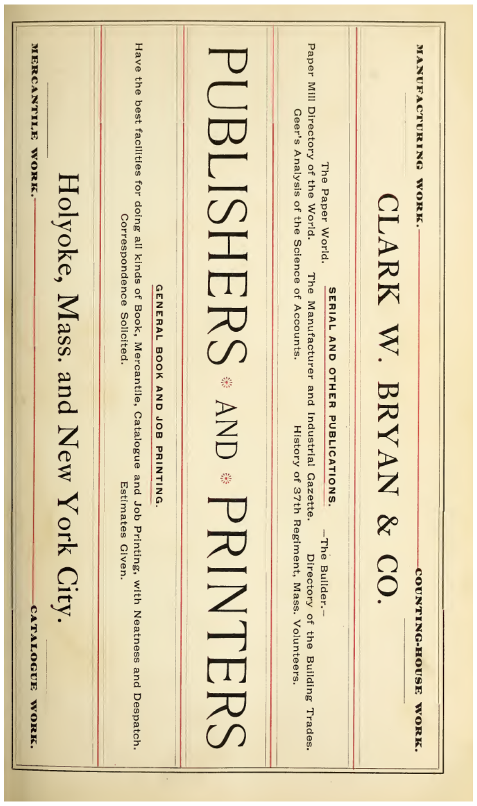
As these were just advertisements in the directory, we decided to not include this information as the addresses were more important. As such, we removed all these errors manually after checking where the advertisements were supposed to be.

Parsing the Address from Text

After removing these errors, we created a script file that contains the functions we used to help parse the information from the filtered text file. Without going into too many technical details about how these functions worked (all the code used will be at the end of this report), we used abbreviation information from the beginning of the directory shown in Figure 2 to help decipher the address and other information.

One challenge we faced while parsing the addresses was including the information for the businesses. The business information had a multitude of edge cases that messed with the algorithm, so we decided to manually filter these cases out (around 30 of them) and manually fill these cases out. We also had to deal with addresses being split into multiple lines. We were able to combine them by detecting the presence of a "," or "-" at the end of a line and joining it with the next non-blank line, or by the length of a string (we found that anything less than ~25 characters was typically one. We manually adjusted the others.)

Furthermore, other cases we had to deal with were that some people had information that they were removed to a different location (like to Canada or New York) which we also preserved


MANUFACTURING WORK. — COUNTING-HOUSE WORK.

CLARK W. BRYAN & CO.

SERIAL AND OTHER PUBLICATIONS.

The Paper World. —The Builder.—
Paper Mill Directory of the World. The Manufacturer and Industrial Gazette. Directory of the Building Trades.
Geer's Analysis of the Science of Accounts. History of 37th Regiment, Mass. Volunteers.

PUBLISHERS AND PRINTERS

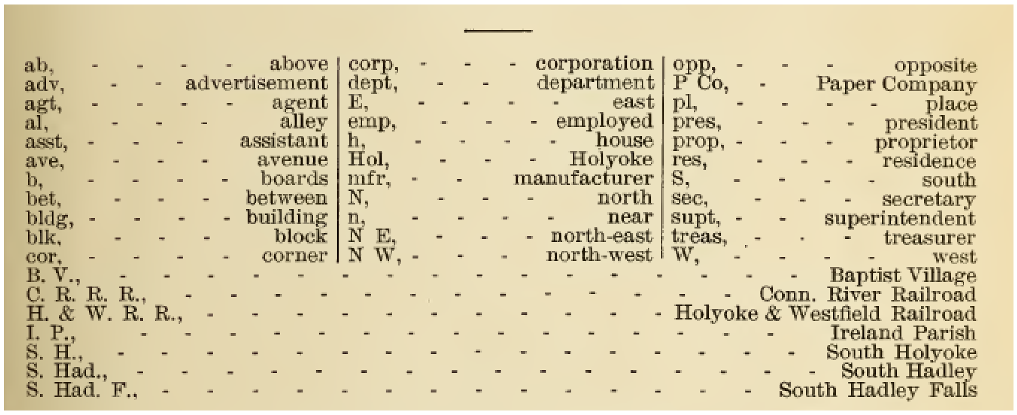
GENERAL BOOK AND JOB PRINTING.

Have the best facilities for doing all kinds of Book, Mercantile, Catalogue and Job Printing, with Neatness and Despatch.
Correspondence Solicited. Estimates Given.

Holyoke, Mass. and New York City.

MERCANTILE WORK. — CATALOGUE WORK.

