《数据仓库与数据挖掘技术》作业报告

RankClus 算法实现

马晋
1400012186
ma_jin@pku.edu.cn

信息科学技术学院

2018 年 6 月 16 日

摘 要

本次报告基于 Python 语言实现了 Yizhou Sun 论文《RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis》中提出的在异构信息网络中基于排名的的无监督聚类算法 RankClus，并在论文中提到的 DBLP（DataBase systems and Logic Programming）数据集上测试了算法性能。

关键词 ： RankClus 数据挖掘 无监督聚类 异构网络

目录

1 算法概述

1.1 算法介绍

RankClus 算法是 Yizhou Sun 和 Jiawei Han 等人在 EDBT 2009 上发表的论文 RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis 中提出的在异构信息网络中基于排名的的无监督聚类算法。

关于异构网络，是指在同一个网络内，内部节点的类型不完全相同，而这些节点之间的关系既可以是同类节点之间，也可以是跨越不同类型的节点之间。对于这种网络，我们需要对节点之间的相似度做出恰当的定义，以同时利用到不同节点和不同连接的信息。

论文的问题背景是作者-刊物的异构网络，这是一种双类型异构网络，即其中含有两种节点。主要解决的问题是，通过作者与刊物的排名信息，利用“不同研究领域作者的排名信息相差很大”这一特征，作者的排名和刊物的排名相互迭代，并通过迭代计算得到的排名来定义相似度，最终来调整刊物的聚类。

论文和我的实验显示，在 DBLP 数据集上，RankClus 能够胜任异构网络中的数据挖掘，可以在作者-刊物双异构网络中无监督地找出最相似地会议集合进行聚类，值得关注的是，整个聚类过程中并没有用到在其他数据聚类中常用到的任何关键词信息或引用信息，这也证明了异构信息网络中包含有更多的信息。

随着数据挖掘领域研究地不断推进，数据的组织结构也向着复杂化、异构化的方向发展。异构网络可以将同构的信息网络连接起来，并且包含了更多的信息，这也意味着可以发掘出更多的有效模式，是数据挖掘领域的研究热点。同时我们也注意到现实生活中的很多信息天然就是以异构信息网络的形式存在，RankClus 这种可以在异构网络中挖掘模式的算法一定会在实际应用诸如推荐系统中大放异彩。

1.2 相关定义

Definition 1. 双类型信息网络（*Bi-type Information Network*）：给定两个类型的对象集合 X 和 Y，其中 $X=\{x_1,x_2,\dots,x_m\}$，$Y=\{y_1,y_2,\dots,y_n\}$，若 $V(G)=X\cup Y$ 并且 $E(G)=\{<o_i,o_j>\}$，$o_i,o_j\in X\cup Y$，我们称 G 为一个双类型信息网络。

为了描述 G 中各连接的权值信息，引入矩阵 W，为了方便，我们将 W 分为四个部分：

$$W=\begin{pmatrix} W_{XX} & W_{XY} \\ W_{YX} & W_{YY} \end{pmatrix}$$

Definition 2. 排名函数（*Ranking Function*）：给定一个双类型信息网络 $G=<\{X\cup Y\},W>$，函数 $f: G\to(\overrightarrow{r_X},\overrightarrow{r_Y})$，对每个 X 或 Y 节点给出一个排名分数，并且 $\overrightarrow{r_X}$ 非负对 X 求和为 1，$\overrightarrow{r_Y}$ 非负对 Y 求和为 1，我们称 f 为 G 的排名函数。

1.3 算法描述

这里对算法进行概览式描述，具体细节与实现在 2.2 中。

输入：双类型异构信息网络 $G=<\{X\cup Y\},W>$，X 类型和 Y 类型上的排名函数和聚类数目 K。

输出：类型 X 的 K 个聚类，且每个聚类内对其中的 X 节点和 Y 节点带有重要性排名值。

- 步骤一：随机对每个 X 节点赋聚类标签，得到初始随机聚类。
- 步骤二：在每个聚类条件下利用排名函数得到每个 X 节点和 Y 节点在不同聚类下的排名值。
- 步骤三：得到每个聚类条件下的排名后，我们需要对每个聚类的大小做出估计，即每个聚类的先验概率 $p(k)$。这里使用 EM 算法，对这个分布 $p(k)$ 引入参数 θ 进行更新，使 θ 下产生 W_{XY} 的似然最大，得到 $p(k)$ 分布。最后基于贝叶斯公式得到每个 X 节点属于每个聚类的概率分布 K 维向量。
- 步骤四：以三中得到的 K 维向量为特征，得到每个聚类的聚类

