ธนกิจ เหล่ายาวิระ

600632014

Computer Assignment 2.2

1.รายละเอียดของทฤษฎีหรือวิธีการต่าง ๆ ที่ใช้

1. Levenshtein Distance คือ ขั้นตอนวิธีการนี้จะเป็นการนำชุดอักขระ 2 ชุด มาเปรียบเทียบจำนวนความแตกต่างกัน โดยจะพิจารณาดังนี้

- การแทรก เป็นการนำเอาอักขระตัวใด ๆ มา เพื่อให้ชุดอักขระชุดนั้นเหมือนกับอีกชุดอักขระหนึ่งในภายหลัง
- การตัดออก เป็นการตัดอักขระออกครั้งละ 1 ตัว จากชุดอักขระตัวหนึ่ง เพื่อให้ชุดอักขระชุดนั้นเหมือนกับอีกชุดอักขระหนึ่งในภายหลัง
- การแทนที่ เป็นการนำอักขระของชุดอักขระหนึ่งไปแทนอักขระของอีกชุดอักขระหนึ่ง เพื่อให้ชุดอักขระชุดนั้นเหมือนกับอีกชุดอักขระหนึ่งในภายหลัง

2. ใช้ NodeJS ในการพัฒนาซอฟท์แวร์

2. Algorithm

- N unlabeled finite strings $\mathbf{X}=\{\alpha_k | \ k=1,\ldots,N\}$
- Select number of clusters $1 \le C \le N$
  set maximum number of iteration (T)
  set termination measure → $\varepsilon_t = \|\mathbf{V}_t - \mathbf{V}_{t-1}\|$ ($\mathbf{V}$ is a set of string prototypes)
  termination threshold: ε
- Initial string prototype $\mathbf{V}_0 = (\alpha'_{1,0}, \ldots, \alpha'_{C,0})$ where $\alpha'_{i,0} \in \mathbf{X}$ and $i=1,\ldots,C$
- $t = 0$
- Repeat
  - $t = t+1$
  - For k=1:N
    - For i=1:C
      - $D_{ik,t} = \delta^2(\alpha_k, \alpha'_{i,t-1})$
    - End For
  - End For
  - For i=1:C
    - $n_{i,t}=0$
    - For k=1:N

$$u_{ik,t} = \begin{cases} 1 & \text{if } D_{ik,t} \le D_{jk,t} \text{ for } j \ne i \\ 0 & \text{else} \end{cases}$$

      - $n_{i,t}=n_{i,t}+u_{ik,t}$
    - End For
  - End For
  - For i=1:C

$$q = \arg\min_{1 \le j \le n_{i,t}} \left\{ c_{i,j} = \sum_{s=1}^{N} \left(\frac{\delta(u_{ij,t}\alpha_j, u_{ik,t}\alpha_s)}{n_{i,t}} \right) \right\}$$

    - $\alpha'_{i,t} = \alpha_q$
  - End For
- Until ($t=T$ or $\varepsilon_t \le \varepsilon$)

3. วิธีการทดลอง

1. รันโปรแกรมจากการสุ่ม prototype ใน train data
2. นำ prototype ที่เสถียรแล้วจาก train data ไปเป็น prototype เริ่มต้นของ test data
3. สังเกตผลการทดลอง

