Project 4 - Example Main Script

Team 11

04/14/2017

Summary:

In this project, we implemented, evaluated and compared the algorithms in paper 1 : Information Processing and Management (Kang 2009), and paper 5: Author Disambiguation using Error-driven Machine Learning with a Ranking Loss Function(Culotta 2007) for Entity Resolution. We created an author disambiguation system that divides the same-name author occurrences in citation data into different clusters, each of which are expected to correspond to a real individual. We used hierarchical clustering for both papers. In addition, we implemented Cluster Scoring Function, Error-driven Online Training, and Ranking MIRA. After comparing these two methods, ...

Step 0: Load the packages, specify directories

```
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(text2vec, plyr,qlcMatrix, kernlab, knitr)
setwd("~/Spr2017-proj4-team-11/doc")
```

Step 1: Load and process the data

For each record in the dataset, there are some information we want to extract and store them in a regular form: canonical author id, coauthors, paper title, publication venue title.

After generated a list of 14 elements using Professor Zheng's code, we reorganizes it into a list of 14 dataframes for easier access and processing.

```
source("../lib/dataclean.R")
```

Step 2: Feature design

Paper 1 : Following the section 5.2, that each name occurrence is represented by a set of his/her coauthor names. We count the number of matched coauthors between two authors.

Paper 5: We want to use coauthors, paper titles and journey titles to design features for citations. We count the same-coauthor occurrences, and used bigram, trigram, TF-IDF, edit distance. . . to extract features from paper title and journey title.

```
# source("../lib/coauthormatrix.R")
load("../data/sim_matrix.RData")
```

Step 3: Clustering

We used a hierarchical clustering method for both paper 1 and paper 5. The algorithm also follows section 5.2 in paper 1.

Algorithm 2

Agglomerative Clustering for Same-name Author Occurrences

Input:
$a_1,\ldots,a_n$; same-name author occurrences
$a_i = \{v_{i1},\ldots,v_{im}\}$; each name occurrence a_i has a set of m ($m \geqslant 0$) his/her coauthor names
θ; a cluster-merging threshold

Initialize:
$c_i = \{a_i\}$; consider each name occurrence a_i as an element of cluster c_i

Loop:
DO
  For each cluster-pair (c_i, c_j), calculate $CSim(c_i,c_j)$
    $CSim(c_i,c_j) = \max(ASim(a_x, a_y))$, $\forall a_x \in c_i$, $\forall a_y \in c_j$
    $ASim(a_x,a_y) = |a_x \cap a_y|$
  Find the most similar cluster-pair (c_u,c_v)
    $(c_u,c_v) = \text{argmax } CSim(c_i, c_j)$
  IF $CSim(c_u,c_v) \geqslant \theta$ THEN
    $c_{u\_v} = c_u \cup c_v$; merge c_u and c_v into a new larger cluster c_{u_v}
  ENDIF
WHILE $(CSim(c_u,c_v) \geqslant \theta)$

Output:
Clusters of author occurrences: $\{c_k\}$

We set the number of overlapping coauthors to 1. We also considered two scenarios : all the single-element cluster are combined; or we don't combine them.

```
source("../lib/singlelink.R")
start.time <- Sys.time()
cluster_temp.list <- NULL
cluster_temp.list <- llply(simmatrix.list,singlecluster,theta=1)
# load("../doc/cluster_temp_1.RData")
cluster.combined <- NULL
cluster.combined <- llply(cluster_temp.list,comninecluster)
cluster.notcombined <- NULL
cluster.notcombined <- llply(cluster_temp.list,splitcluster)
end.time <- Sys.time()
time_scluter <- end.time - start.time
time_scluter
```

```
## Time difference of 14.03978 mins
```

```
# combind cluster table for AGupta
table(cluster.combined[[1]])
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  62 114  58  24  24  24  23  22  19  16  15  14  13  12  11   9   7   7
##  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
##   7   6   5   5   4   4   4   4   3   3   3   3   3   3   3   3   2   2
##  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

