



Project 4 - Group 6

Zeyu Gan, Virgile Mison, Galen Simmons, Siyuan Yao, Qingyuan Zhang

4/14/2017

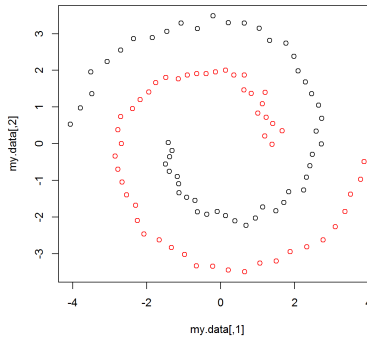
Setup: assign Knitr root directory and load dependencies

We set the `knitr` root.dir to the project directory (`projDir`) and load/install the necessary packages to run our `main.Rmd` script. [Code omitted]

Section I: Paper 3

“Name Disambiguation in Author Citations using a K-way Spectral Clustering Method”

Figure 1: Visual Representation of Spectral Clustering



As discussed in Section 3.1 of the paper, authors Han, Zha, and Giles use **three** citation attributes to design features for name disambiguation. Those attributes are:

- co-author names
- paper titles
- publication venue titles

Together these attributes are called a ‘citation vector.’ For a dataset with m features, each citation is represented as an m -dimensional vector given by $M = (\alpha_1, \dots, \alpha_m)$, where α_i is the weight assigned to feature i . Two types of feature weights assignments are profiled: (i) TFIDF and (ii) normalized TF (“NTF”).

We demonstrate how we create the citation vector from our clean data sources in the `output` library in the chunks below:

Step 1: Load text file

We parse raw data from the `data` folder into the appropriate format using the `Data_Cleaning.R` script contained in the `lib` folder. The parsed datasets are stored in the `output` folder.

```
source(file.path(projDir, "lib", "feature_extraction.R"))
exFile <- file.path(projDir, "output", "A Gupta.csv")
exText <- readTextFile(exFile)
exText %>%
  tbl_df() %>%
  select(QuestAuthor, Coauthor, Paper, Journal) %>%
  head() %>%
  kable(format = "markdown")
```

QuestAuthor	Coauthor	Paper	Journal
A Gupta	NA	Stanford DASH Multiprocessor: The Hardware and Software Approach	PARLE Parallel Architectures and Languages Europe
A Gupta	A Acero; Y Rui	Automatically extracting highlights for TV Baseball programs	ACM Multimedia
A Gupta	A Acharya; M Tambe	Implementation of Production Systems on Message-Passing Computers	IEEE Trans Parallel Distrib Syst
A Gupta	A Agarwal	Memory-Reference Characteristics of Multiprocessor Applications under MACH	SIGMETRICS Measurement and Modeling of Computer Systems
A Gupta	A Agrawal; N Chaddha; T Meng	Variable Compression Using JPEG	ICMCS International Conference Multimedia Computing and Systems
A Gupta	A Balachandran; E Sanocki; G Jancke; J Grudin; J Cadiz	Distance learning through distributed collaborative video viewing	CSCW Conference Computer Supported Cooperative Work

Step 2: Prep the co-author and journal terms to be included in the document term matrix

We want the individual co-author names and journal names to be considered as single ‘terms’ in our corpus. Therefore, we collapse the spaces separating the unique letters in someone’s name, so that it appears to be a term. For example, “C L Zhang” would become “clzhang”. Likewise, with journal names, we combine them into a single string without spaces so that each journal is a unique term in our Document Term Matrix.

```
as_tibble(exText) %>%
  mutate(x = str_replace_all(Coauthor, " ", "")) %>%
  mutate(x = str_replace_all(x, ";", " ")) %>%
  mutate(y = str_replace_all(Journal, " ", "")) %>%
  mutate(term_col = tolower(paste(x, y, Paper))) %>%
  select(term_col) %>%
  head() %>%
  kable(format = 'markdown')
```

term_col

na parleparallelarchitecturesandlanguageeseurope stanford dash multiprocessor: the hardware and software approach

aacero yrui acmmultimedia automatically extracting highlights for tv baseball programs

aacharya mtambe ieeetransparalleldistribsys implementation of production systems on message-passing computers

aagarwal sigmetricsmeasurementandmodelingofcomputersystems memory-reference characteristics of multiprocessor applications under mach

aagrawal nchaddha tmeng icmcsinternationalconferencemultimediacomputingandsystems variable compression using jpeg

abalachandran esanocki gjancke jgrudin jcadiz cscwconferencecomputersupportedcooperativework

