Identifying Patients with Diabetic Complications

Purpose of the Project

This project applies the text processing and clinical natural language processing techniques learned in this course to a real-world clinical problem: identifying diabetic complications from unstructured clinical notes. Using a corpus of notes from patients with diabetes, the goal is to determine whether a note documents the presence of diabetic complications and, if so, which type of complication is described.

Specifically, this project focuses on identifying three common diabetic complications—neuropathy, nephropathy, and retinopathy—by analyzing free-text clinical documentation. A keyword window–based approach is used to capture local clinical context around diabetes-related terms, while exclusion rules are applied to remove negated mentions and references to family history. The resulting predictions are evaluated against a gold-standard dataset to assess model performance.

By completing this project, the workflow demonstrates how clinical text can be systematically processed to extract meaningful phenotypic information that may support clinical research, population health studies, and decision support applications.

Getting data

Load packages, connect to BigQuery, then downloads two talbes into R

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)

library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────────────── t
idyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.6
## ✓ forcats   1.0.1     ✓ stringr   1.6.0
## ✓ ggplot2   4.0.1     ✓ tibble    3.3.0
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.2
## ✓ purrr     1.2.0
## ── Conflicts ────────────────────────────────────────────────────────
──────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to b
ecome errors
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(bigrquery)
library(DBI)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
con <- DBI::dbConnect(
  drv = bigquery(),
  project = "learnclinicaldatascience"
)

my_project <- "learnclinicaldatascience"

patient <- bq_project_query(
  my_project,
  "
  SELECT *
  FROM `course4_data.diabetes_goldstandard`
  "
) %>% bq_table_download()
```

```
## ! Using an auto-discovered, cached token.
##   To suppress this message, modify your code or options to clearly consent to
##   the use of a cached token.
##   See gargle's "Non-interactive auth" vignette for more details:
##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>
## ℹ The bigrquery package is using a cached token for 'erinzhang0220@gmail.com'.
```

```
notes <- bq_project_query(
  my_project,
  "
  SELECT *
  FROM `course4_data.diabetes_notes`
  "
) %>% bq_table_download()
```

Feature engineering: Key window technique

First, create text windows for notes containing the word related to diabetes (window size = 20 words total; ±10 words).

```
extract_text_window <- function(dataframe, keyword, half_window_size) {
  dataframe %>%
    group_by(NOTE_ID) %>%
    mutate(WORDS = TEXT) %>%
    separate_rows(WORDS, sep = "[ \n]+") %>%
    mutate(
      INDEX = row_number(),
      WINDOW_START = if_else(INDEX - half_window_size < 1, 1L, INDEX - half_window_size),
      WINDOW_END   = if_else(INDEX + half_window_size > max(INDEX), max(INDEX), INDEX + half
_window_size),
      WINDOW = stringr::word(TEXT, start = WINDOW_START, end = WINDOW_END, sep = "[ \n]+")
    ) %>%
    ungroup() %>%
    filter(str_detect(WORDS, regex(keyword, ignore_case = TRUE)))
}
```

```
diabetes_notes_window <- notes %>%
  extract_text_window(
    keyword = "(?<![a-zA-Z])diabet(es|ic)?(?![a-zA-Z])",
    half_window_size = 10
  )
```

Exclusion rules (remove negations / family history)

```
diabetes_notes_clean <- diabetes_notes_window %>%
  mutate(
    EXCLUDE = case_when(
      str_detect(WINDOW, regex("no history of (?<![a-zA-Z])diabet(es|ic)?(?![a-zA-Z])", igno
re_case = TRUE)) ~ 1,
      str_detect(WINDOW, regex("den(ies|y)? any comorbid complications", ignore_case = TRU
E)) ~ 1,
      str_detect(WINDOW, regex("family history", ignore_case = TRUE)) ~ 1,
      str_detect(WINDOW, regex("negative for (?<![a-zA-Z])diabet(es|ic)?(?![a-zA-Z])", ignor
e_case = TRUE)) ~ 1,
      str_detect(WINDOW, regex("(father|mother) (also)? (?<![a-zA-Z])diabet(es|ic)?(?![a-zA-
Z])", ignore_case = TRUE)) ~ 1,
      str_detect(WINDOW, regex("no weakness, numbness or tingling", ignore_case = TRUE)) ~
1,
      TRUE ~ 0
    )
  ) %>%
  filter(EXCLUDE != 1)
```

Identify complications from windows

