باسمه تعالی

تاریخ: ۱۴۰۴/۱۰/۱۷

پروژه‌ی سوم درس مبانی علم داده

PageRank with Restarts on WikiVote: Dense vs Sparse and Deadend Handling

در این پروژه شما الگوریتم PageRank با restart را روی شبکه‌ی واقعی WikiVote پیاده‌سازی و تحلیل می‌کنید. ابتدا روی دو زیرگراف کوچک (برای اجرای dense و فهم رفتار الگوریتم) کار می‌کنید، سپس نسخه‌ی sparse را برای اجرای مقیاس‌پذیر روی گراف کامل پیاده‌سازی می‌کنید. بخش کلیدی پروژه، تشخیص و اصلاح مشکل deadend/dummy page و تفسیر نتایج در متن شبکه‌ی رأی‌دهی است.

قوانین پروژه (کتابخانه‌های مجاز/غیرمجاز)

مجاز: numpy, pandas, scipy, matplotlib, gzip, time, os, collections
غیرمجاز: networkx و هر تابع آماده‌ی PageRank/PPR (مثل networkx.pagerank یا هر پیاده‌سازی آماده مشابه)

معرفی دیتاست‌ها

دیتاست اصلی: WikiVote

یک گراف جهت‌دار از رأی‌دادن کاربران ویکی‌پدیا در فرآیند adminship. هر یال $A \to B$ یعنی: کاربر A به کاربر B رأی داده است.

زیرگراف A (نمونه‌گیری BFS/Frontier)

- **فایل‌ها:**
  - wikivote_subgraph_A_edges.tsv
  - wikivote_subgraph_A_mapping.tsv (نگاشت old_id ↔ new_id)

زیرگراف B (نمونه‌گیری Top-k induced)

- **فایل‌ها:**
  - wikivote_subgraph_B_edges.tsv
  - wikivote_subgraph_B_mapping.tsv

نکته: فایل‌های edges.tsv با شناسه‌های reindexed (از ۰ تا $n-1$) هستند تا ساخت ماتریس آسان باشد. فایل mapping برای ترجمه‌ی شناسه‌ها به ID اصلی WikiVote استفاده می‌شود.

تعاریف کلیدی

۱. **csr_matrix (Compressed Sparse Row)**
یک روش ذخیره‌سازی ماتریس تنک است که فقط عناصر غیرصفر را نگه می‌دارد. برای گراف‌ها یعنی: به‌جای نگه‌داشتن کل

ماتریس $n \times n$، فقط یال‌های موجود ذخیره می‌شوند.
مزیت کلیدی: ضرب ماتریس—بردار روی csr معمولاً حدوداً متناسب با nnz (تعداد عناصر غیرصفر (تقریباً برابر با تعداد یال‌ها)) انجام می‌شود، نه n^2.

۲. **adjacency matrix (ماتریس مجاورت)**
اگر یال $u \to v$ وجود داشته باشد، آنگاه $A[u, v] = 1$ درغیر این صورت 0.

۳. **in-degree / out-degree**
out-degree(u): تعداد یال‌های خروجی از u
in-degree(v): تعداد یال‌های ورودی به v

۴. **deadend / dangling**
نودی که $\text{out-degree} = 0$ دارد

۵. **row-normalization (نرمال‌سازی سطری)**
ساختن ماتریس انتقال P از روی A: هر سطر تقسیم بر out-degree همان نود (تا مجموع سطر ۱ شود؛ به‌جز danglingها).

۶. **restart probability / teleport (r)**
همان پارامتر کتاب (مثلاً 0.15)

۷. **residual (معیار توقف)**
$\texttt{residual}: \|p^{(t+1)} - p^{(t)}\|_1$
اگر residual از ε کوچک‌تر شد، می‌گوییم همگرا شده است.

۸. **Personalized PageRank (PPR)**
مثل PageRank است، فقط توزیع restart یکنواخت نیست؛ روی یک seed یا مجموعه‌ای از seedها متمرکز می‌شود.

گام ۰) آماده‌سازی و شناخت داده

کارهایی که باید انجام دهید

۱. دیتاست را بخوانید و تعداد node یکتا و تعداد edge را گزارش کنید.
۲. آمار پایه‌ی درجه‌ها را بدهید:
- out-degree و in-degree: میانگین، میانه، max، درصد صفرها
- تعداد node با out-degree = 0 (یعنی deadend/dangling)

۳. بررسی کنید آیا self-loop دارید یا نه. اگر دارید، تصمیم بگیرید حذف شود یا نه (و یک دلیل کوتاه بنویسید).