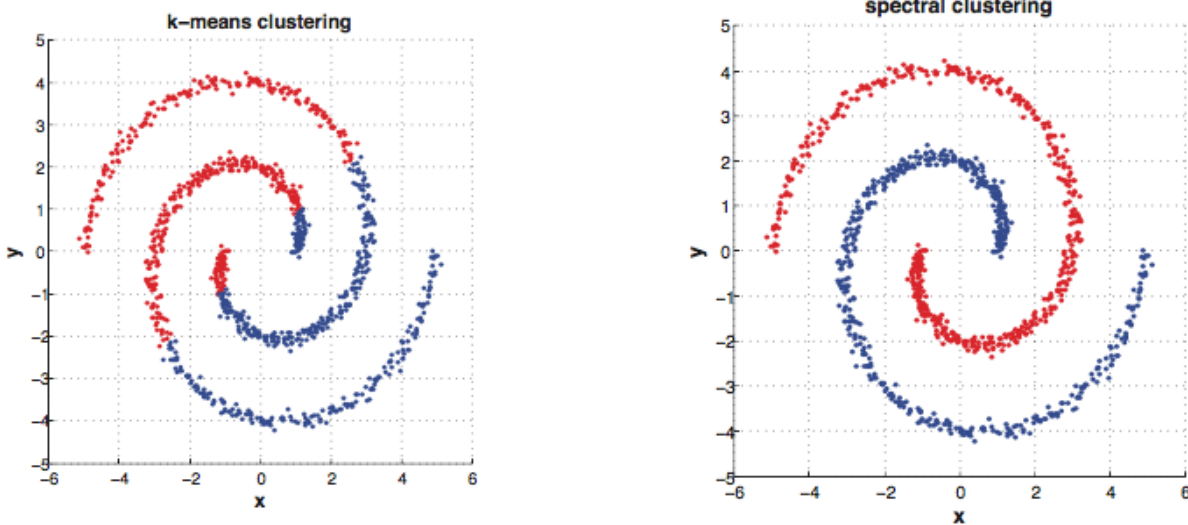
distance learning through distributed collaborative video viewing

Step 3: Run our spectral clustering method on the example dataset

We use the `tm` package to construct citation vectors in the manner described above. We then use the `matching_matrix` and `performance_statistics` functions in the `lib/evaluation_metrics.R` file to benchmark our results. Below we describe the details of our spectral clustering implementation. Our spectral clustering implementation is contained in `lib/SpectralClustering.R` and includes the following key methods:

- **affinity** : creates an affinity matrix
- **simFunction** : similarity function
- **similarity** : creates a similarity matrix from a similarity kernel
- **specClusteringKM** : method for assigning clusters using k-means
- **specClusteringQR** : method for assigning clusters based on cosine similarities using QR decomposition with pivoting

Figure 2: Graphical comparison of kmeans vs spectral clustering



Details of Gaussian-similarity-kernel and k-means clustering implementation

First, we compute the similarity between citations from the TF-IDF or NTF matrix of citations. We use a Gaussian kernel as a measure of similarity. Then, we create an undirected graph based on the similarities to extract some manifold in the data, we thereby obtain A , the affinity matrix. After, we calculate the degree matrix D (diagonal) where each diagonal value is the degree of the respective vertex (*e.g.* sum of rows).

We compute the unnormalized graph Laplacian: $U = D - A$. Then, assuming that we want k clusters, we find the k smallest eigenvectors of U . This represents the points in a new k -dimensional space with low-variance. Finally, in this transformed space, it becomes easy for a standard k-means clustering to find the appropriate clusters.

Gram Matrix

From the TF-IDF or NTF matrix of citations, we compute the cosine similarity between each citation vectors as follows:

$$similarity = \cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

This matrix is called the **Gram matrix** A . In the first step of the algorithm, we determine the k largest eigenvectors of A : X_k , a n -by- k matrix. Each row of X_k corresponds to a citation vector. Then, we compute the **QR decomposition with column pivoting** applied to X_k^T , *e.g.* we find the matrices P (permutation matrix, n -by- n), Q (orthogonal, k -by- k) and R (left-upper-triangular, k -by- n), so that:

$$X_k^T P = QR = Q[R_{11}, R_{12}]$$

R_{11} will be the k -by- k upper-triangular matrix. We then compute the matrix \hat{R} :

$$\hat{R} = R_{11}^{-1} R P^T = R_{11}^{-1} [R_{11}, R_{12}] P^T = [I_k, R_{11}^{-1} R_{12}] P^T$$

Finally, the cluster membership of each citation vector is determined by the row index of the largest element in absolute value of the corresponding column of \hat{R} .

