به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس بازیابی هوشمند اطلاعات

پاسخ بخش تئوری تمرین 3

نام و نام خانودگی: سهیل حاجیان منش

شماره دانشجویی: 810100119

مهر ماه ۱۴۰۳

فهرست

اظهارنامه

تأیید می کنم که از LLM ها مطابق با دستورالعمل های بارگذاری شده در سامانه Elearn درس به طورمسئولانه استفاده کرده ام. تمام اجزای کار خود را درک میکنم و آماده بحث شفاهی درباره آنها هستم.

پاسخ سوال اول

پاسخ بخش اول

چون می خواهیم بر اساس doc_id مرتب سازی انجام شود، اگر از کلید (term, doc_id) به عنوان کلید میانی استفاده کنیم، MapReduce در مرحله Map بصورت خودکار کلیدهای میانی را قبل از ارسال به Reducer مرتب می کند. **یعنی ابتدا بر اساس کلمات مرتب سازی انجام شده، سپس بر اساس شماره اسناد.**

پس میتوانیم خط Sort را از Reducer حذف کنیم و دیگر نیازی به بافر کردن تمام posting های یک کلمه نیست.

پاسخ بخش دوم

```
1  class Mapper:
2      def Map(docid n , doc d):
3          H <= newAssociativeArray
4          for all term t ∈ doc d do
5              H{t}  <= H{t} + 1
6          for term t ∈ H do
7            Emit( (t, n) , H{t})
8  class Reducer:
9      def Reduce((term t, doc_id n) , frequency f):
10         Emit(term t, posting<n, f>)
11
```

پاسخ سوال دوم

پاسخ بخش اول

$$\vec{p} = (\alpha I + (1-\alpha)M)^T \vec{p}$$

$$\alpha = 0.15 \ , \ p\ init = 1/5$$

$$\alpha I = 0.15 * \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix} = \begin{bmatrix} 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix}$$

$$(1-\alpha)M = 0.85 * \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.85 & 0 & 0 & 0 \\ 0 & 0 & 0.85 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.425 & 0 & 0.425 \\ 0 & 0 & 0.85 & 0 & 0 \end{bmatrix}$$

$$A = (\alpha I + (1-\alpha)M) = \begin{bmatrix} 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix} + \begin{bmatrix} 0 & 0.85 & 0 & 0 & 0 \\ 0 & 0 & 0.85 & 0 & 0 \\ 0.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.425 & 0 & 0.425 \\ 0 & 0 & 0.85 & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.455 & 0.03 & 0.455 \\ 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.455 & 0.88 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.455 & 0.03 \end{bmatrix}$$

$$\vec{p}^1 = A^T * \vec{p} = \begin{bmatrix} 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.455 & 0.88 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.455 & 0.03 \end{bmatrix} * \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.455 \\ 0.03 \\ 0.115 \end{bmatrix}$$

$$\vec{p}^2 = A^T * \vec{p}^1 = \begin{bmatrix} 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.455 & 0.88 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.455 & 0.03 \end{bmatrix} * \begin{bmatrix} 0.2 \\ 0.2 \\ 0.455 \\ 0.03 \\ 0.115 \end{bmatrix} = \begin{bmatrix} 0.41675 \\ 0.2 \\ 0.3105 \\ 0.03 \\ 0.04275 \end{bmatrix}$$

پاسخ بخش دوم

$$\alpha I = 0 * \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix} = 0$$

$$(1-\alpha)M = 1 * \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A = (\alpha I + (1-\alpha)M) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

$$\vec{p}^1 = A^T * \vec{p} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \end{bmatrix} * \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.5 \\ 0 \\ 0.1 \end{bmatrix}$$

$$\vec{p}^2 = A^T * \vec{p}^1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \end{bmatrix} * \begin{bmatrix} 0.2 \\ 0.2 \\ 0.5 \\ 0 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \\ 0 \\ 0 \end{bmatrix}$$

پاسخ بخش سوم

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$a^0 = h^0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$h^1 = \begin{bmatrix} a^0[2] + a^0[3] \\ a^0[3] \\ a^0[1] \\ a^0[3] + a^0[5] \\ a^0[3] \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} => normalize \;\; h^1 = \begin{bmatrix} 2/7 \\ 1/7 \\ 1/7 \\ 2/7 \\ 1/7 \end{bmatrix}$$

$$a^1 = \begin{bmatrix} h^0[3] \\ h^0[1] \\ h^0[1] + \; h^0[2] + h^0[4] + h^0[5] \\ 0 \\ h^0[4] \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 4 \\ 0 \\ 1 \end{bmatrix} => normalize \;\; a^1 = \begin{bmatrix} 1/7 \\ 1/7 \\ 4/7 \\ 0 \\ 1/7 \end{bmatrix}$$

$$h^2 = \begin{bmatrix} a^1[2] + a^1[3] \\ a^1[3] \\ a^1[1] \\ a^1[3] + a^1[5] \\ a^1[3] \end{bmatrix} = \begin{bmatrix} 5/7 \\ 4/7 \\ 1/7 \\ 5/7 \\ 4/7 \end{bmatrix} => normalize \quad h^2 = \begin{bmatrix} 5/\sqrt{83} \\ 4/\sqrt{83} \\ 1/\sqrt{83} \\ 5/\sqrt{83} \\ 4/\sqrt{83} \end{bmatrix}$$

$$a^2 = \begin{bmatrix} h^1[3] \\ h^1[1] \\ h^1[1] + \; h^1[2] + h^1[4] + h^1[5] \\ 0 \\ h^1[4] \end{bmatrix} = \begin{bmatrix} 1/7 \\ 2/7 \\ 6/7 \\ 0 \\ 2/7 \end{bmatrix} => normalize \quad a^2 = \begin{bmatrix} 1/\sqrt{83} \\ 2/\sqrt{83} \\ 6/\sqrt{83} \\ 0 \\ 2/\sqrt{83} \end{bmatrix}$$

