Speech Recognition

SMO GROUP

29 March 2016

1. Introduction

Nick

2. Dynamic Time Warping

Nick

2.1 Algorithm

Nick

2.2 Modifications

Major modifications are the step size conditions, step weighting and the global path constraints.

2.3 Applications

3. Speech Recognition

We implemented a speech recognizer using a modified version of the DTW algorithm. Using DTW and speech processing techniques (i.e. Mel Frequency Cepstrum Coefficients), our algorithm is able to detect which action we want to perform: open the google search or open our facebook webpage.

This speech recognizer takes as input a sound file and compares it with some template words: *google* and *facebook*. In order to make our recognizer more robust, we have used two template words for each sound. After comparing the input sound with each template word, the algorithm chooses the one with the shortest path.

3.1 Speech Processing

AINA Blablabla

3.2 Dynamic Time Warping in this project

Blablabla

3.3 Results

Blablabla

4. Conclusions

Blablabla

5. Bibliography

6. Annex

6.1 DTW function

```
TimeWarp<-function(x,y,w=4){

  # define distance function
  distance<-function(a,b){
    dist(rbind(a,b))
    }

  # 1. Compute matrix 11xM

  # set parameters
  m<-dim(x)[2]
  n<-dim(y)[2]
  colnames(x)<-1:m
  colnames(y)<-1:n
  w = max(w, abs(n-m))

  # Create matrix
  DTW<-matrix(Inf,n,m)
  rownames(DTW)<-n:1
  colnames(DTW)<-1:m

  # Initial values
  DTW['1','1']<-distance(x[,'1'], y[,'1'])

   # First row
  for(j in 2:(w+1)){
    cost<-distance(x[,as.character(j)], y[,as.character(1)])
    DTW['1',as.character(j)]<- cost + DTW['1', as.character(j-1)]
  }

   # First column
  for(i in 2:(w+1)){
    cost<-distance(x[,as.character(1)],y[,as.character(i)])
    DTW[as.character(i), '1']<- cost + DTW[as.character(i-1), '1']
  }

  # Fill matrix
  for(i in 2:n){
    for(j in (max(2, i-w)):(min(m, i+w))){

      #current cost
      cost<-distance(x[,as.character(j)], y[,as.character(i)])

      #cumulated cost
      d.cost<-min(DTW[as.character(i-1), as.character(j)] ,
                  DTW[as.character(i), as.character(j-1)],
                  2*DTW[as.character(i-1), as.character(j-1)])
```

```
      #combined cost
      DTW[as.character(i),as.character(j)]<-cost + d.cost

    }
  }

  # 2. Find path
  path<-matrix(c(n,m), 1,2)
  full.path<-(tail(path,1)[1] ==1 & tail(path,1)[2] ==1)

  while(full.path==FALSE ){

    l.path<-tail(path,1)

    if(l.path[1]==1 | l.path[2]==1){
      p<-which(l.path==1)

          if(p==1){new.point<-c(l.path[1], l.path[2]-1)
           }else{
            new.point<-c(l.path[1]-1, l.path[2])
      }

    } else {

    # nearest point
    min.step<-min(DTW[as.character(l.path[1]-1), as.character(l.path[2]-1)],
        DTW[as.character(l.path[1]), as.character(l.path[2]-1)],
        DTW[as.character(l.path[1]-1), as.character(l.path[2])])
    min.step<-which(c(DTW[as.character(l.path[1]-1), as.character(l.path[2]-1)],
                    DTW[as.character(l.path[1]), as.character(l.path[2]-1)],
                    DTW[as.character(l.path[1]-1), as.character(l.path[2])])==min.step)
    min.step<-min.step[1]

    #path to nearest point
    if(min.step==1){
      new.point<-c(l.path[1]-1, l.path[2]-1)
    } else{
      if(min.step==2){
      new.point<-c(l.path[1], l.path[2]-1)
      } else{
        new.point<-c(l.path[1]-1, l.path[2])
      }
    }
    }
    path<-rbind(path,new.point)
    full.path<-(tail(path,1)[1] ==1 & tail(path,1)[2] ==1)

    }


return(list(path=path, DTW=DTW))
}
```

6.2 Speech Recognizer code

```
# input: isound is the path to the wav file with the sound.

SpeechRecognizer <- function(isound){

  if (!require("tuneR")) install.packages("tuneR");library(tuneR)

  # Read the wav file
  sound <- readWave(isound)
  sr    <- sound@samp.rate

  # Compute the mel frequency cepstrum coefficients
  inputWord <- t(melfcc(sound,
                        sr,
                        wintime=0.016,
                        lifterexp=0,
                        minfreq=133.33,
                        maxfreq=6855.6,
                        sumpower=FALSE))

  # Upload the four template sounds and compute their melfcc
  g1 <- readWave("Project\google1.wav")
  g2 <- readWave("Project\google2.wav")
  f1 <- readWave("Project\facebook1.wav")
  f2 <- readWave("Project\facebook2.wav")

  sr1 <- g1@samp.rate
  sr2 <- g2@samp.rate
  sr3 <- f1@samp.rate
  sr4 <- f2@samp.rate

  google1 <- t(melfcc(g1, sr1, wintime=0.016, lifterexp=0,
                      minfreq=133.33, maxfreq=6855.6, sumpower=FALSE))
  google2 <- t(melfcc(g2, sr2, wintime=0.016, lifterexp=0,
                      minfreq=133.33, maxfreq=6855.6, sumpower=FALSE))
  facebook1 <- t(melfcc(f1, sr3, wintime=0.016, lifterexp=0,
                        minfreq=133.33, maxfreq=6855.6, sumpower=FALSE))
  facebook2 <- t(melfcc(b2, sr4, wintime=0.016, lifterexp=0,
                        minfreq=133.33, maxfreq=6855.6, sumpower=FALSE))


  # Compute the distance of the input sound with the template sounds
  distance.sound<-rep(NA, 4)

  dtwg1 <- TimeWarp(google1, inputWord)
  distance.sound[1]<- tail(dtwg1$DTW[,1],1)

  dtwg2 <- TimeWarp(google2, inputWord)
  distance.sound[2]<- tail(dtwg2$DTW[,1],1)

  dtwf1 <- TimeWarp(facebook1, inputWord)
  distance.sound[3]<- tail(dtwf1$DTW[,1],1)
```

```
  dtwf2 <- TimeWarp(facebook2, inputWord)
  distance.sound[4]<- tail(dtwf2$DTW[,1],1)


  # If the minimum distance is to the word gmail, open gmail
  if (which.min(distance.sound) == 1 | which.min(distance.sound) == 2){
    system(paste("open http://google.com"))
  }

   # If the minimum distance is to the word facebook, open facebook
  if (which.min(distance.sound) ==3 | which.min(distance.sound) == 4){
     system(paste("open http://facebook.com"))
  }
}
```