中心，再通过计算每个 X 节点与各聚类中心的相似度对其所属的聚类进行调整。

- 步骤五：检查是否存在空聚类，如果存在回到步骤一，如果不存在回到步骤二迭代进行。
- 步骤六：得到最后的聚类结果。

改进：我在实现代码中排名函数

算法伪代码如下：

Table 4: RankClus Algorithm

Procedure: RankClus()
Input: Bi-type Information Network $G = \langle X, Y; W \rangle$,
Ranking function f,
Cluster Number K.
Output: K clusters X_i, $\vec{r}_{X_i|X_i}$, $\vec{r}_{Y|X_i}$.

//Step 0: Initialization
$t = 0$;
$\{X_i^{(t)}\}_{i=1}^{K}$ = get initial partitions for X;
//Repeat Steps 1-3 until $< \varepsilon$ change or too many iterations
For (iter = 0; iter < iterNum && epsi > ε; iter++)
//Step 1: Ranking for each cluster
if any of the clusters are empty, restart, goto Line 1;
For i = 1 to K
$G_i^{(t)}$ = get subgraph from G, using $X_i^{(t)}$, Y;
$(\vec{r}_{X_i|X_i}^{(t)}, \vec{r}_{Y|X_i}^{(t)}) = f(G_i^{(t)})$; $\vec{r}_{X|X_i}^{(t)} = W_{XY}\vec{r}_{Y|X_i}^{(t)}$;
End for
//Step 2: Get new attributes for objects and cluster
Evaluate Θ for mixture model, thus get $\vec{s}_{x_i}$ for each object x_i;
For i = 1 to K
$\vec{s}_{X_k}^{(t)}$ = get centers for cluster $X_k^{(t)}$;
End for
//Step 3: Adjust each object
For each object x in X
For i = 1 to K
Calculate Distance $D(x, X_k^{(t)})$
End for
Assign x to $X_{k_0}^{t+1}$, $k_0 = \arg\min_k D(x, X_k^{(t)})$
End for
End For

2 算法实现

2.1 数据集

选用了论文中所使用的 DBLP（DataBase systems and Logic Programming）数据集进行效果测试，特别的，选取了其中 2008-2017 年的作者-刊物数据，并剔除了发表刊物数少于 10 的作者。最终数据集包含 13332 个作者，5672 个刊物，共 239243 篇论文数据。处理后的数据格式如下：

data_solved.txt：publication, author_1, author_2, …, author_n

2.2 实现模块

（实现代码和数据已 git 至 https://github.com/Bsdnbo/RankClus）

按照 RankClus 算法的步骤，我将整体分为如下几个部分：

- 数据读取与清洗（dataParser.py）：由于 DBLP 原始数据集大 2.15GB，格式为 xml 文件，这么大的数据一次放入内存显然不是最好的选择，我的处理方法是将 xml 文档以 txt 文件的格式读取，通过判断特定标签字符串来获取年份（<year>）、刊物名（<booktitle>）和作者（<author>），<inproceedings>条目为会议论文集中的一篇文章，遇到这个标签便需要开始注意检索刊物名、作者和年份标签。

  最终数据在 csv 文件中每一篇论文占一行，每一行第一列为刊物名称，后面依次是本篇论文的作者，这些作者之间是合作关系，他们和刊物之间是参与关系。

  为了去掉一些无效的数据，比如发论文数太少的作者，由于他们的信息较少，对刊物聚类的贡献比较小，我对他们进行了去除处理。
- 数据加载和建立双类型异构网络（netWork.py）：这里我选择用集合数据结构来存储刊物和作者，对于它们之间的权值，选