Figure 1: Example of an advertisment that was corrupted.



ab,	above	corp,	corporation	opp,	opposite
adv,	advertisement	dept,	department	P Co,	Paper Company
agt,	agent	E,	east	pl,	place
al,	alley	emp,	employed	pres,	president
asst,	assistant	h,	house	prop,	proprietor
ave,	avenue	Hol,	Holyoke	res,	residence
b,	boards	mfr,	manufacturer	S,	south
bet,	between	N,	north	sec,	secretary
bldg,	building	n,	near	supt,	superintendent
blk,	block	N E,	north-east	treas,	treasurer
cor,	corner	N W,	north-west	W,	west
B. V.,	Baptist Village				
C. R. R. R.,	Conn. River Railroad				
H. & W. R. R.,	Holyoke & Westfield Railroad				
I. P.,	Ireland Parish				
S. H.,	South Holyoke				
S. Had.,	South Hadley				
S. Had. F.,	South Hadley Falls				

Figure 2: Abbreviation Table

in our reverse directory, as this information can be useful to be used. We also had to include other information like the presence of “do” in the text, which we later found to stand for ditto. This was used when the street name was the same as another address listed earlier in the line (like for a business) and sometimes the house number changed. An example is included below that was split into two lines.

```
[1] "DUFRESNE  ALEXANDER  N.,    grocer  and  liquor   dealer   49"
[2] "Cabot,  house  51  do  [See  page  463.]"
```

Output

At the end of extracting this information, we sorted the information lexicographically by first address and then street number (if applicable). An example portion of our data frame output is shown below in Table 1.

Table 1: Example output of reverse directory

location	address	last_name	name	description
Adams	2	Champagne	Joseph	emp Newton Paper Co., house 2 Adams
Adams	4	Gordon	George	emp Springfield Blanket Co., house 4 Adams
Adams	11	Richards	Thomas	emp Landers Bros., house 11 Adams
Adams	12	Hoffman	Gustave E.	cigar maker, emp 512 Main, house 12 Adams
Adams	12	Loomis	Eli	spinner, house 12 Adams
Adams	12	Morey	Annie Mrs.	house 12 Adams

As we can see, we have sorted the information in this directory by first their location/street name, and then by their street number. We have also included the original text information for these individuals to confirm that we correctly parsed the information. After creating this output, we have shared access with it to Eileen for further adjustments that need to be made, but we are confident to say that a majority of the addresses in the directory were able to be stored in our reveres-directory format.

Conclusion

We hope that citizens and other researchers can use this reverse directory of Holyoke from 1885 as a resource to help gain additional information about the history of Holyoke. As Green mentioned, researching the history of Holyoke is important as this history was the history of Massachusetts. Furthermore, we also hope that this reverse directory can be used by Holyoke residents to help track down their family’s history.

References

Green, Constance McLaughlin. 1939. *Holyoke, Massachusetts: A Case History of the Industrial Revolution in America*. Yale University Press, New Haven, CT. https://archives.yale.edu/repositories/12/archival_objects/1572349.

