دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده ی ریاضی و علوم کامپیوتر

پایان‌نامه کارشناسی‌ارشد

گرایش علوم کامپیوتر

سرچ هیوریستیک

گزارش دوم

نگارش

پویاپارسا

استاد راهنما

دکتر قطعی

فروردین ۱۴۰۰


بسم الله الرحمن الرحيم

فصل اول

تعریف مسئله

۱-۱ صورت مسئله

فرض کنید در یک سیستم فرضی ده کاربر و پنج آیتم جمعا وجود دارند و کاربران به مرور زمان از بعضی از آیتم ها بازدید داشته اند. اگر کاربر i ، آیتم j را دیده باشد خانه ی ij در ماتریس تعامل یک و در غیر این صورت صفر است. هدف agent ما پیشنهاد آیتمی از میان موارد دیده نشده به یک کاربر خاص می باشد که از نظر کاربران مشابه این فرد ، بیش ترین محبوبیت را داراست.

Agent ما داده های کاربران را به صورت محیط (Enviroment) دارد و Performance measure نیز احتمال پیشنهاد ارائه شده از سمت agent است هر چند یک معیار واقع گرایانه تر این کانفیگ می تواند باشه که رضایت کاربر یا share کردن آن در نظر گرفته شود.

لذا ورودی مدل به این صورت است که لیستی از آیتم ها ،اطلاعات کل کاربران ، و اطلاعات مربوط به کاربر هدف قرار داده شده را دریافت می کند و در پاسخ به ما آیتمی را خروجی می دهد که بیشترین احتمال قبولی از سمت کاربر وجود دارد.

لذا مسئله به صورت Fully observable Single-agent می باشد.

حال برای مثال فرض کنید ده کاربر و پنج آیتم وجود دارد کاربر مورد نظر امتیاز دهی آن به صورت زیر است

[1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

یعنی آیتم های یک تا پنج را ملاقات کرده است و آیتم های ۶ تا ۱۰ را هنوز بازدید نکرده است ما باید سعی کنیم از میان آیتم های ۶ تا ۱۰ ، آیتم ای که به کاربر نزدیک تر است را پیدا کنیم.

۱-۲ راه کاری ساده

یکی از ساده ترین کار هایی که در وهله ی اول به ذهن می رسد اینست که برای هر آیتم دیده نشده ، میانگین امتیاز کل کاربران را قرار دهیم سپس از میان آن ها ماکزیمم گرفته و بیشترین عدد را به کاربر پیشنهاد دهیم.

هر چند این روش به شدت ساده است ولی دارای معایب قابل توجهی است: پیشنهاد را مستقل از شخصیت فرد می دهد یعنی برای این الگوریتم فرقی نمی کند که فرد مورد نظر خانم است یا آقا ، جوان است یا میانسال و ... که قطعا نرخ پذیرش را به شدت کاهش می دهد.

مشکل دوم : تجربه روی دیتا های واقعی به من این امر را اثبات کرده که معمولا نظرات حول یک آیتم چندان الگوی مشخصی ندارند مگر در موارد استثنا (کمتر از ۱ درصد موارد) و البته چندان هم عجیب نیست زیرا آیتمی از نظر دختر جوانی با وضعیت مالی مناسب قطعا امتیاز متفاوتی می گیرد نسبت به مرد کارمند میانسال با ۳ فرزند و میانگین این تفاوت ها را نادیده گرفته و معمولا عددی حدود وسط بازه را به ما می دهد.

فصل دوم

مدل سازی

۲-۱ رویکرد

برای حل مشکلات مطرح شده در فصل اول ، می توان دو رویکرد کلی را پیش گرفت [۲] :
اول : افراد بر حسب تعامل آنها با آیتم ها هم مقایسه و افرادی که به هم شبیه هستند را پیشنهادات مشابه دهیم
دوم : آیتم ها را با هم مقایسه کنیم و آیتم های مشابه را پیدا کنیم و به افراد پیشنهاد دهیم
من سعی در ارائه ی راه حل از طریق رویکرد اول را دارم
پس ابتدا باید افراد با رفتار مشابه به این فرد را پیدا کنیم و سپس از میانگین استفاده کنیم می توان هر کاربر با بردار امتیازهای آن نمایش داد و سعی به پیدا کردن کاربران نزدیک به کاربر مورد نظر کرد.