用字典这个灵活的数据结构来存储，字典允许了字符串类型去索引权值，很方便，而且效率相对较高。代码如下：

```
def buildGraph(self):
        data = self.data
        dataFile = open(data, 'r')
        csvData = csv.reader(dataFile)
        authors = set()
        publication = set()
        author2pub = {}
        pub2author = {}
        author2author = {}

        for line in csvData:
            publication.add(line[0])
            for i in range(len(line)):
                if i != 0 and i != 1:
                    authors.add(line[i])

        dataFile.seek(0)



        for author in authors:
            author2pub[author] = {}
            author2author[author] = defaultdict(int)
        for pub in publication:
            pub2author[pub] = {}

        csvData = csv.reader(dataFile)
        count = 0
        for line in csvData:
            for i in range(len(line)-2):
```

```
                pub2author[line[0]][line[i+2]] =
pub2author[line[0]].setdefault(line[i+2], 0) + 1
                author2pub[line[i+2]][line[0]] =
author2pub[line[i+2]].setdefault(line[0], 0) + 1
                for j in range(i+1,len(line)-2):
                    author2author[line[i+2]][line[j+2]] += 1
                    author2author[line[j+2]][line[i+2]] += 1
        dataFile.close()
        return pub2author, author2author, author2pub, publication
```

这部分主要是对 csv 文件进行加载和处理即可，对权值进行计数，得到刊物-作者、作者-刊物，作者-作者字典，pub2author, author2pub, author2author，其中的值是在数据中出现的次数。

- 排名函数（rankFunc.py）:这里实现了论文中提到的 simplerank 和 authorityrank（权威排名），simplerank 计算公式如下：

$$\begin{cases} \vec{r}_X(x) = \dfrac{\sum_{j=1}^{n} W_{XY}(x,j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} W_{XY}(i,j)} \\ \vec{r}_Y(y) = \dfrac{\sum_{i=1}^{n} W_{XY}(i,y)}{\sum_{i=1}^{m}\sum_{j=1}^{n} W_{XY}(i,j)} \end{cases}$$

计算较简单，实现为函数 simpleRanking。

权威排名基于这样的假设：1、排名靠前的作者在排名靠前的刊物发表了许多的论文；2、排名靠前的刊物吸引许多排名靠前的作者的论文。

计算公式如下：

$$\vec{r}_X(i) = \sum_{j=1}^{n} W_{XY}(i,j)\vec{r}_Y(j). \qquad i) + (1-\alpha)\sum_{j=1}^{n} W_{YY}(i,j)\vec{r}_Y(j).$$

二者相互迭代计算，对于初值，我使用了 simplerank 的结果来作为权威排名的初始值，这里设置迭代次数限制为 10 次，α值赋为 0.95。代码如下：

```
def simpleRanking(pubFamily, pub2author, author2author, author2pub):
    rank_author = defaultdict(float)
    rank_pub = defaultdict(float)
    sum_W = float(0)

    for pub in pubFamily:
        rank_pub[pub] = float(0)
        for author in pub2author[pub]:
            tmp = float(pub2author[pub][author])
            rank_author[author] = rank_author.setdefault(author, float(0)) + tmp
            sum_W += tmp
            rank_pub[pub] += tmp

    for pub in pubFamily:
        rank_pub[pub] = rank_pub[pub] / sum_W

    for author in author2pub:
        rank_author[author] = rank_author[author] / sum_W
    return rank_pub, rank_author

def authorityRanking(author_confer,confer_author,author_author,clusterList,T=10,alpha=0.95):
```

```
    confer_score_in = defaultdict(float)
    author_score = defaultdict(float)
    confer_score = defaultdict(float)
    iniPercent = 1.0 / float(len(author_confer))
    for author in author_confer:
        author_score[author] = iniPercent
    for i in range(T):
        sumConferScore = 0.0
        for confer in clusterList:
            conferScore = 0.0
            for author in confer_author[confer]:
                conferScore += (
                    confer_author[confer][author]          *
author_score[author])
            confer_score_in[confer] = conferScore
            sumConferScore += conferScore
        for confer in confer_score_in:
            confer_score_in[confer] = confer_score_in[confer] /
float(
                sumConferScore)
        last_author_score = author_score.copy()
        for author in author_score:
            author_score[author] = 0.0
        for confer in clusterList:
            for author in confer_author[confer]:
                author_score[author] += (
                    confer_author[confer][author]          *
confer_score_in[confer] *
                    (alpha))
        for author in author_score:
            for co_author in author_author[author]:
                author_score[author] += (
```

```
                        last_author_score[co_author] *
                        (1 - alpha) *
author_author[author][co_author])
            sumAuthorScore = 0.0
            for author in author_score:
                sumAuthorScore += author_score[author]
            for author in author_score:
                author_score[author] /= float(sumAuthorScore)

        sumConferScore = 0
        for confer in confer_author:
            conferScore = 0
            for author in confer_author[confer]:
                conferScore += (
                    confer_author[confer][author] *
author_score[author])
            confer_score[confer] = conferScore
            sumConferScore += conferScore
        for confer in confer_score:
            confer_score[confer] = confer_score[confer] /
float(sumConferScore)

        return author_score, confer_score, confer_score_in
```

