Project 4

Team 4

Step 0: Load the packages, specify directories

```
start.time0 <- Sys.time()
if (!require("lda")) install.packages("lda")
```

```
## Loading required package: lda
```

```
library(lda)
source("../lib/cleandata.R")
source("../lib/evaluation_measures.R")
```

Step 1: Clean all the data and do the preparation work to run LDA and hclust

```
# set up the corpus of each document
corpus<- function(llist){
  document<- function(lllist) {
    words_apperance <- c(unlist(strsplit(lllist[[4]], " ")), unlist(strsplit(lllist[[5]], " ")))
    name_appearance <- unlist(lllist[[3]])
    doc <- list(words = words_apperance, name = name_appearance)
    return(doc)
  }
  return(lapply(llist, document))
}
doc_corpus<- lapply(data_list, corpus)

# extract all words(coauthor names, title of the paper and published journal of each documentation)
allwords_fun<- function(llist) {
  allwords_fun1<- function(lllist) {
    return(c(lllist$name, lllist$words))
  }
  return(lapply(llist, allwords_fun1))
}
doc_allwords<- lapply(doc_corpus, allwords_fun)

# the vocab(all words) used in the documents of each name)
vocab_fun<- function(llist) {
  vocab<- c()
  n<- length(llist)
  for(i in 1:n){
    vocab<- c(vocab,llist[[i]])
  }
  vocab<- unique(vocab)
  return(vocab)
}
vocab<- lapply(doc_allwords, vocab_fun)

# extract the Gold standard clusters for each author name
```

```r
query.g<- function(llist){
  gold<- function(lllist) {
    gold_id<- lllist[[1]]
    return(gold_id)
  }
  return(sapply(llist, gold))
}
gold_mat<- sapply(data_list, query.g)

# construct the list format we should use in the code
index<- function(x,a) {return(which(a==x))}
doc_format<- vector("list", 14)
for(i in 1:14) {
  format_fun<- function(lllist) {
    t<- table(lllist)
    vec1<- as.numeric(sapply(names(t), index, a=vocab[[i]]))-1
    vec2<- as.numeric(t)
    m<- matrix(as.integer(c(vec1, vec2)), ncol = 2)
    return(t(m))
  }
  doc_format[[i]] <- lapply(doc_allwords[[i]], format_fun)
}
names(doc_format)<- query.list
```

Run LDA

```r
# main function to run LDA model and tune the parameter
# Input: list_num: the index of the each list, for example: list_num = 1 represents the author name is
#        topic_num: the parameter used in LDA
main <- function(list_num, topic_num) {
  start.time <- Sys.time()
  k<- topic_num
  # parameter values
  beta <- 0.01
  alpha <- k/50

  # run the LDA
  runlda <- lda.collapsed.gibbs.sampler(doc_format[[list_num]],
                                        k, vocab[[list_num]],
                                        num.iterations = 1000,
                                        alpha = alpha, eta = beta)

  # Calculate topic-word matrix
  hw <- runlda$topics
  W <- length(vocab[[list_num]])
  sum_h <- rowSums(hw)
  matrix_sumh <- matrix(rep(sum_h,k),nrow=k,ncol=W)
  phi <- (hw+beta)/(matrix_sumh+W*beta)

  # Calculate topic-document probability matrix
  hd <- runlda$document_sums
  D <- length(doc_format[[list_num]])
```

```r
  sum_hd <- colSums(hd)
  matrix_sumhd <- matrix(rep(sum_hd,each=k),nrow=k,ncol=D)
  theta <- (hd+alpha)/(matrix_sumhd+k*alpha)
  colnames(theta) <- c(1:ncol(theta))
  distance<- dist(data.frame(t(theta)))

  # Agglomerative Clustering
  clust.num<- max(gold_mat[[list_num]])
  hcluster <- hclust(distance, "complete")
  hclust_id<- cutree(hcluster, gold_mat[[list_num]])

  # compute accuracy based on precision anf recall
  match_mat<- matching_matrix(gold_mat[[list_num]], hclust_id)
  perform<- performance_statistics(match_mat)
  end.time <- Sys.time()
  return(c(unlist(perform), cluster.time = end.time- start.time))
}
```