خروجی‌های این گام

- یک جدول خلاصه‌ی آمار
- Top-20 بر اساس out-degree و in-degree
- نمودار Rank-Degree(log-log): محور x رتبه‌ی گره‌ها بر اساس درجه (۱ = بیشترین درجه)، محور y مقدار درجه. دو منحنی جدا برای out-degree و in-degree.

سؤال‌های تحلیلی

۱. شبکه‌ی WikiVote بیشتر شبیه شبکه‌ی وب است یا شبکه‌ی ایمیل؟ چرا؟
۲. اگر تعداد زیادی out-degree صفر وجود داشته باشد، چه مشکلی برای تفسیر «random walk with restart» ایجاد می‌شود؟ (فقط پیش‌بینی کنید.)

پارامترها

r **(restart/teleport):** 0.15

ε: 10^{-9} (پیشنهادی)

`max_iters`: 500

گام ۱) PageRank نسخه‌ی Dense روی دو زیرگراف کوچک از WikiVote

داده‌های ورودی

- زیرگراف A (روش نمونه‌گیری BFS/Frontier)
- زیرگراف B (روش نمونه‌گیری Top-k induced)

کارهایی که باید انجام دهید (برای A و B جداگانه)

۱) بارگذاری و ساخت ماتریس مجاورت

- فایل edges.tsv را بخوانید.
- n را از فایل mapping یا $\max(\texttt{id}) + 1$ به دست آورید.
- یک ماتریس $A \in \mathbb{R}^{n \times n}$ بسازید که اگر یال $\text{u} \to \text{v}$ وجود داشت، آنگاه $A[u, v] = 1$ شود.

خروجی مورد انتظار:

- چاپ n و تعداد یال‌ها m
- چاپ تعداد self-loop

۲) ساخت ماتریس انتقال (Row-normalized)

- out-degree هر نود را حساب کنید.
- ماتریس انتقال را بسازید.
- نودهای با out-degree == 0 را فقط بشمارید (فعلاً درستش نکنید).

خروجی مورد انتظار:

- تعداد deadendها (out_degree == 0)

۳) اجرای PageRank — نسخه‌ی Basic (بدون رفع deadend)

- $r = 0.15$ و بردار شروع را یکنواخت بگیرید(یعنی به همه‌ی نودها مقدار برابر دهید).
- در هر iteration مقدار residual (نرم L_1) را حساب و ذخیره کنید.
- با شرط $\text{residual} < \varepsilon$ توقف کنید.

خروجی مورد انتظار:

- تعداد iteration تا توقف
- residual_last
- $\sum_i p_i$
- Top-10 نود (بر حسب new_id)
- نمودار residual بر حسب iteration با محور y لگاریتمی

۴) اجرای PageRank — نسخه‌ی اصلاح‌شده (رفع نشتی deadend)

- همان PageRank را اجرا کنید، اما طوری اصلاح کنید که در هر iteration مجموع احتمال‌ها باید تقریباً برابر ۱ بماند: $\sum_i p_i \approx 1$. (نشتی را جبران کنید).

خروجی مورد انتظار:

- تعداد iteration
- $\sum_i p_i \approx 1$
- Top-10 نودها و مقایسه با نسخه‌ی Basic (آیا تغییر کرد یا خیر؟)
- نمودار residual مثل قبل

سؤال‌های تحلیلی

۱. چرا در زیرگراف A در نسخه‌ی Basic مقدار $\sum_i p_i$ کمتر از ۱ شد؟ (ارتباط با deadend/dummy/restart)
۲. چرا در زیرگراف B نسخه‌ی Basic و اصلاح‌شده تقریباً یکسان شد؟ (به تعداد deadendهای B اشاره کنید.)
۳. اگر ε را از 10^{-9} به 10^{-8} تغییر دهید، Top-10 چقدر عوض می‌شود و چرا؟

گام ۲) PageRank نسخه‌ی Sparse و مقایسه‌ی زمان/حافظه

کارهایی که باید انجام دهید

ساخت ماتریس Sparse

برای هر زیرگراف (A و B) و سپس برای گراف کامل:

- از روی لیست یال‌ها یک ماتریس مجاورت csr_matrix بسازید.

نرمال‌سازی ردیفی

- ماتریس انتقال P را با نرمال‌سازی ردیفی (row-normalized) بسازید.
- تعداد نودهای با $\text{out-degree} = 0$ را گزارش کنید.
- **نکته:** در این گام فقط گزارش کنید؛ رفع مشکل dangling مربوط به گام ۳ است.