```
# do not combine single-element cluster
table(cluster.notcombined[[1]])
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 114  58  24  24  24  23  22  19  16  15  14  13  12  11   9   7   7   7
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   6   5   5   4   4   4   4   3   3   3   3   3   3   3   3   2   2   2
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 109 110 111 112 113 114 115
##   1   1   1   1   1   1   1
```

Here, I only showed the cluster result of a subset of the data, "AGupta.txt" to further illustrate the difference of these two scenarios.

For paper 5,

Step 4: Evaluation

To evaluate the performance of the method, it is required to calculate the degree of agreement between a set of system-output partitions and a set of true partitions. In general, the agreement between two partitioins is measured for a pair of entities within partitions. The basic unit for which pair-wise agreement is assessed is a pair of entities (authors in our case) which belongs to one of the four cells in the following table (Kang et at.(2009)):

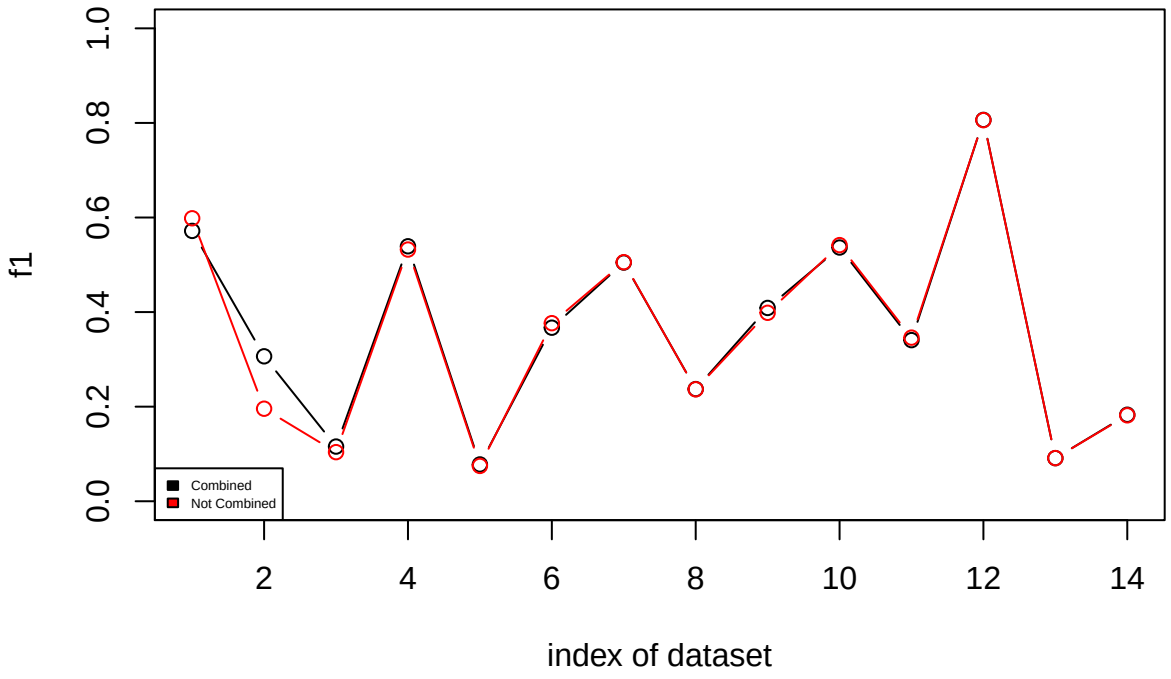
Matching matrix for the agreement between two sets of clusters

		Gold standard clusters (G)	
		Match	Mismatch
Machine-generated clusters (M)	Match	a	b
	Mismatch	c	d

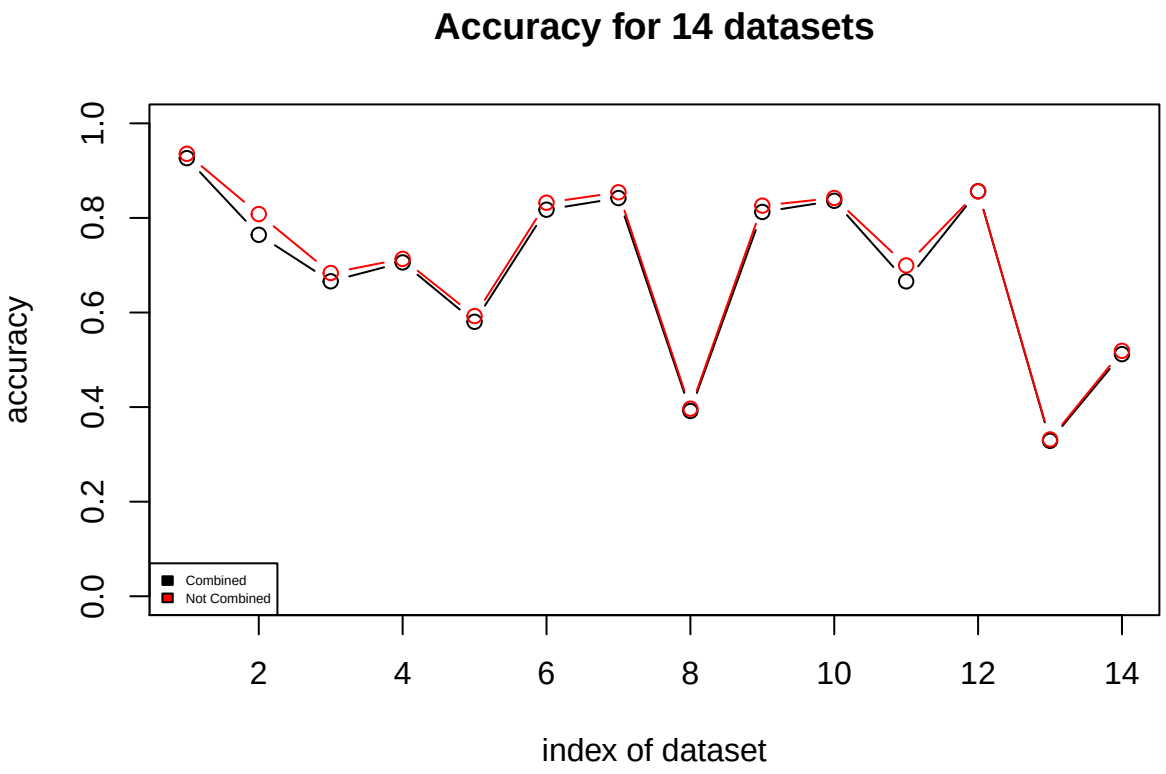
Let M be the set of machine-generated clusters, and G the set of gold standard clusters. Then. in the table, for example, a is the number of pairs of entities that are assigned to the same cluster in each of M and G. Hence, a and d are interpreted as agreements, and b and c disagreements. When the table is considered as a confusion matrix for a two-class prediction problem, the standard "Precision", "Recall","F1", and "Accuracy" are defined as follows.

$$\text{Precision} = \frac{a}{a+b}$$
$$\text{Recall} = \frac{a}{a+c}$$
$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{Accuracy} = \frac{a+d}{a+b+c+d}$$