Tune the parameter topic number to get the best result

```r
# tune the parameter(topic number) to get the max accuracy
# the best topic number is either 5 or 10 based on the paper
tune <- function(list_num) {
  start.time <- Sys.time()
  k_vec<- c(5, 10)
  perform_mat <- sapply(k_vec ,main, list_num = list_num)
  acc_vec <- perform_mat[4,]
  best_ind <- which.max(acc_vec)
  best_k <- k_vec[best_ind]
  end.time <- Sys.time()
  return(c(best.k = best_k, perform_mat[,best_ind],
           tune.time = end.time - start.time))
}
```

Final results for each author names

```r
# final results
final_result<- sapply(1:14, tune)
colnames(final_result)<- query.list
final_result
```

```
##                  A Gupta    A Kumar     C Chen  D Johnson      J Lee
## best.k        10.0000000 10.0000000 10.0000000 10.0000000 10.0000000
## precision      0.4338890  0.4382329  0.2735917  0.5723818  0.2568857
## recall         0.4646897  0.2533901  0.5773682  0.2099941  0.2151242
## f1             0.4487615  0.3211110  0.3712587  0.3072610  0.2341575
## accuracy       0.8876432  0.7707954  0.9029650  0.7365093  0.9660412
## cluster.time   4.9249589  1.4877319  8.2993550  2.3734829 22.1419289
## tune.time      9.1305680  2.5784249 15.4836991  4.2802410 42.1893818
##                 J Martin J Robinson    J Smith   K Tanaka    M Brown
## best.k        10.0000000 10.0000000  5.0000000  5.0000000 10.0000000
```

```
## precision     0.5211480  0.4791425  0.5151446 0.6111917  0.6110260
## recall        0.5674342  0.3943729  0.7188079 0.4189234  0.4904733
## f1            0.5433071  0.4326445  0.6001690 0.4971144  0.5441527
## accuracy      0.9066924  0.8507740  0.8952495 0.8036866  0.8850189
## cluster.time  0.5156772  0.8751171  9.2509940 1.3461051  0.7534811
## tune.time     0.8907182  1.5158129 19.8247139 3.1432021  1.2847831
##                 M Jones   M Miller      S Lee     Y Chen
## best.k       10.0000000 10.0000000 10.0000000 10.0000000
## precision     0.5969715  0.8083791  0.2356007  0.2824923
## recall        0.4353366  0.2802381  0.6250594  0.4895228
## f1            0.5034999  0.4161952  0.3422127  0.3582480
## accuracy      0.8799228  0.7269979  0.9055538  0.8881911
## cluster.time  1.5625038  3.0785370 23.3412421 18.2842610
## tune.time     2.7778888  5.5492792 44.4391489 34.5199609
```

```r
end.time0<- Sys.time()

# the whole time to complete our algorithm on Name Disambiguation
end.time0 - start.time0
```

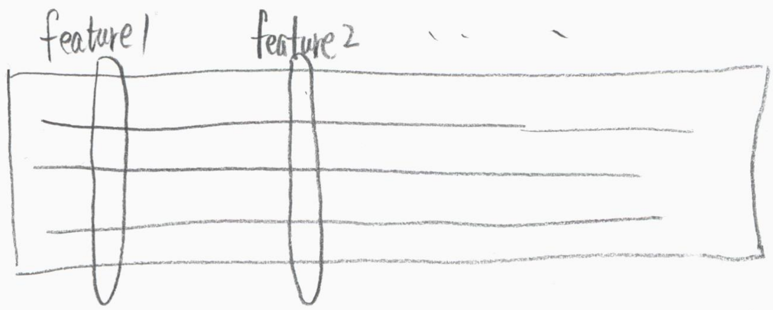
```
## Time difference of 3.366437 mins
```

Akuma.txt (102 features)

① Score function

Suppose we have a partition T now. $T = \{t_1, t_2, \cdots, t_n\}$

t_1 (cluster 1)



$f(t^1) \leftarrow ($ Var1, Var2, . . . , Var102 $)$

t_2 (Cluster 2)

$f(t^2) \leftarrow$ . . .