۲-۲ تعریف ریاضی مسئله

سعی در تشکیل گرافی از کاربران داریم به طوری که:

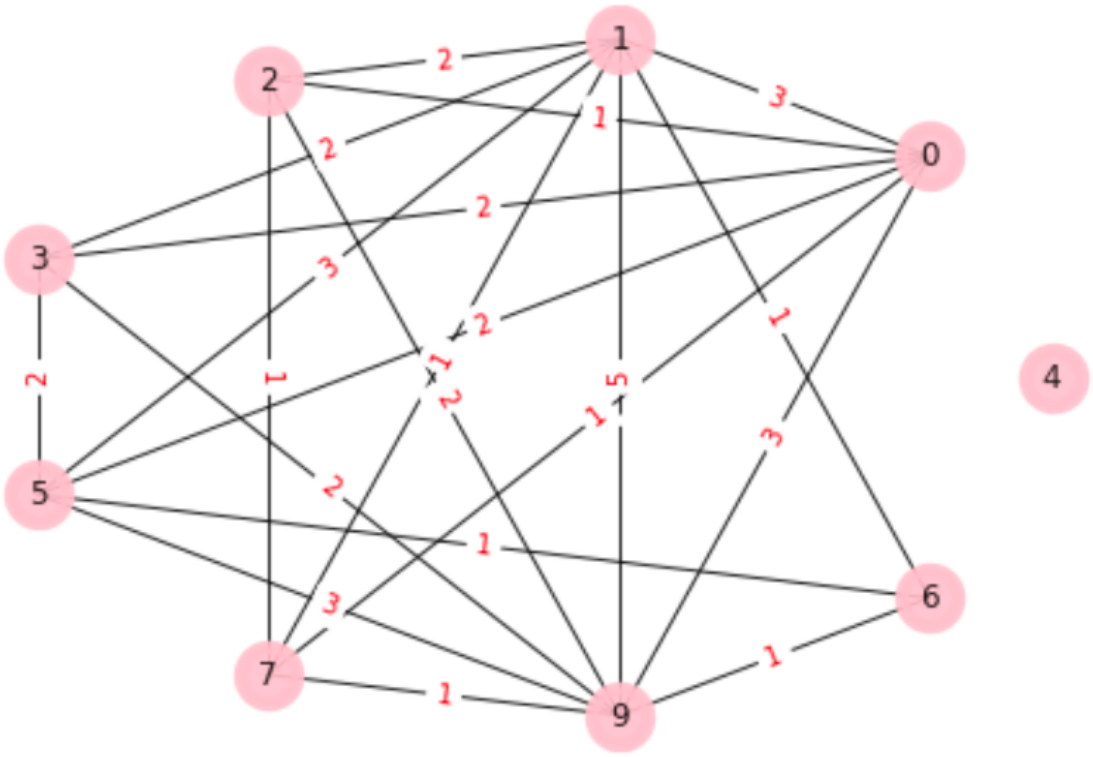
۱. دو کاربر i و j به هم یال دارند اگر این دو کاربر حداقل یک آیتم ملاقات کرده ی مشترک داشته باشند و در غیر این صورت یالی بین این دو وجود ندارد.

۲. وزن هر یال متناسب با تعداد آیتم های ملاقات شده ی مشترک است یعنی اگر کاربرi و j ، n آیتم مشترک ملاقات شده داشته باشند وزن یال بین آن ها n است.

هدف از جستجو در این گراف اینست که : با مشخص بودن یک راس (کاربر هدف) تعداد k کاربر را پیدا کنیم به طوری که به کاربر هدف نزدیک ترین شباهت رفتاری را داشته باشند.

۲-۳ چالش های مسئله

در پیمایش گراف باید توجه کنیم که ممکن است دور وجود داشته باشد زیرا فرد ۱ با فرد ۲ آیتم مشترک داشته باشد و فرد ۱و ۳ آیتم مشترک داشته باشند و فرد ۲ و۳ هم همینطور.
گراف ممکن است هم بند نباشد یا به بیان دقیق تر از کاربر هدف به کمتر از k راس دسترسی داشته باشیم.
برای حل مشکل اول : ما هر راسی را که ملاقات می کنیم پرچمی را برای آن تغییر می دهیم که باعث می شود دوباره به آن راس برنگردیم.
برای حل مشکل دوم :به این صورت عمل می کنیم که اگر راسی فرزندی نداشت ولی هنوز k کاربر مشابه را پیدا نکرده باشیم در این صورت به تعداد کاربر های پیدا شده بسنده می کنیم.



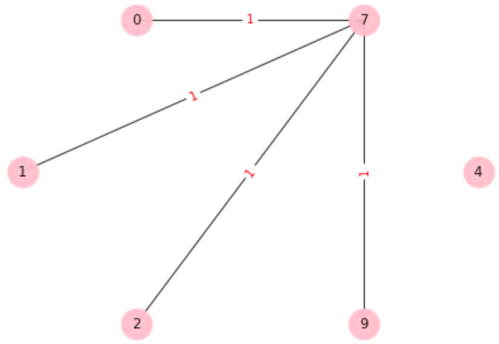
شکل ۲-۱: گراف متناظر با ماتریس تعامل و قوانین مطرح شده.