Appendix with All Code

```
library(mosaic)
library(tidyverse)
options(digits = 4)
knitr::opts_chunk$set(
  fig.height=2,
  tidy=FALSE,     # display code as typed
  size="small")   # slightly smaller font for code
library(stringr)
library(mosaic)
library(dplyr)
library(tidyverse)
source("../functions/scripts.R")
full_data <- readLines("../data/full_directory_data.txt")
print(full_data[2102:2111])
knitr::include_graphics("images/example ad.png")
###### Script Functions
## Find what the last character of a string matches a pattern
last_char_match <- function(str, pattern){
  len <- str_length(str)
  return(substr(str, start = len, stop = len) == pattern)
}


## Remove hyphenation from a string
remove_hyphenation <- function(str){
  return(strsplit(str, split = "-")[[1]][1])
}


## Function used during my for loop to keep track of if to use same index as previous
## line or for the next line to use the same index
## Returns a list of fixed string and the bools
idx_manager <- function(str){
  len <- str_length(str)
  ret_str <- str
  prev_bool <- FALSE
  next_bool <- FALSE

  # Keep track if should use same index as previous
  ## Chose 25 because saw that entries less than 25 don't typically exist.
  ## Will keep this in mind as I continue doing work
  if(len < 25){
    prev_bool <- TRUE
  } else if(!detect_page_header(str) & str_detect(str, pattern = "\\]")){
    prev_bool <- TRUE
  }

  # Keep track if next line should use same index as current
  if(last_char_match(str = str, pattern = "-")){
    next_bool <- TRUE
    ##
    ret_str <- remove_hyphenation(ret_str)
  } else if(last_char_match(str = str, pattern = ",")){
    next_bool <- TRUE
  }

  return(list(ret_str, prev_bool, next_bool))
}

## Count commas in a string
count_commas <- function(str){
  length(str_split(str, ",")[[1]]) - 1
}

## Detect for when a page header is seen
detect_page_header <- function(str, vec = page_header_vec){
  bool <- sum(sapply(vec, FUN = function(pattern, str){
    return(str_detect(str, pattern))
  }, str = str)) > 0
  return(bool)
}

page_header_vec <- c("HOLYOKE", "DIRECTORY")

filter_address <- function(str){
  address <- str
  if(str_detect(str, pattern = "[0-9]")){
    address = sub("^\\D+", "", str)
  } else if(str_detect(str, pattern = " res ")){
    address = sub(" res ", "", str)
  } else if(str_detect(str, pattern = "residence")){
    address = sub("residence", "", str)
  } else if(str_detect(str, pattern = " h ")){
    address = sub(" h ", "", str)
  } else if(str_detect(str, pattern = " house ")){
    address = sub(" house ", "", str)
  } else if(str_detect(str, pattern = "removed")){
    str_vec <- str_split(str, pattern = " ")[[1]]
    str_vec <- str_vec[str_vec != ""]
    n <- length(str_vec)
    address <- paste(str_vec[3:n], collapse = " ")
  } else if(str_detect(str, pattern = " rem ")){
    str_vec <- str_split(str, pattern = " ")[[1]]
    str_vec <- str_vec[str_vec != ""]
    n <- length(str_vec)
    address <- paste(str_vec[3:n], collapse = " ")
  } else if(str_detect(str, pattern = " rem ")){
    vec <- str_split(str, pattern = " ")[[1]]
    vec <- vec[vec != ""]
    n <- length(vec)
    address <- paste(vec[3:n], collapse = " ")
  } else if(str_detect(str, pattern = " boards ")){
    address = sub(" boards ", "", str)
  } else if(str_detect(str, pattern = " b ")){
    address = sub(" b ", "", str)
  }
  return(address)
}

extract_address <- function(str, last_name){
  # Label Businesses
  if(last_name == toupper(last_name)){
    return("Business")
  }
  ## Addresses will be XXX for special cases
  address <- "XXX"
  commas = count_commas(str)
  if(commas == 0){
    address <- filter_address(str)
    return(address)
  }

  ## If multiple commas, prioritize the last phrase
  str_vec <- str_split(str, pattern = ",")[[1]]


  ## If negative number, don't need, if it's not negative, replace the current number with
  number <- -1
  for(i in (commas+1):1){
    curr_str <- str_vec[i]
    temp_address <- filter_address(curr_str)
    ## If temp_address is not the same, filter_address did something so keep results
    if(temp_address != curr_str){
      ## if we detect Do, we want to use the number found to replace the street from befor
      if(str_detect(temp_address, pattern = " do", negate = TRUE)){
        address <- temp_address
        break
      }else{
        if(str_detect(temp_address, pattern = "[0-9]")){
          number <- as.numeric(gsub("[^0-9.-]", "", temp_address))
        }
      }
    }
  }
  ## Special case to account for that
  if(number > 0){
    address <- paste(number, trimws(gsub("[[:digit:]]", "", address)))
  }

  return(address)
}
###############
knitr::include_graphics("images/abbreviations.png")
print(full_data[5286:5287])
n <- length(full_data)

last_text_idx <- 0
empty_line_counter <- 0
idx_vec <- rep(0, n)
edge_case_vec <- rep(0, n)
comma_count_vec <- rep(0, n)
```

```r
new_data <- rep("", n)


prev_idx_bool <- F