⋮

Score for cluster 1 : $S(t^1) = f(t^1)^T \times \Lambda$

where $f(t^1) \in R^{102}$, $\Lambda \in R^{102}$

Score for cluster 2 : $S(t^2) = f(t^2) \times \Lambda$

⋮

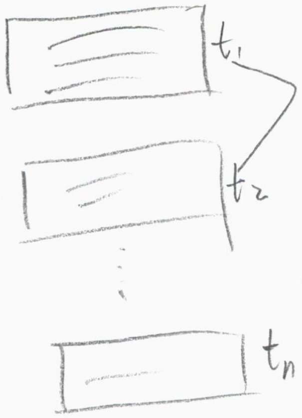
Score for partition T is $S(T) = S(t^1) + S(t^2) + \cdots + S(t^n)$

$$= \left[\sum_i f(t^i)^T \right] \times \Lambda$$

$S(T)$ smaller $\rightarrow$ Partition is better

$S^*(T) =$ The accuracy of T

step x



$\rightarrow$ step $x+1$

$$S(t_1 \cup t_2) + S(t_3) + \cdots S(t_n)$$

$\because S(t_1) + \cdots + S(t_n)$ is fixed

$\therefore$ We just calculate $\underline{S(t_1 \cup t_2) - S(t_1) - S(t_2)}$

that is, choose i, j that minimizes

$$\boxed{S(t_i \cup t_j) - S(t_i) - S(t_j)}$$

And merge t_i and t_j

Problem in our Algorithm

For example, 100 clusters at the begining, use Λ^0

$100 \xleftarrow{\Lambda'}$

99

$\vdots$

Λ' 90 if $\boxed{\begin{matrix} S^*(T^*) > S^*(\hat{T}) \\ \text{tune } \Lambda^0 \to \Lambda' \end{matrix}}$

Λ^k

$\vdots$

Λ^x

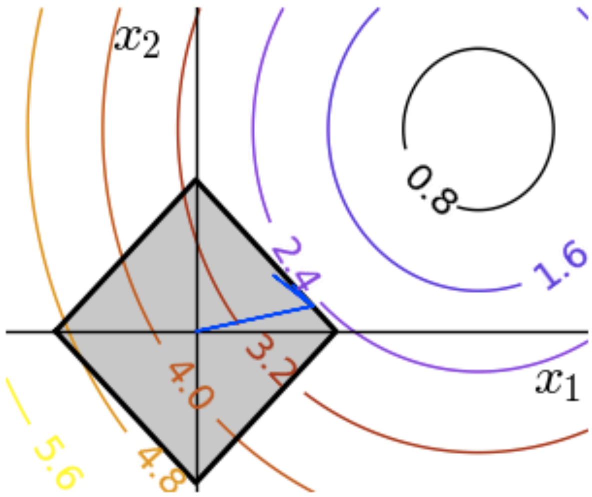
14 Aim

Ranking MIRA We use a variant of MIRA (Margin Infused Relaxed Algorithm), a relaxed, online maximum margin training algorithm (Crammer & Singer 2003). We updates the parameter vector with three constraints: (1) the better neighbor must have a higher score by a given margin, (2) the change to Λ should be minimal, and (3) the inferior neighbor must have a score below a user-defined threshold τ (0.5 in our experiments). The second constraint is to reduce fluctuations in Λ. This optimization is solved through the following quadratic program:

$$\Lambda^{t+1} = \operatorname*{argmin}_{\Lambda} ||\Lambda^t - \Lambda||^2 \text{ s.t.}$$

$$S(N^*(T), \Lambda) - S(\hat{N}(T), \Lambda) \geq 1$$
$$S(\hat{N}, \Lambda) < \tau$$

```
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 2.47487373504
            Iterations: 5
            Function evaluations: 20
            Gradient evaluations: 5
```



Package Scipy in Python

Comparison of performance for two clustering methods					
method	precision	recall	f1	accuracy	time
LDA	0.438	0.253	0.321	0.771	2.578s
error-driven	0.336	0.506	0.404	0.676	65min