فصل سوم

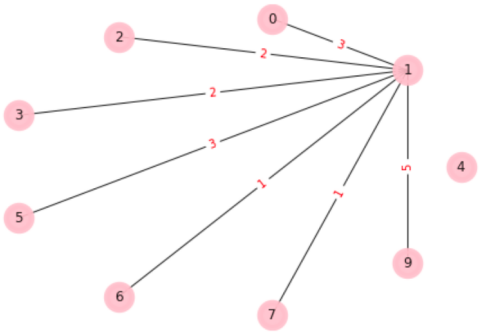
الگوریتم جستجو

۳-۱ پیدا کردن کاربران مشابه

از لحاظ هیوریستیک می توان ادعا نمی نمود که هر چه کاربری اختلاف عمق کمتری نسبت به کاربر هدف داشته باشد به آن کاربر شبیه تر است برای واضح تر شدن این امر به مثال زیر توجه کنید فرض کنید کاربر شماره ی ۷ را در نظر گرفته ایم ، ارتباطات این کاربر با سایر کاربران به شکل زیر است :



شکل ۳-۱: ارتباطات کاربر ۷.



شکل ۳-۲: ارتباطات کاربر ۱.

همانطور که در شکل ۳-۱ دیده می شود کاربر شماره ی ۷ با کاربران ۰ و ۱ و ۲ و ۹ به طور آیتم ملاقات شده ی مشترک دارد اما کاربر شماره ۶ به طور مستقیم آیتم مشترکی با کاربر مورد هدف یعنی ۹ ندارد ولی با کاربر ۱ آیتم مشترک دارد لذا به طور هیوریستیک می توان گفت هر چه کاربر در لایه ی نزدیک تری به کاربر هدف باشه به اون از لحاظ رفتاری شبیه تر است.

لذا با استفاده از همین شهود بر روی کاربر مورد نظر شروع به جستجو کرده . فرزندان را به ترتیب اشتراکات ملاقات می کنیم یعنی هر چی آیتم مشترکی بیشتر با نود والد خود داشته باشد زودتر ملاقات می شود . جستجو را تا زمانی ادامه می دهیم که تعداد کاربران مشابه مورد نظر (k) را پیدا کنیم و یا کاربر مشابه ای وجود نداشته باشد.

۳-۲ پیاده سازی

برای پیمایش سطح به سطح گراف از الگوریتم معروف و کارای BFS استفاده شده است با این شروط اضافی که هر گاه تعداد کاربران مشابه از k بیشتر شد پیمایش متوقف خواهد شد کد مربوط به این الگوریتم در تابع find similar نوشته شده است.

کد استفاده شده در این گزارش از طریق این لینک در دسترس است .

```python
# look for children with highest similarity and then
# the children of the most similar child
# aiming to found k most similar users


def find_similar(G, node, k):

    similar_nodes = []

    # Mark all the vertices as not visited
    visited = [False] * (nx.number_of_nodes(G) + 1)
    # Create a queue for BFS
    queue = []

    # Mark the source node as
    # visited and enqueue it
    queue.append(node)
    visited[node] = True

    while queue:
        s = queue.pop(0)

        for i in neighbours(s):
            if len(similar_nodes) >= k :
                return similar_nodes

            if visited[i] != True:
                queue.append(i)
                similar_nodes.append(i)
                visited[i] = True

    return similar_nodes

```

شکل ۳-۳: پیاده سازی جستجو در گراف با استفاده از [۱]

فصل چهارم

تحلیل حساسیت

۴-۱ مقدمه

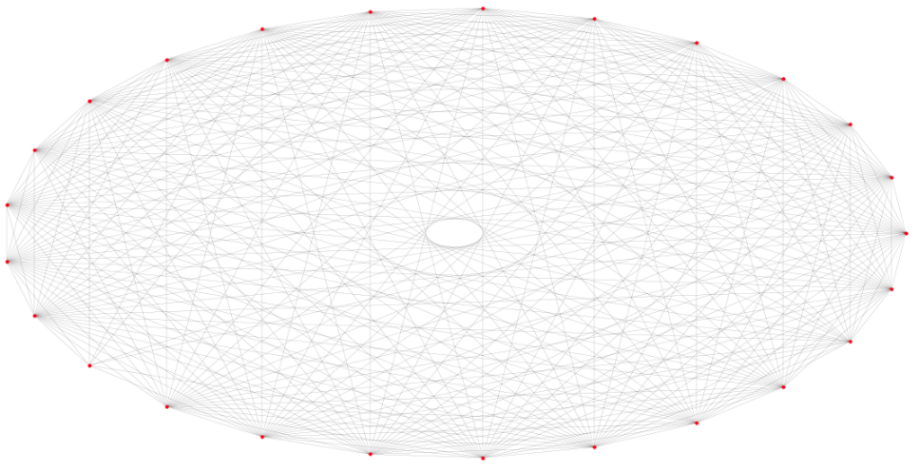
ورودی این Agent گراف تعامل کاربران است که طی یک پیش پردازش ساخته می شود . لازم به ذکر است که ساخت این گراف ، به تعداد کاربران و تعداد آیتم ها بستگی دارد و می توان گفت مرتبه ی زمانی ساخت این گراف از $O(n^۲)$ است

