به نام خدا

داده کاوی

پروژه نهایی

عنوان پروژه:

پیاده سازی الگوریتم های PageRank و H&A با معماری MapReduce

استاد:

دکتر هراتی زاده

نویسنده:

خشایار فتحی نژاد

تابستان 99

شرح پیاده‌سازی:

فایل pre.py:

ابتدا بر اساس پیاده سازی قبلی ماتریس گوگل را برای همه حالات جست‌وجو میسازیم و ذخیره می‌کنیم.

برای الگوریتم H&A نیز ماتریس لینک ها را ذخیره می‌کنیم.

ذخیره سازی به دو صورت شکل می‌گیرد، یکی برای مقادیر اولیه بردار ها و یکی برای ماتریس ها.

ذخیره سازی ماتریس به فرم زیر می باشد:

‘Row-number Col-number Value’

ذخیره سازی بردار ها به فرم زیر می‌باشد:

‘Col-number Value’

خروجی این قسمت فایل‌های زیر می باشد:

برای pagerank:

A.txt – A_news.txt – A_sport.txt – A_fun.txt – R.txt

برای H&A:

Hvec.txt – L.txt

فایل های مربوط به pagerank:

فایل pagerank.py:

در پیاده سازی معماری MapReduce دو تابع داریم، Mapper و Reducer:

```
def mapper(self, _, line):
    a=line.split()
    if len(a)==2:
      for i in range(nodeno):
        yield (i,a)
    elif len(a)==3:
      yield (int(a[0]),a[1:3])
```

تابع Mapper فایل ها را خط به خط می‌خواند و با جدا کردن اجزای هر خط با تابع split طول لیست حاصل را بررسی می‌کند.

اگر لیست به طول 3 بود یعنی یکی از خطوط ماتریس بوده و اگر به طول 2 بود یعنی از بردارهای رنک می‌باشد.

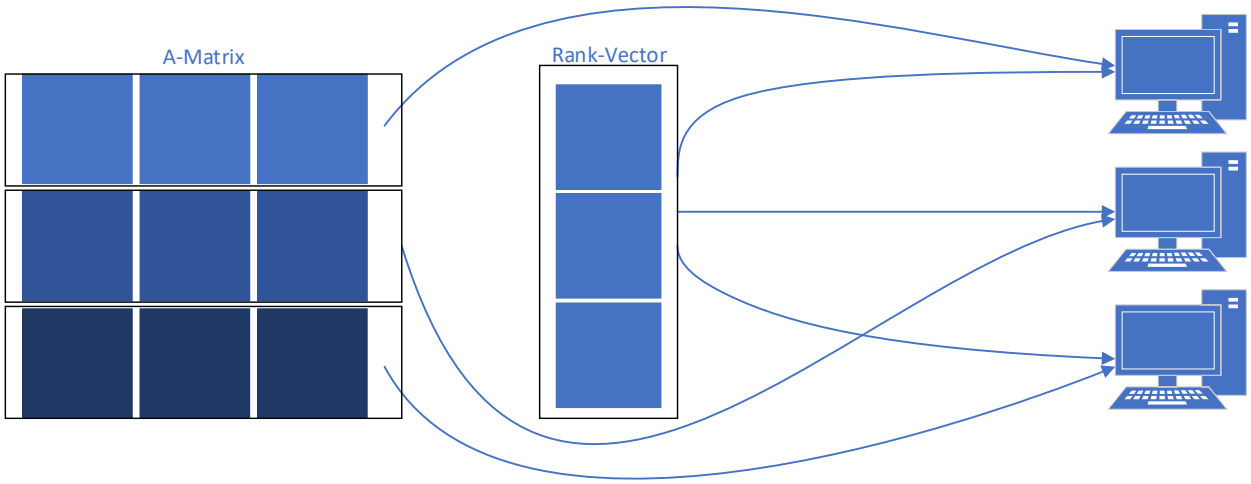
در صورتی که از اعضای ماتریس بود با کلید Row-number و مقادیر (Col-number و Value) برای سطح بعد ارسال می شود.