- RankClus 算法（rankClus.py）：以下部分均实现在 rankClus 类中
- 初始化聚类（initialGroup）：先对每一个聚类随机放入一个刊物，最后对剩余的刊物随机打上聚类标签。代码如下：

```
    def initialGroup(self, K = 15):
        initialGroup = {i : list() for i in range(K) }
        pub = list(self.publication)
        for i in range(K):
```

```
                initialGroup[i].append(pub.pop(random.randint(0,
len(pub) - 1)))
        pub = set(pub)
        for p in pub:
            initialGroup[random.randint(0,K - 1)].append(p)
        return initialGroup
```

- EM 得到聚类的后验分布（EM）：得到初始化聚类后便可以开

$$p(z=k)=\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}W_{XY}(i,j)p(z=k|x_i,y_j,\Theta^0)}{\sum_{i=1}^{m}\sum_{j=1}^{n}W_{XY}(i,j)}.$$

始迭代聚类的过程，这里的依据是认为如果一个作者是某个聚类的高质量成员，那么它在这个聚类下的排名分数应显著大于它在其他聚类下的排名分数，同理如果一个刊物是某个聚类的高质量成员，它在这个聚类下的排名分数也应显著大于在其他聚类下的排名分数。计算得到每个聚类下的作者和刊物排名分数，再使用 EM 估计方法去最大化似然函数，得到$p(k)$的更新函数，其中θ值是不断迭代的。

我们得到了聚类的先验分布，我们想得到对于每个刊物的聚类的后验分布，此时可用贝叶斯公式：

$$\pi_{i,k}=p(z=k|x_i)=\frac{p_k(x_i)p(z=k)}{\sum_{l=1}^{K}p_l(x_i)p(z=l)}$$

这里迭代次数设为 5 即可收敛。代码实现如下：

```
    def EM(self, rank_pub, rank_author, pubGroup, pub2author,
author2author, author2pub, emT = 5, K = 15):

        sum_cross = float(0)
        # initalize p_k
        p_k = np.zeros(K)
```

```
        sum_pub = 0.0
        for i in range(K):
            for pub in pubGroup[i]:
                for author in pub2author[pub]:
                    p_k[i] += pub2author[pub][author]
                    sum_pub += pub2author[pub][author]
        for i in range(K):
            p_k[i] /= float(sum_pub)

        new_p_k = np.zeros(K)
        while emT > 0:
            emT -= 1
            condition_p_k = {}
            for pub in pub2author:
                condition_p_k[pub] = {}
                for author in pub2author[pub]:
                    condition_p_k[pub][author] = {}

                    sum_cross += pub2author[pub][author]

                    sump = float(0)
                    for k in range(K):
                        tmp = rank_pub[k][pub] *
rank_author[k][author] * p_k[k]
                        sump += tmp
                        condition_p_k[pub][author][k] = tmp
                    for k in range(K):
                        condition_p_k[pub][author][k] /= sump

            for k in range(K):
                for pub in pub2author:
                    for author in pub2author[pub]:
                        new_p_k[k] +=
condition_p_k[pub][author][k] * pub2author[pub][author]

            new_p_k /= sum_cross
            p_k = new_p_k
```

```
            new_p_k = np.zeros(K)

        #使用 Bayes 规则计算每个刊物属于每个聚类的概率分布：
Pi(pub) = [p1, p2, ., pk]
        Pi = {}
        for pub in pub2author:
            normalization = float(0)
            Pi[pub] = np.zeros(K)
            for k in range(K):
                tmp = rank_pub[k][pub] * p_k[k]
                normalization += tmp
                Pi[pub][k] = tmp
            Pi[pub] /= normalization
        return Pi
```