4. ผลการทดลอง

Train data	Test data
round: 1 current prototype : 2047,40,1627 (index) cluster 1 count: 435 cluster 2 count: 388 cluster 3 count: 1377 ========== round: 2 current prototype : 2023,225,1196 (index) et: 18.681541692269406 cluster 1 count: 724 cluster 2 count: 461 cluster 3 count: 1015 ========== round: 3 current prototype : 1850,245,1196 (index) et: 11.681541692269406 cluster 1 count: 815 cluster 2 count: 531 cluster 3 count: 854 ========== round: 4 current prototype : 1850,245,1168 (index) et: 1 cluster 1 count: 889 cluster 2 count: 489 cluster 3 count: 822 ========== round: 5 current prototype : 1850,245,1152 (index) et: 8 cluster 1 count: 918 cluster 2 count: 453 cluster 3 count: 829 ========== round: 6 current prototype : 1850,245,1152 (index) et: 0 cluster 1 count: 918 cluster 2 count: 453 cluster 3 count: 829 ==========	round: 1 current prototype : 1850,245,1152 (index) cluster 1 count: 921 cluster 2 count: 472 cluster 3 count: 807 ========== round: 2 current prototype : 1825,322,1050 (index) et: 0 cluster 1 count: 949 cluster 2 count: 433 cluster 3 count: 818 ==========

```
round: 1
current prototype : 268,972,125 (index)
cluster 1 count: 415
cluster 2 count: 1630
cluster 3 count: 155
================================

round: 2
current prototype : 350,1496,49 (index)
et: 16
cluster 1 count: 568
cluster 2 count: 1494
cluster 3 count: 138
================================

round: 3
current prototype : 546,1496,49 (index)
et: 4.248456731316587
cluster 1 count: 650
cluster 2 count: 1382
cluster 3 count: 168
================================

round: 4
current prototype : 1108,1578,49 (index)
et: 6.30337936338692
cluster 1 count: 918
cluster 2 count: 1083
cluster 3 count: 199
================================

round: 5
current prototype : 1152,1825,73 (index)
et: 5.3386326591070805
cluster 1 count: 1009
cluster 2 count: 923
cluster 3 count: 268
================================

round: 6
current prototype : 1152,1850,225 (index)
et: 21.213203435596427
cluster 1 count: 960
cluster 2 count: 891
cluster 3 count: 349
================================

round: 7
current prototype : 1152,1850,225 (index)
et: 0
cluster 1 count: 960
cluster 2 count: 891
cluster 3 count: 349
================================
```

```
round: 1
current prototype : 1152,1850,225 (index)
cluster 1 count: 918
cluster 2 count: 886
cluster 3 count: 396
================================

round: 2
current prototype : 1050,1847,322 (index)
et: 3
cluster 1 count: 897
cluster 2 count: 903
cluster 3 count: 400
================================

round: 3
current prototype : 1050,1825,322 (index)
et: 3
cluster 1 count: 884
cluster 2 count: 916
cluster 3 count: 400
================================

round: 4
current prototype : 1050,1825,322 (index)
et: 0
cluster 1 count: 884
cluster 2 count: 916
cluster 3 count: 400
================================
```

```
round: 1
current prototype : 2190,1458,1465 (index)
cluster 1 count: 420
cluster 2 count: 1701
cluster 3 count: 79
================================

round: 2
current prototype : 2023,1152,1465 (index)
et: 15.684387141358123
cluster 1 count: 631
cluster 2 count: 1253
cluster 3 count: 316
================================

round: 3
current prototype : 2001,1022,1634 (index)
et: 3.6843871413581226
cluster 1 count: 585
cluster 2 count: 1079
cluster 3 count: 536
================================

round: 4
current prototype : 2001,1108,1634 (index)
et: 12
cluster 1 count: 585
cluster 2 count: 1006
cluster 3 count: 609
================================

round: 5
current prototype : 2001,1108,1634 (index)
et: 0
cluster 1 count: 585
cluster 2 count: 1006
cluster 3 count: 609
================================
```

```
round: 1
current prototype : 2001,1108,1634 (index)
cluster 1 count: 630
cluster 2 count: 1110
cluster 3 count: 460
================================

round: 2
current prototype : 1882,1005,1615 (index)
et: 10
cluster 1 count: 581
cluster 2 count: 1005
cluster 3 count: 614
================================

round: 3
current prototype : 1882,1005,1359 (index)
et: 10
cluster 1 count: 671
cluster 2 count: 916
cluster 3 count: 613
================================

round: 4
current prototype : 1882,1005,1359 (index)
et: 0
cluster 1 count: 671
cluster 2 count: 916
cluster 3 count: 613
================================
```