Example study using methods in our lib/SpectralClustering.R script

For reproducibility, we avoid computing k-means in this Rmd file because of running time. We run this study on the Normalized Term Frequencies (NTF) and the TF-IDF objects we extract in the lib/feature_extraction.R script.

```
source(file.path(projDir,"lib","SpectralClustering.R"))
source(file.path(projDir,"lib","evaluation_measures.R"))

# estimated time 6 sec
start.time <- Sys.time()

num_authors <- length(unique(exText$AuthorID))
author_id <- exText$AuthorID
exDTM <- createCitationMatrix(exText)

exTFIDF <- weightTfIdf(exDTM, normalize = FALSE)
exNTF <- weightTf(exDTM)

exResultsTFIDF <- runSpectralClusteringQR(exTFIDF, num_authors)
exResultsNTF <- runSpectralClusteringQR(exNTF, num_authors)
```

```

m0 <- matching_matrix(author_id,exResultsTFIDF)
m1 <- matching_matrix(author_id,exResultsNTF)

p0 <- performance_statistics(m0)
p1 <- performance_statistics(m1)

end.time <- Sys.time()
study.time <- end.time - start.time

kable(data.frame(study_time = study.time), format = "markdown")

```

study_time
4.444655 secs

Step 4: View results from sample study

We examine some results from our sample methods to confirm our methodology is producing reasonable results.

```

ex_df <- data.frame(
  study=rep("Agupta", 2),
  method=c("QR Spectral Clustering - TFIDF",
            "QR Spectral Clustering - NTF"),
  precision=c(p0$precision, p1$precision),
  recall=c(p0$recall, p1$recall),
  f1=c(p0$f1, p1$f1),
  accuracy=c(p0$accuracy, p1$accuracy),
  mcc=c(p0$mcc, p1$mcc)
)

kable(ex_df, format = "markdown", digits = 2)

```

study	method	precision	recall	f1	accuracy	mcc
Agupta	QR Spectral Clustering - TFIDF	0.13	0.25	0.17	0.76	0.05
Agupta	QR Spectral Clustering - NTF	0.11	0.28	0.16	0.70	0.02

NOTE: Notice the f1 score. The F1 score can be interpreted as a weighted average of the precision and recall (with both weighted equally), where an F1 score (i.e. $\beta = 1$) reaches its best value at 1 and worst at 0. It is given by:

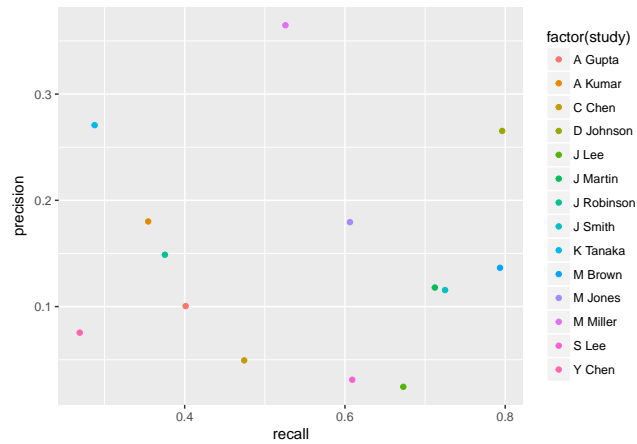
$$F_{\beta} = \frac{(1 + \beta^2) \cdot \text{true positive}}{(1 + \beta^2) \cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}}$$

Step 5: Load results from all studies

Separately, we have run our spectral clustering methods on all of our cleaned and labeled datasets in `output` and saved the results in `output/studies`. [code omitted] We plot the results of our studies below.

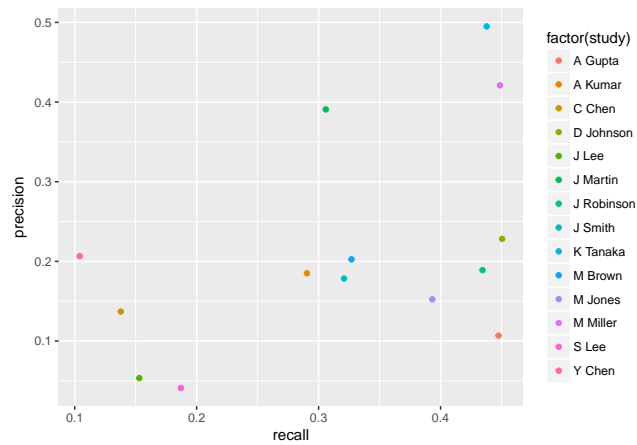
Precision-Recall Scatterplot of Spectral Clustering with QR Decomposition

```
ggplot(data=tb_qr, aes(x=recall, y=precision, col=factor(study))) + geom_point()
```



