خزش وب با استفاده از روش‌های یادگیری تقویتی

آزاده سالاری [۱]، دکتر ولی درهمی[۲]، دکتر جواد پاکسیما[۳]

[۱]دانشگاه علم و هنر یزد، گروه کامپیوتر، azade.salari@gmail.com

[۲]دانشگاه یزد، پردیس فنی و مهندسی، گروه کامپیوتر، vderhami@yazd.ac.ir

[۳]دانشگاه پیام نور یزد، گروه کامپیوتر، paksima@ymail.com

چکیده

خزشگر یکی از اصلی‌ترین بخش‌های یک موتور جستجو می‌باشد که وظیفه‌ی آن کشف و دانلود صفحات وب است. هیچ موتور جستجویی نمی‌تواند کل وب را پوشش دهد و تنها به درصدی از صفحات با ارزش بالاتر اکتفا می‌کند؛ بنابراین چالش اصلی در موتورهای جستجو خزش صفحات مؤثر وب است به‌طوری که در سریع‌ترین زمان صفحات مهم را خزش نمایند. در این مقاله جهت پوشش مناسب صفحات وب (پیدا کردن سریع صفحات مهم)، الگوریتم خزشی مبتنی بر یادگیری تقویتی ارائه می‌گردد. سیگنال تقویتی براساس تابعی از درجه خروجی هر صفحه تعریف می‌شود و ارزش هر صفحه برابر مجموع تخفیف یافته‌ی جوایز دریافتی در گذر از صفحات وب تا رسیدن به صفحه‌ی جاری است. جهت ارزیابی الگوریتم پیشنهادی از گراف وب ایران استفاده شده است. نتایج آزمایش‌ها نشان می‌دهد که روش پیشنهادی روی گراف مذکور نسبت به سایر روش‌های بررسی شده کاراتر است..

واژه‌های کلیدی

موتور جستجو، خزشگر، رتبه‌بندی، گراف وب، یادگیری تقویتی.

۱- مقدمه

امروزه وب جهان گستر به عنوان بهترین محیط برای تولید اطلاعات، انتشار و دسترسی به دانش مورد نیاز کاربران تبدیل شده است، اما محیط وب از پویایی بالایی برخوردار بوده و از لحاظ ساختار و محتوا در حال رشد می‌باشد. هر روز حدود ۸٪ صفحه جدید ساخته می‌شود و در یک سال ۸۰٪ صفحات ناپدید می‌شوند، در هر هفته ۲۵٪ پیوند جدید ساخته می‌شود و در یک سال ۸۰٪ پیوندها ناپدید می‌شوند [۲]. یکی از اصلی‌ترین بخش‌های یک موتور جستجو خزشگر[۱] می‌باشد. خزشگر برنامه‌ای است برای دانلود کردن بخش عمده‌ای از صفحات وب [۳]، به این صورت که مجموعه‌ای از صفحات وب را به عنوان ورودی دریافت کرده (صفحات ریشه) و لینک‌های خروجی صفحات را به‌عنوان خروجی استخراج می‌کند و بر اساس یک معیار خاص لینکی که در مرحله‌ی بعد باید ملاقات شود را مشخص می‌کند. اکثر خزشگرها به سه دلیل پر هزینه بودن پهنای باند [۴] ، محدود بودن فضای ذخیره‌سازی [۵] و حفظ تازگی صفحات نمایه شده نمی‌توانند تمام صفحات موجود در وب را بار گذاری کنند.

این آمارها حاکی از حیاتی بودن دقت موتورهای جستجو در پیدا کردن صفحات مهم در کوتاه‌ترین زمان ممکن است. الگوریتم‌های خزش معمولاً از یک مکانیزم رتبه‌دهی در حین خزش جهت تعیین اولویت صفحات برای بارگذاری استفاده می‌کنند [۱]. به عبارتی دیگر یک مکانیزم رتبه‌دهی بر گراف ناقصی که در حین فرآیند خزش ایجاد شده است اعمال می‌شود و براساس آن اولویت صفحات مشخص می‌گردد. در این مقاله به منظور پیاده‌سازی روش پیشنهادی از روش رتبه‌بندی RL-Rank[۲] استفاده می‌شود. روش پیشنهادی RL-Crawler[۳] نامیده شد.