next_idx_bool <- F

for(i in 1:n){
  curr_idx <- i
  curr_line <- full_data[i]
  curr_line_len <- str_length(curr_line)

  if(curr_line_len == 0){
    empty_line_counter = empty_line_counter +1
    new_data[i] <- curr_line
  }
  else{
    if(empty_line_counter >= 2){
      if(detect_page_header(str = curr_line, vec = page_header_vec)){
        curr_idx <- 0
      }else if(idx_vec[i-1] != 0){
        edge_case_vec[i] <- 1
      }
    }

    ## if the last text line had a hyphen at the end
    if(next_idx_bool){
      curr_idx <- idx_vec[last_text_idx]
      idx_vec[i] <- curr_idx
      curr_line_list <- idx_manager(curr_line)
      new_data[i] <- curr_line_list[[1]]
      prev_idx_bool <- curr_line_list[[2]]
      next_idx_bool <- curr_line_list[[3]]
      empty_line_counter <- 0
    }
    else{
      curr_line_list <- idx_manager(curr_line)
      new_data[i] <- curr_line_list[[1]]
      prev_idx_bool <- curr_line_list[[2]]
      next_idx_bool <- curr_line_list[[3]]
      if(prev_idx_bool & (curr_idx != 0)){
        curr_idx <- idx_vec[last_text_idx]
      }
      idx_vec[i] <- curr_idx
      last_text_idx <- i
      empty_line_counter <- 0
    }

  }
  comma_count_vec[i] <- count_commas(curr_line)
}

## example
collapsed <- tibble(text = new_data, index = idx_vec,
                    comma_count = comma_count_vec) |>
  group_by(index) |>
  summarise(
    combined_text = paste(text, collapse = ""),
    comma_count = sum(comma_count)
  ) |>
  filter(index != 0)
text_matrix <- str_split_fixed(collapsed$combined_text, ",", 2)
collapsed$full_name <- text_matrix[,1]
collapsed$description <- text_matrix[,2]
full_name_matrix <- str_split_fixed(collapsed$full_name, " ", 2)
collapsed$last_name <- full_name_matrix[,1]
collapsed$name <- trimws(full_name_matrix[,2])
## Override function from scripts
extract_address <- function(str, last_name){
  # Label Businesses
  business <- FALSE
  if(last_name == toupper(last_name)){
    business <- TRUE
  }
  ## Addresses will be XXX for special cases
  address <- "XXX"

  if(!business){
    commas = count_commas(str)
    if(commas == 0){
      address <- filter_address(str)
      return(address)
    }
```

```r
    ## If multiple commas, prioritize the last phrase
    str_vec <- str_split(str, pattern = ",")[[1]]


    ## If negative number, don't need, if it's not negative, replace the current number wi
    number <- -1
    for(i in (commas+1):1){
      curr_str <- str_vec[i]
      temp_address <- filter_address(curr_str)
      ## If temp_address is not the same, filter_address did something so keep results
      if(temp_address != curr_str){
        ## if we detect Do, we want to use the number found to replace the street from bef
        if(str_detect(temp_address, pattern = " do", negate = TRUE)){
          address <- temp_address
          break
        }else{
          if(str_detect(temp_address, pattern = "[0-9]")){
            number <- as.numeric(gsub("[^0-9.-]", "", temp_address))
          }
        }
      }
    }
    ## Special case to account for that
    if(number > 0){
      address <- paste(number, trimws(gsub("[[:digit:]]", "", address)))
    }

  } else{
    commas = count_commas(str)
    if(commas == 0){
      address <- filter_address(str)
      return(address)
    }
    commas = count_commas(str)
    if(commas == 0){
      address <- filter_address(str)
      return(address)
    }

  }
  return(address)
}

n <- nrow(collapsed)
address <- rep("", n)
exceptions = c(132,357,755,1133,1455,1860,1918,2306,2960,3123,3663,4205,4398,
               4562,4757,4772,4928,5842,6769,6771,6772,6872,7168,8228,8236,8277,
               8460,8485,8671,9080)
for(i in 1:nrow(collapsed)){
  last_name <- collapsed$last_name[i]
  description <- collapsed$description[i]
  if(i %in% exceptions){
    address[i] <- description
  }
  else{
    address[i] <- trimws(extract_address(description, last_name))
  }
}
collapsed$full_address <- address
output <- collapsed |>
  #filter(full_address != "Business") |>
  mutate(location = case_when(str_detect(full_address, "[0-9]") ~
                              ## Drop all numerics from letters
                                trimws(gsub("[[:digit:]]", "", full_address)),
                              T ~ full_address),
         address = case_when(str_detect(full_address, "[0-9]") ~
                             ## drop all letters from numbers
                                trimws(gsub("[^0-9.-]", "", full_address)),
                              T ~ "")) |>
  select(location, address, last_name, name)
write.csv(output, file = "../output/full_data_output2.csv")
output <- read.csv("../output/full_data_output2.csv") |>
  filter(X > 135) |>
  select(-X)
output |>
  head() |>
  kableExtra::kable(booktab = T, align = "c", label = NA,
                    caption = "\\label{tab1}Example output of reverse
                    directory") |>
  kableExtra::kable_styling(latex_options = "HOLD_position")
```