Similarity Scores and Pipeline Approaches Applied to Conflict Data in El Salvador

STA 490/690: Assignment 1, Spring 2020

Due TBD

General instructions for homeworks: Please follow the uploading file instructions according to the syllabus. Your code must be completely reproducible and must compile.

Advice: Start early on the homeworks and it is advised that you not wait until the day of as these homeworks are meant to be longer and treated as case studies.

Commenting code Code should be commented. See the Google style guide for questions regarding commenting or how to write code https://google.github.io/styleguide/Rguide.xml. No late homework's will be accepted.

R Markdown Test

0. Open a new R Markdown file; set the output to HTML mode and "Knit". This should produce a web page with the knitting procedure executing your code blocks. You can edit this new file to produce your homework submission.

Working with data

Total points on assignment: 5 (reproducibility) + 10 points for the assignment.

Recall that between 1980 and 1991, the Republic of El Salvador witnessed a civil war between the central government, the left-wing guerrilla Farabundo Marti National Liberation Front (FMLN), and right-wing paramilitary death squads. After the peace agreement in 1992, the United Nations created a Commission on the Truth (UNTC) for El Salvador, which invited members of Salvadoran society to report war-related human rights violations, which mainly focused on killings and disappearances.In order to collect such information the UNTC invited individuals through newspapers, radio, and television advertisements to come forward and testify. The UNTC opened offices through El Salvador where witnesses could provide their testimonials, and this resulted in a list of potential victims with names, date of death, and reported location.

In this assignment, you will explore the UNTC data set to get a better understanding of how to work with real data versus toy data. Let's read in the data.

```
library(knitr)
library(RecordLinkage)
```

```
# read in data
df <- read.csv("./sv-mauricio/sv-mauricio.csv")
head(df)
```

```
##     X  ID         lastname firstname day month year geocode HandID dept muni
## 1  26  32 ASENSIO ERNANDES   ALBERTO  NA     2 1981  150000     NA   15   NA
## 2  84  95   PALASIOS AYALA    OBIDIO  NA    10 1985  150000     NA   15   NA
## 3 100 117            PALMA SEBASTIAN  13     5 1980   40000     NA    4   NA
## 4 143 173            PERES   ARCADIO  NA     8 1984   40000     NA    4   NA
## 5 170 205     MAYA QUESADA   ANTONIO  22     9 1984       0     NA    0   NA
## 6 189 227            MEJIA   ALFONSO  13     5 1980   40000     NA    4   NA
```

```r
dim(df)
```

```
## [1] 5395   11
```

Next, let's filter out any records that do not have ground truth information.

```r
ent_id <- df$HandID
# Filter out records with ground truth, leaving dept 1 and 7
df <- df[!is.na(ent_id),]
ent_id <- ent_id[!is.na(ent_id)]
head(df)
```

```
##        X   ID      lastname firstname day month year geocode HandID dept muni
## 26   543  654  ALEMAN SOLIS   ALFREDO   2     5 1984   70000    136    7   NA
## 64  1406 1687          CRUS    CARMEN  21    10 1981   10000    639    1   NA
## 66  1470 1772       MONTOYA    CARMEN  NA     3 1982   70000    201    7   NA
## 70  1486 1792 PAS SINGUENSA JUAN JOSE  22    10 1980   70000    202    7   NA
## 112 2461 2942        GUIYEN   TEODORO  NA    NA 1983   70000    310    7   NA
## 144 3140 3750      MANOQUIN     JULIA  NA     3 1982   70000      6    7   NA
```

```r
tail(df)
```

```
##         X   ID lastname          firstname day month year geocode HandID dept
## 4737 4150 4972 PICHINTE        FELIX JESUS  22     3 1980   71608    490    7
## 4738 4151 4973 PICHINTE FRANSISCO JERONIMO  22     3 1980   71608    491    7
## 4739 4599 5524   RIBERA          FRANSISCO  22     7 1984   71600    533    7
## 4740 4679 5623   RIBERA              TOMAS  22     7 1984   71600    536    7
## 4741 5123 6161 SIGUENSA      OSCAR ANTONIO  NA     4 1981   71609    580    7
## 4742 5301 6372    BAYES      JOSE OLIBERIO  19     9 1982   71601    595    7
##      muni
## 4737  716
## 4738  716
## 4739  716
## 4740  716
## 4741  716
## 4742  716
```

```r
dim(df)
```

```
## [1] 735  11
```

Observe that we are only considering two municipalities in El Salvador now, which is what was considered in Sadinle (2014).

Task 1

Consider the similarity of first name and last name. What type of distance metric would you use for this data set and why?

Consider comparing the following two names using Edit distance: ALFREDO and CARMEN. We find that using the following code below that the distance is just 0.1428571. This seems quite reasonable given that both names are quite different.

```r
levenshteinSim(df$firstname[1], df$firstname[2])
```

```
## [1] 0.1428571
```

Consider comparing the following two names using Edit distance: FRANSISCO JERONIMO and FRANSISCO. We find the Edit distance is 0.5.