- 调整聚类（adjustGroup）：至此已经得到了每个聚类对于刊物的后验分布，接下来对于每一个聚类将这个分布计算均值，得到每一个聚类的聚类中心，再分别计算出每个刊物距离最近的聚类中心，将这个刊物的标签打为该聚类，这里的相似度度量使用余弦相似度即可。
- 检查是否有聚类为空（isEmpty）：每次调整完聚类后需检查是否有聚类为空，如果有需要置零迭代次数重新开始。代码实现如下：

```
def isEmpty(self, group_list):
    result = False
    K=15
    for i in range(K):
        if len(group_list[i]) == 0:
            result = True
            break
return result
```

- 迭代进行这几个步骤（pipe），设置总的迭代次数限制为 25 次，聚类中心为 15，最终即得到 15 个聚类。代码实现如下：

```
    def pipe(self, iterNum = 25, K = 15, rankT = 10, alpha = 0.95,
emT = 5):
        group = self.initialGroup(K)
        rank_pub = {}
        rank_author = {}
        iters = 0
        while iters < iterNum:
            for i in range(K):
                rank_author[i], rank_pub[i],tmp=
authorityRanking(self.author2pub, self.pub2author,
self.author2author,group[i], rankT, alpha)
            Pi = self.EM(rank_pub, rank_author, group,
self.pub2author, self.author2author,self.author2pub, emT, K)
            new_group = self.adjustGroup(Pi, group, K)
            del group
            group = new_group
            if self.isEmpty(group):
                group = self.initialGroup(K)
                print('Empty group !')
                iters = 0
            else:
            iters += 1
```

一个商店实体往往还包括财务管理、人员管理等管理系统，实际运作时这些系统都会有一定的联系。

2.3 程序使用说明

该实现基于 Python3.6，依赖第三方库：numpy

首先需下载 DBLP 数据集，将 dblp.xml 和 dblp.dtd 放在工作目录 data 文件夹下。

运行 python dataParser.py

得到处理后的数据 data_solved.txt

运行 python rankClus.py

运行算法，屏幕会显示每个步骤的耗时，迭代完成后会输出聚类结果。

3 实验

3.1 实验环境

处理器：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
内存：4GB
硬盘：500GB SSD

3.2 性能测试

数据集选取了其中2008-2017年的作者-刊物数据，并剔除了发表刊物数少于10的作者。最终数据集包含13332个作者，5672个刊物，共239243篇论文数据。

每一轮循环大约需要耗时50秒左右，其中排名过程耗时16秒左右，EM算法大约耗时33秒左右，总体聚类过程25轮约耗时21分钟左右。

建立网络和调整聚类的时间非常短（1秒以内），未记录。

3.3 实验结果及分析

对于聚类的效果很难定量的去描述，我将某一次聚类的结果放在下面：

注：每一个 Group 是一个聚类，接着分为两部分输出，前一部分是在该聚类中排名前十的刊物名，后一部分是在该聚类中排名前十的作者名。

Group 0
CHI
CHI Extended Abstracts
UIST
CSCW
Conference on Designing Interactive Systems
UbiComp
Tangible and Embedded Interaction
Mobile HCI
ASSETS
ITS