اگر بردار بود با تمام Row-number های ممکن به عنوان کلید، مقدار (Col-number و Value) برای سطح بعد ارسال می‌شود.

```
def reducer(self,key,values):
    lst=[]
    vals=[]
    for i in values:
      lst.append(int(i[0]))
      vals.append(float(i[1]))
    val=[x for _,x in sorted(zip(lst,vals))]
    value=0
    for i in range(1000):
      value+=val[2*i]*val[2*i+1]
    yield (key, value)
```

در reduce ها به ازای هر کلید که در حقیقت شماره راس می باشد، سطر مربوط به راس از ماتریس و بردار رنک را دریافت می کنیم.

پس در این تابع به ازای هر کلید دو برابر تعداد راس های گراف، و به ازای هر Col-number دو مقدار دریافت می‌شود. مقادیر را بر اساس col-number مرتب می‌کنیم و آنها را در هم ضرب می‌کنیم و نتایج را با هم جمع می‌کنیم. نتیجه مقدار رنک راس مربوطه در مرحله بعد است.



فایل pagerankmain.py:

تابع rank_generator با دریافت مقادیر نوع شخصی سازی و تعداد iteration بردار rank را با استفاده از فایل pagerank.py می سازد و با استفاده از تابع rank_vector_generator فایل های خروجی را تبدیل به فرمت تعریف شده برای بردار رنک (در ابتدای کار) تبدیل می کند.

تابع rank_generator دو فایل ورودی می‌خواند، یکی بردار اولیه رنک که برای همه ثابت است(R.txt) و دیگری بر اساس نوع شخصی سازی یکی از مقادیر زیر:

A.txt – A_news.txt – A_sport.txt – A_fun.txt

در هر iteration در فولدری به نام [R_{category}_{iteration-number}] ذخیره می‌شود.

در مرحله بعد از خروجی مرحله قبل استفاده می‌کنیم.

بعد از آخرین مرحله یک فایل به نام [R_{category}_{last-iteration-number}.txt] با استفاده از تابع rank_vector_generator ایجاد می شود که برای جست و جو استفاده می شود.

فایل ها و توابع مربوط به H&A:

فایل h.py:

کپی فایل pagerank.py می‌باشد.

فایل a.py:

همان منطق را دارد فقط مپ ها ماتریس را Transpose شده برای مرحله بعد ارسال می‌کنند.

فایل hmain.py:

در تکرار اول، بردار اولیه Hvec.txt و ماتریس لینک ها را می‌خواند و طبق الگوریتم از بردار H بردار A را می‌سازد و در فایلی به اسم [a_{iter-num}] ذخیره می‌کند. سپس با تابع nomalize() که در فایل normalize.py می‌باشد بردار A را می سازد، نرمال می کند و در فایلی به اسم [a_{iter-num}.txt] ذخیره می‌کند. با استفاده از این بردار H بعدی را محاسبه می‌کند.

در تکرار های بعدی از بردار A،H را می‌سازد و نرمال می‌کند و سپس از H،A را می‌سازد و نرمال می‌کند. با رسیدن به حداکثر تکرار متوقف می شود.

فایل main_search.py:

همانند پیاده سازی در تمرین چهارم می باشد. تابع search عبارت جست و جو شده را در سایت‌ها پیدا می‌کند و بین سایت هایی که شامل عبارت هستند، بر اساس دسته‌بندی انتخاب شده، امتیازشان در بردارهای ذخیره شده در مراحل قبل ترتیبی را ارائه می‌دهد.

شرح نتایج:

در پیاده‌سازی تحت هادوپ موفق نبودم و فقط تحت شبیه سازی MRJob برنامه اجرا می‌شود.

به علت محدودیت وقت و زمان زیاد اجرای برنامه تحت mapreduce به تعداد تکرار کافی برنامه را اجرا نکردم که بتوان با نتایج پیاده‌سازی قبل مقایسه کرد، اما چون ماتریس های اولیه و بردار ها دقیقا یکی هستند و پیاده سازی بدون خطا اجرا می‌شود قاعدتا باید بدون مشکل کار کند.

فایل های لازم جهت انجام جست و جو توسط تابع سرچ ارسال شده اما ماتریس ها و بردار های اولیه ارسال نشده و توسط فایلpre.py قابل تولید هستند.