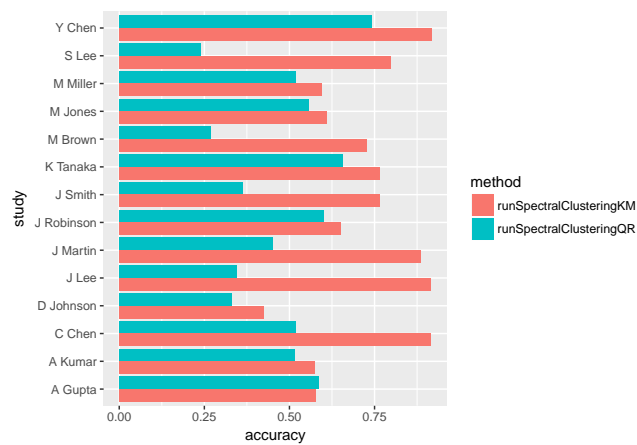
Precision-Recall Scatterplot of Spectral Clustering KMeans

```
ggplot(data=tb_km, aes(x=recall, y=precision, col=factor(study))) + geom_point()
```



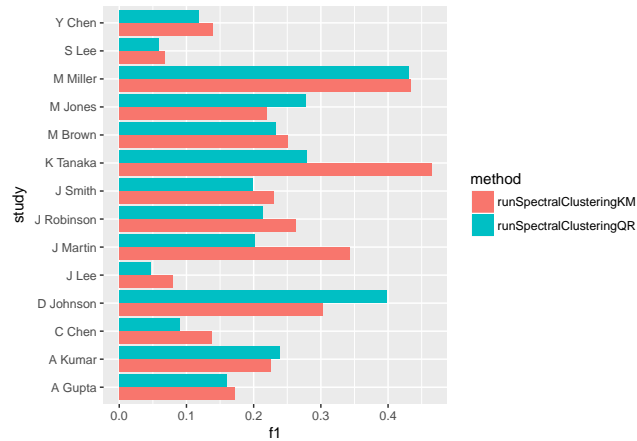
Comparison of Accuracy of Spectral Clustering with QR Decomposition vs Spectral Clustering KMeans

```
ggplot(data=tb_join, aes(x=study, y=accuracy, fill=method)) +  
  geom_bar(stat="identity", position=position_dodge()) + coord_flip()
```



Comparison of F1 score of Spectral Clustering with QR Decomposition vs Spectral Clustering KMeans

```
ggplot(data=tb_join, aes(x=study, y=f1, fill=method)) +
  geom_bar(stat="identity", position=position_dodge()) + coord_flip()
```



General Notes on Spectral Clustering vs K-Means using Term Frequency:

Why TF-IDF? Term frequency counts the number of times a word from a corpus appears in a particular document. Some words appear frequently in all documents (i.e. they are common locally). Other words appear rarely in the corpus (i.e. they are rare globally). Inverse document frequency weights words that appear rarely more heavily, implying that rare words are more important for determining the cluster assignment of a particular document than words that are common to all documents in a corpus.

Distance measures

Distance measures are the way convergence is measured. The selection of distance measures is very important to consider because different clustering problems require different measures of similarity. In fact, while researching distances, we discovered that there is a textbook called “Encyclopedia of Distances” by Michael Deza and Elena Deza which contains helpful information on the different ways of selecting the correct distance metric for a particular problem. Below are two common measures of distance.

Scaled Euclidean Distance:

$$d(x_i, x_q) = \sqrt{a_1(x_i[1] - x_q[1])^2 + \dots + a_d(x_i[d] - x_q[d])^2}$$

Cosine Similarity:

$$\frac{x_i^T x_q}{\|x_i\| \|x_q\|} = \cos(\theta)$$

K-means algorithm

1. Initialize cluster centers
2. Assign observations to closest cluster center (hard assignment)

3. Revise cluster centers as mean of assigned observations
4. Repeat steps 1 & 2 until convergence

Figure 3: Kmeans pseudocode

Algorithm 11.1: K-means algorithm

```

1 initialize  $\mathbf{m}_k$ ;
2 repeat
3   Assign each data point to its closest cluster center:  $z_i = \arg \min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$ ;
4   Update each cluster center by computing the mean of all points assigned to it:
      $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i: z_i=k} \mathbf{x}_i$ ;
5 until converged;
```

Complexity of brute-force search

Given a query point, $O(N)$ distance computations per 1-NN query. By extension, $O(N \log k)$ distance computations per k -NN query.