در جدول زیر می توان زمان اجرای این الگوریتم را به ازای سایز ورودی متفاوت و کاربران مشابه متفاوت مشاهده کرد:

تعداد کاربر مشابه	تعداد کل کاربران	تعداد آیتم ها	زمان(ثانیه)
۵	۱۰۰۰	۱۰۰۰	۴۱۴.۶۵
۵	۱۰۰	۱۰۰	۰.۵۷
۵	۱۰۰۰	۱۰۰	۴۸.۱۷
۱۰	۱۰۰۰	۱۰۰	۴۶.۳۷
۱۰	۱۰۰	۱۰۰	۰.۶۱
۱۰	۱۰۰۰	۱۰۰۰	۴۱۵.۴۵

جدول ۴-۱: زمان اجرا با ساخت گراف

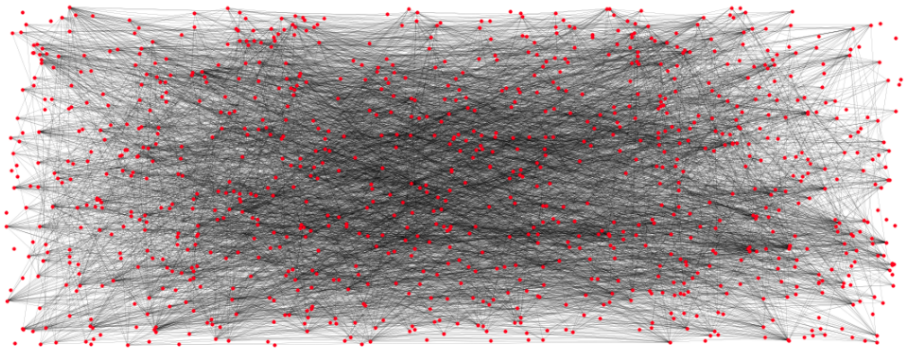
در آزمایش های بالا فرض بر آن بود که هر کاربر نیمی از آیتم ها را به صورت تصادفی مشاهده کرده است این امر موجب متقارن شدن ماتریس و بالا بودن درجه هر راس می شود. به نمونه گرافی این چنینی توجه کنید :



شکل ۴-۱: گراف تعامل با نرخ بازدید ۵۰ درصدی .

۴-۲ تاثیرات تنک بودن ماتریس تعامل

در طی آزمایش های بالا ، فرض بر این انگاشته شده بود که هر کاربر نیمی از آیتم ها را به طور تصادفی ملاقات کرده است این امر سبب بالا رفتن درجه ی هر راس می شود و در نتیجه در عمق کمتری کاربران مشابه بیشتری پیدا می شود حالا شاید این سوال به وجود بیاید که در دنیای واقعی معمولا تعداد کاربران بسیار بیشتر از تعداد آیتم هاست و ماتریس تعامل بسیار تنک است در این شرایط گراف چگونه خواهد بود و زمان اجرا آن چه تفاوتی خواهد کرد .
در جدول زیر زمان اجرای برنامه را در صورتی که هر کاربر تنها پنج درصد از آیتم ها را به طور رندوم تماشا کرده باشید مشاهده می کنید:



شکل ۴-۲: گراف تعامل با نرخ بازدید پنج درصد

تعداد کاربر مشابه	تعداد کل کاربران	تعداد آیتم ها	زمان(ثانیه)
۵	۱۰۰۰	۱۰۰۰	۳۵۳.۵۲
۵	۱۰۰	۱۰۰	۰.۶۴
۵	۱۰۰۰	۱۰۰	۳۸.۶۶
۱۰	۱۰۰۰	۱۰۰	۴۰.۲۵
۱۰	۱۰۰	۱۰۰	۰.۷۶
۱۰	۱۰۰۰	۱۰۰۰	۳۴۶.۱۳

جدول ۴-۲: زمان اجرا با ساخت گراف حالت تنک

همان طور که در دو جدول اخیر نشان داده شد. به نظر می رسد زمان اجرا با تعداد کاربران و تعداد آیتم ها رابطه ی خطی دارد و همچنین تعداد کاربران مشابه چندان تفاوتی بر زمان اجرا در مقایسه با سایر عوامل نمی گذارد.

منابع و مراجع

[1] Geeksforgeeks. Breadth first search or bfs for a graph. https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/.

[2] Kordík, Pavel. Machine learning for recommender systems — part 1 (algorithms, evaluation and cold start)). https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-sta