```
diabetes_complic <- diabetes_notes_clean %>%
  mutate(
    COMPLICATIONS = case_when(
      str_detect(WINDOW, regex("(?<![a-zA-Z])neuropath(y|ic)?(?![a-zA-Z])", ignore_case = TR
UE)) ~ "neuropathy",
      str_detect(WINDOW, regex("diabetic nerve pain", ignore_case = TRUE)) ~ "neuropathy",
      str_detect(WINDOW, regex("tingling", ignore_case = TRUE)) ~ "neuropathy",

      str_detect(WINDOW, regex("(?<![a-zA-Z])nephropathy(?![a-zA-Z])", ignore_case = TRUE))
~ "nephropathy",
      str_detect(WINDOW, regex("renal (insufficiency|disease)", ignore_case = TRUE)) ~ "neph
ropathy",

      str_detect(WINDOW, regex("(?<![a-zA-Z])retinopath(y|ic)?(?![a-zA-Z])", ignore_case = T
RUE)) ~ "retinopathy",

      TRUE ~ ""
    )
  ) %>%
  arrange(desc(COMPLICATIONS))
```

Build note-level predictions

```
pred_neuro <- diabetes_complic %>%
  filter(COMPLICATIONS == "neuropathy") %>%
  distinct(NOTE_ID) %>%
  mutate(predicted_neuropathy = 1)

pred_nephro <- diabetes_complic %>%
  filter(COMPLICATIONS == "nephropathy") %>%
  distinct(NOTE_ID) %>%
  mutate(predicted_nephropathy = 1)

pred_retino <- diabetes_complic %>%
  filter(COMPLICATIONS == "retinopathy") %>%
  distinct(NOTE_ID) %>%
  mutate(predicted_retinopathy = 1)
```

Evaluate performance

```
getStats <- function(df, ...) {
  df %>%
    select(...) %>%
    mutate(across(everything(), ~ factor(.x, levels = c(1, 0)))) %>%
    table() %>%
    confusionMatrix()
}
eval_df <- patient %>%
  left_join(pred_neuro,  by = "NOTE_ID") %>%
  left_join(pred_nephro, by = "NOTE_ID") %>%
  left_join(pred_retino, by = "NOTE_ID") %>%
  mutate(
    predicted_neuropathy  = coalesce(predicted_neuropathy, 0),
    predicted_nephropathy = coalesce(predicted_nephropathy, 0),
    predicted_retinopathy = coalesce(predicted_retinopathy, 0),
    predicted_complic = if_else(
      predicted_neuropathy == 1 | predicted_nephropathy == 1 | predicted_retinopathy == 1,
      1, 0
    )
  ) %>%
  select(NOTE_ID, ANY_DIABETIC_COMPLICATION, predicted_complic)

getStats(eval_df, predicted_complic, ANY_DIABETIC_COMPLICATION)
```

```
## Confusion Matrix and Statistics
##
##                   ANY_DIABETIC_COMPLICATION
## predicted_complic   1   0
##                 1  22   4
##                 0   5 110
##
##                Accuracy : 0.9362
##                  95% CI : (0.8823, 0.9704)
##     No Information Rate : 0.8085
##     P-Value [Acc > NIR] : 1.455e-05
##
##                   Kappa : 0.7909
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8148
##             Specificity : 0.9649
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.9565
##              Prevalence : 0.1915
##          Detection Rate : 0.1560
##    Detection Prevalence : 0.1844
##       Balanced Accuracy : 0.8899
##
##        'Positive' Class : 1
##
```

From the results, 22 cases were true positives and 110 were true negatives (correctly identified) while 4 were false positives and 5 were false negatives.

In conclusion, the accuracy of this keyword window technique used is 0.9362, while the sensitivity is 0.8148 and specificity is 0.9649. There is a balance of sensitivity and specificity, with a relatively high accuracy. Therefore, this method can be used to classify such clinical notes in the future. We could try to use the note section technique as an alternative method where we use the context provided by the note section to understand the meaning of the keyword matches in the text.

Rendered Report

The rendered HTML report is available here: - https://erin0220.github.io/Identify-diabetic-complications-from-unstructured-clinical-notes.html (https://erin0220.github.io/Identify-diabetic-complications-from-unstructured-clinical-notes.html)

Please download and open in a browser to view the full report.