Failure modes of k-means

- disparate cluster sizes
- overlapping clusters
- different shaped / oriented clusters

In the most basic implementation, k-means tries to make cluster assignments of documents using euclidean distances based on distances from cluster centers:

$$z_i \leftarrow \arg \min \|u_j - \mathbf{x}_i\|_2^2$$

This method is problematic because only the center of the cluster matters. In other words, this method assumes that clusters are spherically symmetric. Although weighted Euclidean distances could be used, we would need to know which weights were appropriate, and even with weights, the clusters would still have the same axis-aligned ellipses.

In the author disambiguation problem, we are faced with a conundrum, namely that kmeans generally fails to give good results when the distance metric used produces ‘overlapping’ clusters. Since authors that share names may work in related fields, this type of disambiguation problem calls for a more probabilistic approach. This is the motivation for using ‘mixture models,’ which enables clustering based on **learned weightings** of features when computing cluster assignments.

Section II: Paper 6

“A Constraint-Based Probabilistic Framework for Name Disambiguation”

In this paper, Zhang, Tang, Li, and Wang propose a mixture model using the expectations maximization algorithm to perform name disambiguation. They propose a ‘semi-supervised’ framework to “combine the advantages of supervised and unsupervised methods.” What does this mean in practice? First, we randomize the weights we assign to the constraints per **section 3.2**. Then, we try to use EM to minimize our objective function with two weighted constraints. The steps in the EM algorithm can be summarized at a high level as follows:

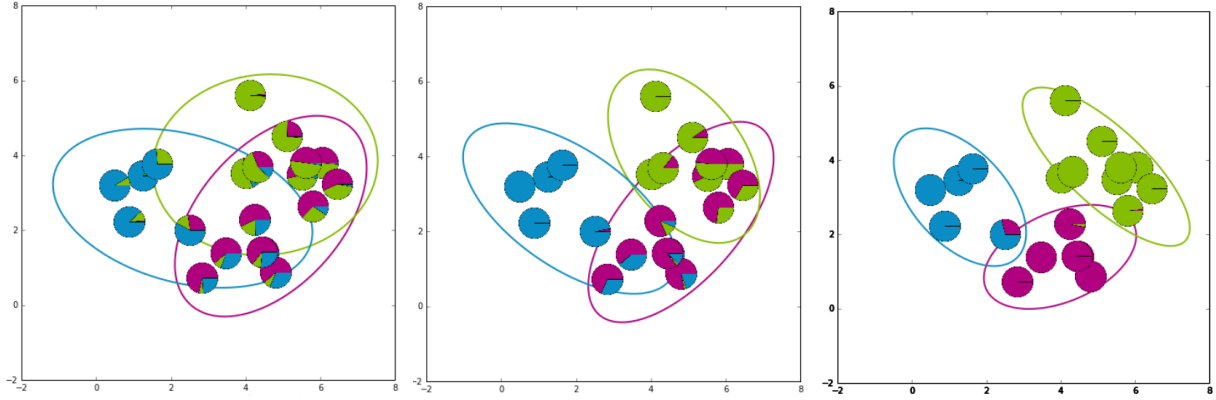
Figure 4: EM pseudocode

Algorithm 11.3: Incremental EM algorithm

```

1 initialize  $\mathbf{s}_i$  for  $i = 1 : N$ ;
2  $\boldsymbol{\mu} = \sum_i \mathbf{s}_i$ ;
3 repeat
4   for each example  $i = 1 : N$  in a random order do
5      $\mathbf{s}_i^{new} := \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}_i, \boldsymbol{\theta}(\boldsymbol{\mu})) \phi(\mathbf{x}_i, \mathbf{z})$ ;
6      $\boldsymbol{\mu} := \boldsymbol{\mu} + \mathbf{s}_i^{new} - \mathbf{s}_i$ ;
7      $\mathbf{s}_i := \mathbf{s}_i^{new}$ ;
8 until converged;
```

In the EM algorithm, $p(z_i = k|x_i, \theta)$ represents the posterior probability that point i belongs to cluster k . This is known as the responsibility of cluster k for point i , and can be computed using Bayes rule. This procedure is called *soft clustering*. EM is an iterative algorithm which alternates between inferring the missing values given the parameters (E step), and then optimizing the parameters given the “filled in” data (M step). An illustration of this iterative process is shown below:

**Table 2 constraints:**

c_2 for any two papers, if they share a co-author, c_2 is 1, otherwise is 0 c_6 if co-authors of any two papers are also