* * * * * * *
Tovi Grossman
Patrick Olivier
Jacob O. Wobbrock
Carl Gutwin
George W. Fitzmaurice
Patrick Baudisch
Albrecht Schmidt 0001
Carman Neustaedter
Pattie Maes
Hiroshi Ishii 0001
- - - - - - - - - - - - - - - - -
Group 1
GCCE
ICTC
ICCE
IECON
GLOBECOM
ICC
ISIE
ICCE-Berlin
IGARSS
AIM
* * * * * * *
Tomio Goto
Satoshi Hirano
Miki Haseyama
Shingo Yamaguchi
Il-Woo Lee
Takahiro Ogawa
Masaru Sakurai
Jun Kyun Choi
Won Ryu
Yoshihiro Ito
- - - - - - - - - - - - - - - - -
Group 2
GECCO (Companion)

GECCO
CEC
FOGA
SSCI
EUROCAST (1)
SEAL
ECAL
ALIFE
Australasian Conference on Artificial Intelligence
* * * * * * *
Mengjie Zhang
Nikolaus Hansen
Anne Auger
Dimo Brockhoff
Lee Spector
Benjamin Doerr
Kalyanmoy Deb
Risto Miikkulainen
Peter A. N. Bosman
Carlos A. Coello Coello
- - - - - - - - - - - - - - - - -
Group 3
AMIA
CRI
MedInfo
Nursing Informatics
ICBO
IHI
BioNLP@ACL
ICBO/BioCreative
Pacific Symposium on Biocomputing
MIX-HS
* * * * * * *
David W. Bates
Hongfang Liu
Joshua C. Denny
Adam Wright

Suzanne Bakken
Charlene R. Weir
David K. Vawdrey
Chunhua Weng
Dean F. Sittig
Patricia C. Dykes
- - - - - - - - - - - - - - - - -
Group 4
SIGCSE
EDM
FLAIRS Conference
ITiCSE
ICER
LAK
FDG
AIED Workshops
L@S
AIED
* * * * * * *
Tiffany Barnes
Daniel D. Garcia
Neil T. Heffernan
Danielle S. McNamara
Kenneth R. Koedinger
Stephen H. Edwards
Ryan S. Baker
Arthur C. Graesser
Kristy Elizabeth Boyer
Mark Guzdial
- - - - - - - - - - - - - - - - -
Group 5
ICIS
AMCIS
ECIS
PACIS
Wirtschaftsinformatik
HICSS

MKWI
GI-Jahrestagung
Multikonferenz Wirtschaftsinformatik
Bled eConference
* * * * * * *
Helmut Krcmar
Jan Marco Leimeister
Alexander Benlian
Thomas Hess
Paul A. Pavlou
Dirk Neumann 0001
Sven Laumer
Kalle Lyytinen
Tim Weitzel
Rüdiger Zarnekow
- - - - - - - - - - - - - - - - -
Group 6
INTERSPEECH
ICASSP
SSW
Odyssey
SIGDIAL Conference
ICPhS
SLaTE
IWSLT
ICMI
ASRU
* * * * * * *
Shrikanth S. Narayanan
Haizhou Li
John H. L. Hansen
Hermann Ney
Björn W. Schuller
Bin Ma
Junichi Yamagishi
Mark J. F. Gales
Ralf Schlüter

Paavo Alku
- - - - - - - - - - - - - - - - -
Group 7
EMBC
BIOSIGNALS
ICORR
BioRob
ESANN
Wireless Health
RTSI
BIODEVICES
CBS
MeMeA
* * * * * * *
Nigel H. Lovell
Theodore W. Berger
Dimitrios I. Fotiadis
Hung T. Nguyen
Masakatsu G. Fujie
Socrates Dokos
Nitish V. Thakor
Gaetano Valenza
Dong Song
Helge B. D. Sørensen
- - - - - - - - - - - - - - - - -
Group 8
NIPS
ICML
AISTATS
COLT
UAI
ICML (3)
STOC
ICML (1)
ACML
SODA
* * * * * * *