پاسخ بخش چهارم

در بخش ب که $\alpha = 0$ است، هیچ random jump ی بین صفحات نداریم و از از طریق لینکها بین صفحات حرکت می کنیم. و چون D هیچ لینک ورودیی ندارد و از طرفی E هم تنها یک لینک ورودی آن هم از D دارد، پس این دو از چرخه Crawl حذف می شوند و مقدارشان نیز پس از دو مرحله صفر می شود.

اما با داشتن damping factor در بخش الف، عملا در هر قسمت با احتمال 0.15 ممکن است به هر گره ای بپریم حتی اگر لینک خروجی به آن نداشته باشیم. پس باعث می شود مقادیر C,D هیچگاه صفر نشوند.

وجود Damping factor همچنین کمک می کند از خوردن به بن بست جلوگیری شود. در همین مثال یک حلقه و بن بست بین گره های A,B,C وجود دارد که وقتی وارد آن می شویم دیگر امکان خروج را نداریم که این باعث می شود به مرور بعد از همگرایی کامل گره های خارج از این بن بست مقدار خیلی کم و صفر داشته باشند و گره های این بن بست مقدارشان زیاد باشد. با damping factor>0 با احتمال α در هر مرحله به یک گره تصادفی می رویم و از بن بست خارج می شویم.

پاسخ بخش پنجم

نه مطابقت ندارند. بیشترین امتیاز Authority مربوط به C است ولی بالاترین rank در بخش الف مربوط به A می باشد.

در الگوریتم pagerank چون C گره مهمی و امتیازات زیادی از گره های دیگر میگیرد و از آن تنها به A می رویم، پس هر بار تمام امتیازات ان به گره A می رسد پس طبیعتا بیشترین امتیاز را خواهد داشت. اینطور می توان برداشت کرد که اگر یک صفحه ای پر بازدید و معتبر باشد و از آن به صفحه دیگری اشاره شده باشد، امتیاز صفحه دوم زیاد میشود.

اما در الگوریتم HITS چون چهارگره به C اشاره کرده اند که همگی بجز D امتیاز hob نسبتا خوبی دارند پس امتیاز authority گره C زیاد می شود. اما به گره A فقط C اشاره کرده است که گره C از لحاظ hob امتیاز یکسانی با گره های E , B دارد پس امتیاز authority گره A نمی تواند به C برسد.

سوال سوم

در شبه کد اول خروجی نهایی بعد از اجرای MapReduce برای هر واژه بصورت زیر است :

(term, (DocID, PRscore, position))

یعنی هر تکرار واژه، در هر سند همراه با PageRank آن سند و همراه با موقعیت آن واژه در سند بصورت یک رکورد جداگانه در خروجی می آید.

main.py

```
class Mapper
    def __init__(self, source_type, input_file):
        self.source = source_type          # "PageRank_File" or "Document_File"
        self.input_file = input_file       # file path

    def Map(self, line):

        doc_id, payload = parse(line)

        # case 1: PageRank file
        if self.source == "PageRank_File":
            pr_value = float(payload)
            Emit(doc_id, ("PR", pr_value))

        # case 2: Document text file
        elif self.source == "Document_File":
            position = 1
            tokens = tokenize(payload)

            for term in tokens:
                Emit(doc_id, ("TERM", term, position))
                position += 1

class Reducer
    def Reduce(doc_id, values):
        PR = 0
        term_positions = []

        for v in values:
            if v.tag == "PR":
                PR ← v.score
            else if v.tag == "TERM":
                Append(term_positions, (v.term, v.position))

        for (term, pos) in term_positions:
            Emit(key=term, value=(doc_id, PR, pos))
```

خروجی شبه کد دوم که در زیر آمده است، برای هر واژه لیستی از اسنادی که واژه در آن آمده است به همراه لیستی از مکان های تکرار واژه در آن سند به همراه امتیاز PageRank آن سند داریم. این لیستی بر اساس PageRank مرتب شده است.

فرمت نمونه خروجی :

term → [

(doc_id1, PR1, [pos1, pos2, ...]),

(doc_id2, PR2, [pos...]),

...

]

```
class Mapper2
    def Map(term, posting):
        Emit(term, posting)


class Reducer2
    def Reduce(term, postings):

        postings_dict # map DocID → (PR, list_of_positions)

        for (doc, pr, pos) in postings:
            if doc not in postings_dict:
                postings_dict[doc] ← (pr, [])
            Append(postings_dict[doc].positions, pos)

        P ← [(doc, pr, positions) for each doc in postings_dict] # convert to List

        SortDescending(P, key = pr) # sort based on pagerank

        Emit(term, P)
```