Constraints: The constraint function is a Boolean-valued function and is defined as follow:

$$c(p_i, p_j) = \begin{cases} 1, & \text{if } p_i \text{ and } p_j \text{ satisfy the constraint } c_l \\ 0, & \text{otherwise} \end{cases}$$

Goal is to minimize the objective function by using EM algorithm. **Objective Function:**

$$\max p(Y|X) \propto \min f_{obj} = \sum_i \sum_j \{D(x_i, x_j) I(l_i \neq l_j) \sum_{c_k \in C} [w_k c_k(p_i, p_j)]\} + \sum_{x_i \in X} D(x_i, y_{l_i}) + \log Z$$

\ where, \ \$ D(x_i, y_{l_i}) \$ is the distance between paper i and researcher y_{l_i} \ w_k is the weight of for c_k \ Z is the normalization factor \

We define six types of constraints and formalize them in the framework. We propose employing EM algorithm to learn different distance metric for different persons.

The rest of the paper is organized as follows. In Section 2, we review related work. In Section 3, we formalize the disambiguation problem. In Section 4, we explain our approach to the problem and in Section 5, we give the experimental results. We conclude the paper in Section 6.

Our method is based on a unified probabilistic model using Hidden Markov Random Fields (HMRF). This model incorporates constraints and a parameterized-distance measure. The disambiguation problem is cast as assigning a tag to each paper with each tag representing an actual researcher y_i .

a-posteriori probability as the objective function

We aim at finding the maximum of the objective

Six types of constraints are incorporated into the objective function, where constraints are considered as a form of supervision. If one paper's label assignment violates a constraint, it will be penalized in some sense, which in turn affects the result.

The EM process can be summarized as follows: in the E-step, given the current researcher representatives (the set of assigned papers), every paper is re-assigned to the researcher by maximizing $p(Y|X)$. In the M-step, the researcher representative y_h is re-estimated from the assignments to maximize $p(Y|X)$ again, and the distance measure is updated to increase the objective function.

In the E-step, assignments of data points to researchers are updated to maximize the $p(Y|X)$. A greedy algorithm is used to sequentially update the assignment of each paper

Three tasks are executed by the Expectation Maximization (EM) method: learning parameters of the distance measure, re-assignment of paper to researchers, and update of researcher representatives y_k . We define our distance function $D(x_i, x_j)$ as follows:

$$D(x_i, x_j) = 1 - \frac{x_i^T \mathbf{A} x_j}{\|x_i\|_A \|x_j\|_A}$$

, where

$$\|x_i\|_A = \sqrt{x_i^T \mathbf{A} x_i}$$

Each paper x_i is assigned to y_h that minimizes the function:

$$f(y_h, x_i) = \sum_i D(x_i, y_h) + \sum_{i,j \neq i} \{D(x_i, x_j) \sum_{c_k \notin C} [w_k c_k(x_i, x_j)]\}$$

In the M-step, each researcher representative is updated by the arithmetic mean of its points:

$$y_h = \frac{\sum_{i:l_i=h} x_i}{\left\| \sum_{i:l_i=h} x_i \right\|_A}$$

Step 1: Load and process the data

```
AKumar <- read.csv(file.path(projDir,"output","AKumar.csv"))
AGupta <- read.csv(file.path(projDir,"output","Agupta.csv"))
CChen <- read.csv(file.path(projDir,"output","CChen.csv"))
DJohnson <- read.csv(file.path(projDir,"output","DJohnson.csv"))
JLee <- read.csv(file.path(projDir,"output","JLee.csv"))
JMartin <- read.csv(file.path(projDir,"output","JMartin.csv"))
JRobinson <- read.csv(file.path(projDir,"output","JRobinson.csv"))
JSmith <- read.csv(file.path(projDir,"output","JSmith.csv"))
KTanaka <- read.csv(file.path(projDir,"output","KTanaka.csv"))
YChen <- read.csv(file.path(projDir,"output","YChen.csv"))
Slee <- read.csv(file.path(projDir,"output","Slee.csv"))
MMiller <- read.csv(file.path(projDir,"output","MMiller.csv"))
```

```
MJones <- read.csv(file.path(projDir, "output", "MJones.csv"))
MBrown <- read.csv(file.path(projDir, "output", "MBrown.csv"))

# Write a function to calculate the DTM for paper
dtm_paper <- function(df) {
  it_train <- itoken(as.character(df$Paper),
    preprocessor = tolower,
    tokenizer = word_tokenizer,
    ids = nrow(df),
    # turn off progressbar because it won't look nice in rmd
    progressbar = FALSE)

  vocab <- create_vocabulary(it_train, stopwords = c("a", "an", "the", "in", "on", "at", "of", "above", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58", "59", "60", "61", "62", "63", "64", "65", "66", "67", "68", "69", "70", "71", "72", "73", "74", "75", "76", "77", "78", "79", "80", "81", "82", "83", "84", "85", "86", "87", "88", "89", "90", "91", "92", "93", "94", "95", "96", "97", "98", "99", "100", "101", "102", "103", "104", "105", "106", "107", "108", "109", "110", "111", "112", "113", "114", "115", "116", "117", "118", "119", "120", "121", "122", "123", "124", "125", "126", "127", "128", "129", "130", "131", "132", "133", "134", "135", "136", "137", "138", "139", "140", "141", "142", "143", "144", "145", "146", "147", "148", "149", "150", "151", "152", "153", "154", "155", "156", "157", "158", "159", "160", "161", "162", "163", "164", "165", "166", "167", "168", "169", "170", "171", "172", "173", "174", "175", "176", "177", "178", "179", "180", "181", "182", "183", "184", "185", "186", "187", "188", "189", "190", "191", "192", "193", "194", "195", "196", "197", "198", "199", "200", "201", "202", "203", "204", "205", "206", "207", "208", "209", "210", "211", "212", "213", "214", "215", "216", "217", "218", "219", "220", "221", "222", "223", "224", "225", "226", "227", "228", "229", "230", "231", "232", "233", "234", "235", "236", "237", "238", "239", "240", "241", "242", "243", "244", "245", "246", "247", "248", "249", "250", "251", "252", "253", "254", "255", "256", "257", "258", "259", "260", "261", "262", "263", "264", "265", "266", "267", "268", "269", "270", "271", "272", "273", "274", "275", "276", "277", "278", "279", "280", "281", "282", "283", "284", "285", "286", "287", "288", "289", "290", "291", "292", "293", "294", "295", "296", "297", "298", "299", "300", "301", "302", "303", "304", "305", "306", "307", "308", "309", "310", "311", "312", "313", "314", "315", "316", "317", "318", "319", "320", "321", "322", "323", "324", "325", "326", "327", "328", "329", "330", "331", "332", "333", "334", "335", "336", "337", "338", "339", "340", "341", "342", "343", "344", "345", "346", "347", "348", "349", "350", "351", "352", "353", "354", "355", "356", "357", "358", "359", "360", "361", "362", "363", "364", "365", "366", "367", "368", "369", "370", "371", "372", "373", "374", "375", "376", "377", "378", "379", "380", "381", "382", "383", "384", "385", "386", "387", "388", "389", "390", "391", "392", "393", "394", "395", "396", "397", "398", "399", "400", "401", "402", "403", "404", "405", "406", "407", "408", "409", "410", "411", "412", "413", "414", "415", "416", "417", "418", "419", "420", "421", "422", "423", "424", "425", "426", "427", "428", "429", "430", "431", "432", "433", "434", "435", "436", "437", "438", "439", "440", "441", "442", "443", "444", "445", "446", "447", "448", "449", "450", "451", "452", "453", "454", "455", "456", "457", "458", "459", "460", "461", "462", "463", "464", "465", "466", "467", "468", "469", "470", "471", "472", "473", "474", "475", "476", "477", "478", "479", "480", "481", "482", "483", "484", "485", "486", "487", "488", "489", "490", "491", "492", "493", "494", "495", "496", "497", "498", "499", "500", "501", "502", "503", "504", "505", "506", "507", "508", "509", "510", "511", "512", "513", "514", "515", "516", "517", "518", "519", "520", "521", "522", "523", "524", "525", "526", "527", "528", "529", "530", "531", "532", "533", "534", "535", "536", "537", "538", "539", "540", "541", "542", "543", "544", "545", "546", "547", "548", "549", "550", "551", "552", "553", "554", "555", "556", "557", "558", "559", "560", "561", "562", "563", "564", "565", "566", "567", "568", "569", "570", "571", "572", "573", "574", "575", "576", "577", "578", "579", "580", "581", "582", "583", "584", "585", "586", "587", "588", "589", "590", "591", "592", "593", "594", "595", "596", "597", "598", "599", "600", "601", "602", "603", "604", "605", "606", "607", "608", "609", "610", "611", "612", "613", "614", "615", "616", "617", "618", "619", "620", "621", "622", "623", "624", "625", "626", "627", "628", "629", "630", "631", "632", "633", "634", "635", "636", "637", "638", "639", "640", "641", "642", "643", "644", "645", "646", "647", "648", "649", "650", "651", "652", "653", "654", "655", "656", "657", "658", "659", "660", "661", "662", "663", "664", "665", "666", "667", "668", "669", "670", "671", "672", "673", "674", "675", "676", "677", "678", "679", "680", "681", "682", "683", "684", "685", "686", "687", "688", "689", "690", "691", "692", "693", "694", "695", "696", "697", "698", "699", "700", "701", "702", "703", "704", "705", "706", "707", "708", "709", "710", "711", "712", "713", "714", "715", "716", "717", "718", "719", "720", "721", "722", "723", "724", "725", "726", "727", "728", "729", "730", "731", "732", "733", "734", "735", "736", "737", "738", "739", "740", "741", "742", "743", "744", "745", "746", "747", "748", "749", "750", "751", "752", "753", "754", "755", "756", "757", "758", "759", "760", "761", "762", "763", "764", "765", "766", "767", "768", "769", "770", "771", "772", "773", "774", "775", "776", "777", "778", "779", "780", "781", "782", "783", "784", "785", "786", "787", "788", "789", "790", "791", "792", "793", "794", "795", "796", "797", "798", "799", "800", "801", "802", "803", "804", "805", "806", "807", "80
```