اجرای PageRank با ضرب Sparse (Power Iteration)

- PageRank را با تکرارهای متوالی (Power Iteration) اجرا کنید.
- شرط توقف پروژه را اعمال کنید

مقایسه‌ی زمان اجرا

برای زیرگراف‌های A و B زمان این بخش‌ها را جداگانه اندازه‌گیری کنید:

- زمان ساخت csr
- زمان نرمال‌سازی
- زمان اجرای iteration
- زمان کل

اجرای Sparse روی گراف کامل

همین نسخه‌ی Sparse را روی FullGraph اجرا کنید و گزارش دهید:

- تعداد iteration تا همگرایی
- زمان کل iteration (یا زمان متوسط هر iteration)
- $\sum_i p_i$ در پایان

برای زیرگراف‌های A و B، زمان اجرای PageRank را در دو حالت Dense و Sparse مقایسه کنید. (اختیاری)

خروجی‌های مورد انتظار

جدول زمان برای A و B با ستون‌های زیر:
graph, build_csr_s, normalize_s, iterate_s, total_s, iters, sum_p, nnz

گزارش عددی برای FullGraph:
n, nnz, iters, time_iterate_s, sum(p)

سؤال‌های تحلیلی

۱. با استفاده از عددهای واقعی دیتاست WikiVote (تعداد نودها و تعداد یال‌ها)، توضیح دهید چرا ساخت و نگه‌داری ماتریس Dense برای گراف کامل عملی نیست، اما نمایش Sparse (CSR) عملی است. ؟ جواب را هم با Big-O و هم با اعداد واقعی WikiVote توضیح دهید.

۲. اگر $\sum_i p_i$ در FullGraph کمتر از ۱ شد، این نشانه‌ی چیست و چه چیزی در گراف باعث آن می‌شود؟

گام ۳) رفع مشکل Personalized PageRank + deadend/dummy

A) رفع مشکل deadend (روی FullGraph)

ورودی: همان P_{full} و $P_{\text{full}}^{\mathsf{T}}$ که در گام ۲ ساخته‌اید. P ماتریس انتقال (نرمال‌سازی سطری) و P^{T} (ترانهاده‌ی آن).

کارهایی که باید انجام دهید

- دوباره PageRank را اجرا کنید ولی این بار رفع deadend را اضافه کنید.
- دو نسخه را اجرا کنید و مقایسه کنید:
  - q-fix:نشت را به همان بردار teleport (یعنی q) برگردانید.
  - u-fix: نشت را یکنواخت بین همه نودها پخش کنید.

خروجی‌های مورد انتظار

- جدول method، sum_p، iters، residual_last
- Top-20 برای هر روش
- میزان هم‌پوشانی Top-20 بین روش‌ها (overlap)

سؤال تحلیلی

- چرا با رفع deadend، $\sum_i p_i$ باید نزدیک ۱ شود؟

B) Personalized PageRank (کاربردی، روی FullGraph)

کارهایی که باید انجام دهید

- **Seed را انتخاب کنید:** Seed = old_id = 4037
- PPR را اجرا کنید (با restart روی seed).

خروجی‌های مورد انتظار

- $\sum_i p_i$
- Top-20

سوالات تحلیلی

۱. PPR در WikiVote چه نوع «اطرافیان مرتبط» را برجسته می‌کند؟
۲. چرا ممکن است بعضی نودها عملاً به seed ربطی نگیرند (حس disconnected)؟

سؤالات مفهومی

۱. در گراف WikiVote یال $A \to B$ یعنی «A به B رأی داده». پس PageRank بالا برای B دقیقاً چه تعبیر اجتماعی دارد؟
۲. «چطور ممکن است کسی رأی‌های کمتری گرفته باشد ولی PageRank بالاتری بگیرد؟»
۳. توضیح دهید چرا اگر دو کاربر به هم رأی بدهند (دوطرفه)، می‌تواند PageRank را بالا ببرد.
۴. چرا restart با $r = 0.15$ جلوی «بی‌نهایت بالا رفتن» را می‌گیرد؟
۵. اگر فقط اجازه داشته باشید **یک معیار** برای «مهم بودن» تعریف کنید، کدام را انتخاب می‌کنید و چرا؟ گزینه‌ها: in-degree، out-degree، PageRank، PPR.
پاسخ باید شامل ۲ مزیت و ۱ محدودیت باشد.