```r
source("../lib/evaluation_measures.R")

#### paper 1
matching_matrix_single <- NULL
matching_matrix_combined <- NULL
for (i in 1:14){
  matching_matrix_single[[i]] <- matching_matrix(data[[i]],cluster.notcombined[[i]])
  matching_matrix_combined[[i]] <- matching_matrix(data[[i]],cluster.combined[[i]])
}

f1.list.single <- NULL
accuracy.list.single <- NULL
f1.list.combined <- NULL
accuracy.list.combined <- NULL
clustering_errors_single <- NULL
clustering_errors_combined <- NULL
for (i in 1:14){
  f1.list.single[i] <- performance_statistics(matching_matrix_single[[i]])$f1
  f1.list.combined[i] <- performance_statistics(matching_matrix_combined[[i]])$f1
  accuracy.list.single[i] <- performance_statistics(matching_matrix_single[[i]])$accuracy
  accuracy.list.combined[i] <- performance_statistics(matching_matrix_combined[[i]])$accuracy
}

# f1 result plot
plot(f1.list.combined,
     xlab="index of dataset",
     ylab="f1",
     type="b",
     col="1",
     ylim = range(0,1),
     main="f1 result for 14 datasets")
points(f1.list.single,
       type="b",
       col="2")
legend("bottomleft",
       c("Combined","Not Combined"),
       fill=c("1","2"),
       cex=0.45)
```




```r
# accuracy result plot
plot(accuracy.list.combined,
     xlab="index of dataset",
     ylab="accuracy",
     type="b",
     col="1",
     ylim = range(0,1),
     main="Accuracy for 14 datasets")
points(accuracy.list.single,
       type="b",
       col="2")
legend("bottomleft",
       c("Combined","Not Combined"),
       fill=c("1","2"),
       cex=0.45)
```



Step 3: Clustering

Following suggestion in the paper, we carry out spectral clustering on the Gram matrix of the citation vectors by using R function `specc()` in *kernlab*. The number of clusters is assumed known as stated in the paper.