Then, we want to use DTM to compute TF-IDF transformation on DTM.

```

dtm_paper_tfidf <- function(df) {
  tfidf <- TfIdf$new()
  return(fit_transform(dtm_paper(df), tfidf))
}

dtm_coauthor_tfidf <- function(df) {
  tfidf <- TfIdf$new()
  return(fit_transform(dtm_coauthor(df), tfidf))
}

dtm_journal_tfidf <- function(df) {
  tfidf <- TfIdf$new()
  return(fit_transform(dtm_journal(df), tfidf))
}

# We want to combine the three citations together
dtm_tfidf <- function(df) {
  total <- cbind(dtm_paper_tfidf(df), dtm_coauthor_tfidf(df), dtm_journal_tfidf(df))
  return(total)
}

# pap <- dtm_paper_tfidf(AGupta)
# coa <- dtm_coauthor_tfidf(AGupta)
# jor <- dtm_journal_tfidf(AGupta)

## compute cosine similarities matrix (Gram Matrix)
docsdissim <- function(df) {
  cosSparse(t(dtm_tfidf(df)))
}

```

Step 3: Clustering

Following suggestion in the paper 6, we carry out hierarchical clustering method under the cosine distance as the baseline method. The number of clsters is assumed known as stated in the paper.

```

basemodel <- function(df) {
  cossim <- docsdissim(df)
  rownames(cossim) <- c(1:nrow(dtm_tfidf(df)))
  colnames(cossim) <- c(1:nrow(dtm_tfidf(df)))

  h <- hclust(as.dist(cossim), method = "ward.D") #
  result_hclust <- cutree(h, length(unique(df$AuthorID))) # {stats} package
  return(result_hclust)
}

basemodel_AGupta <- basemodel(AGupta)
basemodel_AKumar <- basemodel(AKumar)
# start.time <- Sys.time()
# cossim <- docsdissim(AGupta)
# rownames(cossim) <- c(1:nrow(dtm_tfidf(AGupta)))
# colnames(cossim) <- c(1:nrow(dtm_tfidf(AGupta)))
# compute pairwise cosine similarities using cosSparse function in package qIcMatrix
# h <- hclust(as.dist(cossim), method = "ward.D")

```

```

# result_hclust <- cutree(h,length(unique(AGupta$AuthorID)))
# end.time <- Sys.time()
# time_hclust <- end.time - start.time
# table(result_hclust)

matching_matrix_hclust_AGupta <- matching_matrix(AGupta$AuthorID,basemodel_AGupta)
performance_hclust_AGupta <- performance_statistics(matching_matrix_hclust_AGupta)

matching_matrix_hclust_AKumar <- matching_matrix(AKumar$AuthorID,basemodel_AKumar)
performance_hclust_AKumar <- performance_statistics(matching_matrix_hclust_AKumar)

compare_df <- data.frame(author=c("AGupta","AKumar"),
                          precision=c(performance_hclust_AGupta$precision, performance_hclust_AKumar$pre
                          recall=c(performance_hclust_AGupta$recall, performance_hclust_AKumar$recall),
                          f1=c(performance_hclust_AGupta$f1, performance_hclust_AKumar$f1),
                          accuracy=c(performance_hclust_AGupta$accuracy, performance_hclust_AKumar$accur

kable(compare_df, format = "markdown", caption="Comparision of performance for two clustering methods",

```

author	precision	recall	f1	accuracy
AGupta	0.08	0.04	0.05	0.86
AKumar	0.19	0.07	0.10	0.74