بازیابی هوشمند اطلاعات

تمرین سوم

مینا فریدی

810100430

بخش یک

سوال ۱)

روش stripes:

در این روش تابع map به این صورت کار می کند که به ازای هر کلمه اطلاعات همجواری آن با کلمات دیگر را به صورت یک آرایه به نام H نشان می دهد. این تابع خروجی key value را به این صورت تولید می کند که قسمت key یک کلمه را نشان می دهد و در قسمت value آرایه مربوطه قرار دارد. تابع reduce جمع مقادیر موجود مربوط به هر عنصر را در تمام آرایه های با key یکسان جمع می کند و در نهایت خروجی key value با آن key و آرایه جمع شده به دست می آید. در واقع برخلاف روش pairs هر جفت key-value یک سطر را در ماتریس co-occurrence نشان می دهد. این روش تعداد جفت های key-value کمتری را نسبت به روش pairs تولید می کند و در کل سریعتر است اما پیاده سازی پیچیده تری دارد.

شکل زیر شبه کد مربوط به روش stripes را نمایش می دهد:

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term w ∈ doc d do
4:             H ← new AssociativeArray
5:             for all term u ∈ Neighbors(w) do
6:                 H{u} ← H{u} + 1                ▷ Tally words co-occurring with w
7:             Emit(Term w, Stripe H)

1: class Reducer
2:     method Reduce(term w, stripes [H_1, H_2, H_3, ...])
3:         H_f ← new AssociativeArray
4:         for all stripe H ∈ stripes [H_1, H_2, H_3, ...] do
5:             Sum(H_f, H)                        ▷ Element-wise sum
6:         Emit(term w, stripe H_f)
```

سوال ۲)

در روش pairs تابع map به این صورت کار می کند که ابتدا متن را خوانده و به ازای هر کلمه برای تمامی جفت کلمات قابل حصول از ترکیب کردن این کلمه با سایر کلمات متن یک جفت مقدار key و value ایجاد می‌کند که در این جفت مقدار key برابر با جفت لغت بوده و مقدار value برابر با یک می‌باشد. در این روش تابع reduce با جمع بستن مقادیر value هر جفت کلمه، فرکانس نسبی[1] دو کلمه در متن را محاسبه می کند. یکی از مزایای استفاده از روش pairs سهولت درک و همینطور پیاده سازی آن است. اما استفاده از این روش منجر به ایجاد جفت key value های زیادی می شود. که همین امر انجام پردازش بر روی این مقدارها را از نظر منابع حافظه‌ای و پردازشی می تواند دچار مشکل کند.

شبه کد زیر تابع map و reduce را در روش pair نشان می‌دهد.

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term w ∈ doc d do
4:             for all term u ∈ Neighbors(w) do
5:                 Emit(pair (w, u), count 1)        ▷ Emit count for each co-occurrence
1: class Reducer
2:     method Reduce(pair p, counts [c1, c2, ...])
3:         s ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             s ← s + c                              ▷ Sum co-occurrence counts
6:         Emit(pair p, count s)
```

[1] Relative Frequency

سوال ۳)

مقایسه دو روش pairs و stripes

یکی از مزایای استفاده از روش pairs سهولت درک و همینطور پیاده سازی آن است. اما استفاده از این روش منجر به ایجاد جفت key value های زیادی می شود که همین امر انجام پردازش بر روی این مقدارها را از نظر منابع حافظه‌ای و پردازشی می تواند دچار مشکل کند. روش stripes در کل بسیار فشرده تر است تعداد جفت های key-value کمتری را نسبت به روش pairs تولید می کند و در کل سریعتر است زیرا مرتب کردن جفت key-value های فراوان ایجادشده در روش pairs بسیار زمان می برد. اما در روش stripes مقادیر key-value پیچیده تر هستند.

در روش stripes باید اطمینان حاصل کرد که در هر زمانی آرایه ها به اندازه کافی کوچک هستند که در حافظه جای بگیرند که با افزایش اندازه متن این موضوع اهمیت پیدا می کند.

در کل روش stripes هم از نظر سرعت اجرا و هم از نظر حافظه مصرفی عملکرد مناسب تری نسبت به روش pairs دارد.

بخش دو

سوال ۱)

بن بست به گره هایی گفته می شود که یال خروجی نداشته باشند یا همه یال های خروجی آن ها بن بست باشند. گره های بن بست در گراف باعث می شوند که spider میزان رنک صفر به آن اسناد بدهد یا موجب همگرایی نتایج می شود.

برای پیدا کردن گره های بن بست به این طریق عمل می کنیم که ابتدا گره هایی که هیچ یال خروجی ندارند را پیدا می کنیم و گره بن بست نام گذاری می کنیم. سپس به صورت حلقه تا زمانی که گره بن بست پیدا نکنیم دنبال گره هایی می گردیم که تمام یال های خروجی آن ها به گره بن بست می رود. یعنی اگر تمام یال های خروجی یک گره به نودهای بن بست برود آن گره نیز بن بست نامیده می شود. در نهایت گره هایی که مانده اند گره بن بست نیستند.

در این سوال پس از خواندن فایل با استفاده از تابع read_edgelist فایل را به گراف تبدیل کرده و سپس گره های آن را با تابع نوشته شده می گردیم تا بن بست ها را پیدا کنیم. در نهایت خروجی را در فایل چاپ می کنیم.

سوال ۲)

الگوریتم pageRank رنکینگ گره های گراف را بر حسب ساختار یال های ورودی به دست می آورد.

برای انجام این سوال ابتدا باید گراف مورد نظر را از روی فایل بخوانیم و به ساختار گراف کتابخانه networkx تبدیل کنیم.

سپس گراف مورد نظر را به تابع pageRank کتابخانه networkx می دهیم تا عمل رنک کردن را انجام دهد.

در نهایت خروجی که از تایپ دیکشنری است را به شکل فایل csv در می آوریم.