4. สรุปผลการทดลอง

เมื่อใช้ prototype จาก train data มาเป็น prototype เริ่มต้นของ test data แล้ว จำนวนรอบของการจัดกลุ่มจะน้อยลง เพราะ prototype เริ่มต้นของ test มีความเสถียรแล้ว

ภาคผนวก

Code: String Grammar Hard C-Means (NodeJS)

```
let fs = require('fs')
const levenshtein = require('js-levenshtein');
const cNum = 3 //Change number of clusters here
const maxT = 100
let e = 0.005
let dataSet = 'da'

const fetchData = new Promise((resolve) => {
    let res = []
    for (let i = 1; i <= 22; i++) {
        let input = fs.readFileSync(`chrom/dif${i}${dataSet}`,
'utf8').trim().split('\r\n').map(x => x.split('\t'))
        res = res.concat(input)
    }
    resolve(res)
})

const main = async () => {
    let prototype = []
    let p1 = null
    let p2 = null
    let v1 = 0
    let v2 = 0
    let cluster = []
    let eCal = 99999
    let sourceData = await fetchData.then((values) => {
        return values.map((value) => value[1])
    })

    // random choose prototype
    if (prototype.length === 0) {
        for (let i = 0; i < cNum; i++) {
            let rand = Math.floor(Math.random() * sourceData.length)
            prototype.push(rand)
        }
    }
    for (let i = 0; i < cNum; i++) {
        cluster.push([])
    }
    p1 = prototype
    for (let t = 0; t < maxT; t++) {
        cluster = []
        for (let i = 0; i < cNum; i++) {
            cluster.push([])
        }
```

```
        for (let n = 0; n < sourceData.length; n++) {
            let minDist = 999999
            let chooseC = null
            for (let i = 0; i < cNum; i++) {
                let dist = await levenshtein(sourceData[n],
sourceData[prototype[i]])
                if (dist < minDist) {
                    chooseC = i
                    minDist = dist
                }
            }
            cluster[chooseC].push(n)
        }
        console.log(`round: ${t + 1}`)
        prototype = []
        let minDist = 999999
        let chooseV = null
        let sumDist = 0
        let cen = 0
        for (let i = 0; i < cNum; i++) {
            minDist = 999999
            chooseV = null
            for (let j = 0; j < cluster[i].length; j++) {
                sumDist = 0
                for (let k = 0; k < cluster[i].length; k++) {
                    if (cluster[i][j] != cluster[i][k]) {
                        let dist = await
levenshtein(sourceData[cluster[i][j]], sourceData[cluster[i][k]])
                        sumDist = sumDist + dist
                    }
                }
                cen = sumDist / cluster[i].length
                if (cen < minDist) {
                    chooseV = cluster[i][j]
                    minDist = cen
                }
            }
            prototype.push(chooseV)
        }

        if (t === 0) {
            p2 = prototype
        }
        else if (t === 1) {
            p1 = p2
            p2 = prototype
        }
        else {
```

```
            p1 = p2
            p2 = prototype
        }
        if (t > 0) {
            let sumPDist = 0
            for (let i = 0; i < cNum; i++) {
                let dist = await levenshtein(sourceData[p2[i]],
sourceData[p1[i]])
                sumPDist = sumPDist + Math.pow(dist, 2)
            }
            if (t === 0) {
                v1 = 0
                v2 = Math.sqrt(sumPDist)
            }

            else {
                v1 = v2
                v2 = Math.sqrt(sumPDist)
                eCal = Math.abs(v1 - v2)
            }
        }
        console.log(`current prototype : ${p1} (index)`)
        if (t != 0) console.log(`et: ${eCal}`)
        for (let c = 0; c < cNum; c++) {
            console.log(`cluster ${c + 1} count: ${cluster[c].length}`)
        }
        console.log(`================================= \n`)
        if (t > 0 && eCal <= e) break
    }
}

main()
```