We have now seen an interesting case. Hopefully, you have noticed that Hispanic first names, have two tokens, and thus, are much longer than Western names. Perhaps we might want to change the distance function.

Let's try using the Monge Elkan string distance metric. First, we will define the normalized Edit distance.

```
# Normalized Levenshtein similarity function used below
unitLevenshteinSimilarity <- function(v1, v2) {
  totalLength <- matrix(nchar(v1), nrow=length(v1), ncol=length(v2))
  totalLength <- sweep(totalLength, 2, nchar(v2), FUN = "+")
  dist <- adist(v1, v2)
  ifelse(totalLength > 0, 1.0 - 2.0 * dist / (totalLength + dist) , 1.0)
}
```

Now, we define the Monge Elkan metric.

```
#' Similarity function for Hispanic names based upon the Monge Elkan metric
#'
#' @param x a character vector
#' @param y a character vector
#' @param sep separator for tokens/words (uses white space by default)
#' @param knownTokens a character vector of known tokens (default is NULL)
#' @returns a length(x) × length(y) similarity matrix
unitHispanicSimilarity <- function(x, y, sep = '\\s+', knownTokens = NULL) {
  # Split into tokens (words)
  tokens1 <- strsplit(x, sep)
  tokens2 <- strsplit(y, sep)

  # Preallocate similarity matrix for output
  out <- matrix(0.0, nrow = length(tokens1), ncol = length(tokens2))

    if (!is.null(knownTokens)) {
    # Convert known tokens to environment for faster look-up
    knownList <- setNames(replicate(length(knownTokens), 1, simplify = FALSE), knownTokens)
    knownEnv <- list2env(knownList, hash = TRUE, size = length(knownList))
  }

  # Function to compute the symmetrized Monge-Elkan similarity for a single
  # pair of tokens
  meSim <- function(t1, t2) {
    maxSim1 <- numeric(length=length(t1))
    knownDistinct1 <- logical(length=length(t1))
    maxSim2 <- numeric(length=length(t2))
    knownDistinct2 <- logical(length=length(t2))
    for (i in seq_along(t1)) {
      for (j in seq_along(t2)) {
        sim <- unitLevenshteinSimilarity(t1[i], t2[j])
        bothKnownDistinct <- FALSE
        if (!is.null(knownTokens) && t1[i] != t2[j] &&
            exists(t1[i], envir = knownEnv, inherits = FALSE) &&
            exists(t2[i], envir = knownEnv, inherits = FALSE)) {
          bothKnownDistinct <- TRUE
        }
        if (sim > maxSim1[i]) { maxSim1[i] <- sim; knownDistinct1[i] <- bothKnownDistinct }
        if (sim > maxSim2[j]) { maxSim2[j] <- sim; knownDistinct2[j] <- bothKnownDistinct }
      }
    }
```

```r
    maxSim1 <- ifelse(knownDistinct1, 0, maxSim1)
    maxSim2 <- ifelse(knownDistinct2, 0, maxSim2)
    # Symmetrize
    return(max(length(t1)/sum(1.0/maxSim1), length(t2)/sum(1.0/maxSim2)))
  }

  # Function to compute an asymmetric similarity for a single pair of tokens
  asymSim <- function(t1, t2) {
    if (length(t1) < length(t2)) {
      # If t2 contains extra tokens, similarity is zero (can't distort
      # true name by adding names)
      return(0)
    } else {
      # Get symmetrized Monge-Elkan similarity
      me <- meSim(t1, t2)
      # Assign 0.95 weight to Monge-Elkan and 0.05 weight to num. tokens
      # similarity
      #return(1.0/(0.95/me + 0.05*length(t1)/length(t2)))
      return(me)
    }
  }

  # Loop over all combinations in input character vectors
  for (i in seq_len(length(tokens1))) {
    for (j in seq_len(length(tokens2))) {
      out[i, j] <- asymSim(tokens1[[i]], tokens2[[j]])
    }
  }

  return(out)
}
```

Now when comparing ALFREDO and CARMEN under the Monge Elkan metric our score is 0.3684211.

Now when comparing FRANSISCO JERONIMO and FRANSISCO under the Monge Elkan metric our score is 1.

Play around with this metric more to see if this is a good fit.

Task 2

How does exact matching work on this data set? What about off by one matching? Be sure to provide the precision and recall.

Task 3

How would you build a decision rule for matches/non-matches based upon scoring rules. What would your scoring rule be? Write this up as an algorithm.

Task 4

Code up your algorithm in Task 3 and provide the precision and recall. Did your method do better or worse than exact matching?

Task 5

Give insights into how you might be able to improve deterministic approaches moving forward if you re-did your analysis. What advice would you give to a new member that is just joining the project after working on this project (assume that they have just joined your team and your job is to bring them up to speed).