Michael I. Jordan
Lawrence Carin
Pradeep Ravikumar
Zoubin Ghahramani
Inderjit S. Dhillon
Yoshua Bengio
Csaba Szepesvári
Francis R. Bach
Andreas Krause 0001
Eric P. Xing
- - - - - - - - - - - - - - - - -
Group 9
Medical Imaging: Computer-Aided Diagnosis
ACM Multimedia
Medical Imaging: Image-Guided Procedures
Medical Imaging: Image Processing
ICMR
ICIMCS
Medical Imaging: Digital Pathology
ISBI
MMSys
CIVR
* * * * * * *
Heang-Ping Chan
Lubomir M. Hadjiiski
Hiroshi Fujita 0001
Chuan Zhou
Jun Wei 0002
Ronald M. Summers
Berkman Sahiner
Takeshi Hara
Jianhua Yao
Anthony P. Reeves
- - - - - - - - - - - - - - - - -
Group 10
ACM Conference on Computer and Communications Security
SIGCOMM

USENIX Security Symposium
NDSS
NSDI
Internet Measurement Conference
MobiSys
CoNEXT
HotNets
MobiCom
* * * * * * *
Christopher Kruegel
Nick Feamster
Giovanni Vigna
Jennifer Rexford
Wenke Lee
Dina Katabi
Vyas Sekar
Vern Paxson
Ion Stoica
Ahmad-Reza Sadeghi
- - - - - - - - - - - - - - - - -
Group 11
SIGIR
CIKM
TREC
WWW
WSDM
KDD
WWW (Companion Volume)
SIGMOD Conference
ICWSM
CLEF (Working Notes)
* * * * * * *
Maarten de Rijke
W. Bruce Croft
ChengXiang Zhai
Iadh Ounis
Craig Macdonald

Leif Azzopardi
Ryen W. White
Gerhard Weikum
Oren Kurland
Pavel Serdyukov
- - - - - - - - - - - - - - - - -
Group 12
AAMAS
AAAI
CogSci
IJCAI
LREC
EMNLP
AAMAS (2)
ICAPS
ACL (1)
EC
* * * * * * *
Milind Tambe
Nicholas R. Jennings
Peter Stone
Sarit Kraus
Vincent Conitzer
Makoto Yokoo
Tuomas Sandholm
Michael Wooldridge
Alex Rogers
Edith Elkind
- - - - - - - - - - - - - - - - -
Group 13
ISMIR
ICMC
NIME
Semantic Audio
Audio Mostly Conference
DLfM@JCDL
iPRES

DAFx
DLfM@ISMIR
WASPAA
* * * * * * *
Gerhard Widmer
Meinard Müller
Xavier Serra
Simon Dixon
Ichiro Fujinaga
Masataka Goto
Kazuyoshi Yoshii
Markus Schedl
Mark B. Sandler
Sebastian Böck
- - - - - - - - - - - - - - - - -
Group 14
Description Logics
KR
OWLED
SEBD
International Semantic Web Conference (Posters & Demos)
AMW
OM
ORE
SWAT4LS
CILC
* * * * * * *
Rafael Peñaloza
Diego Calvanese
Carsten Lutz
Ian Horrocks
Michael Zakharyaschev
Frank Wolter
Bernardo Cuenca Grau
Roman Kontchakov
Bijan Parsia
Ulrike Sattler

通过比对 CCF 的计算机领域推荐论文，我们发现例如 Group11：

SIGIR
CIKM
TREC
WWW
WSDM
KDD
WWW (Companion Volume)
SIGMOD Conference
ICWSM
CLEF (Working Notes)
* * * * * * *
Maarten de Rijke
W. Bruce Croft
ChengXiang Zhai
Iadh Ounis
Craig Macdonald
Leif Azzopardi
Ryen W. White
Gerhard Weikum
Oren Kurland
Pavel Serdyukov

其中 SIGIR，CIKM，TREC，WWW，WSDM，SIGMOD 等等都是检索领域的知名度很高的会议，作者中 Maarten de Rijke 就是从事检索领域的大牛。

3.4 关于随机性的说明

该算法实现在初始化聚类的时候存在随机性，每次的结果可能稍有不同，但总体来看分类结果还是比较稳定的。

参考文献

[1] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, T. Wu, RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis, EDBT'09, 2004.

[2] 孙艺洲，韩家炜. 《异构信息网络挖掘：原理和方法》. 机械工业出版社，2017.