در بخش بعد کارهای مرتبط گذشته مرور می‌شود. بخش ۳ به معرفی روش پیشنهادی می‌پردازد و نهایتاً در بخش ۴ نتیجه‌گیری و کارهای آینده گفته خواهد شد.

۲- پیشینه و کارهای مرتبط

به دلیل حجم زیاد و پویایی اطلاعات در وب کشف صفحات مهم از اهمیت ویژه‌ای برخوردار است. در بخش خزش موتور جستجو مسئله‌ی مهم ارزش‌گذاری صفحات است تا براساس آن صفحات برای بارگذاری انتخاب

[1] Crawler

[2] Reinforcement Learning Rank

[3] Reinforcement Learning Crawler

شوند. در حوزه بازیابی اطلاعات وب روش‌های زیادی جهت خزش ارائه شده‌اند که در ادامه تعدادی از مهم‌ترین و کاراترین آن‌ها مورد بحث قرار می‌گیرد.

در [۴] از الگوریتم عرض اول به‌عنوان یک الگوریتم خزش استفاده شده است. در این الگوریتم پس از تعیین صفحه‌ی هسته کلیه‌ی صفحات هم عمق با یکدیگر مشخص می‌شوند و پس از رجوع به کلیه‌ی صفحات موجود در آن سطح، سطح دوم مورد بررسی قرار می‌گیرد. درواقع خزشگر در الگوریتم عرض اول از ریشه شروع و گراف را به‌صورت درختی (سطح به سطح) پیمایش می‌کند. به‌طوری‌که فاصله صفحات بارگذاری شده از ریشه همیشه کمتر یا مساوی صفحات بارگذاری نشده می‌باشد.

الگوریتم PageRank روش رتبه‌بندی مشهوری است که موتور جستجوی گوگل برای تعیین اهمیت صفحات وب از آن استفاده می‌کند. در این الگوریتم هر لینک بر اساس اهمیت سندی که از آن منتشر شده و تعداد لینک‌های خروجی آن سند وزن دهی می‌شود. ایده‌ی الگوریتم PageRank بر مبنای قدم زدن تصادفی است که مدل موج‌سوار تصادفی نامیده می‌شود [۶]. در مدل موج‌سوار تصادفی کاربران بر روی لینک‌های صفحات ملاقات شده به‌طور تصادفی کلیک می‌کنند. در این روش اگر خزشگر به صفحه‌ای برود که لینک خروجی نداشته باشد به‌صورت تصادفی به صفحه‌ی دیگری پرش می‌کند در واقع فرض می‌شود که کاربر یا لینک صفحه‌ی جاری را دنبال می‌کند یا به یک صفحه‌ی تصادفی در گراف وب پرش می‌کند. رتبه صفحه p به صورت زیر محاسبه می‌شود [۴]:

$$R(p) = \frac{1-d}{n} + d * \sum_{j \in B(p)} \frac{R(j)}{O(j)} \quad (۱)$$

در این رابطه n تعداد کل صفحات وب را نشان می‌دهد. O(j) بیانگر تعداد لینک‌های خروجی صفحه j و B(p) مجموعه صفحاتی است که به p اشاره می‌کنند. از پارامتر d، ضریب استهلاک، به منظور تضمین همگرایی الگوریتم PageRank و از بین بردن تأثیر صفحات بدون ورودی و خروجی و یا اصطلاحاً صفحات چاهک استفاده می‌شود.

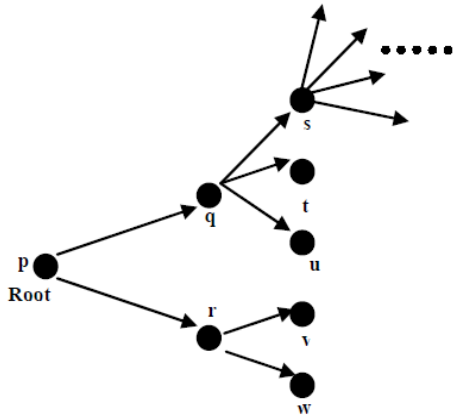
در [۷] مقایسه‌ای بین الگوریتم‌های PageRank، درجه ورودی و عرض-اول انجام شده است. آن‌ها به این نتیجه رسیدند که الگوریتم خزش مبتنی بر PageRank صفحات مهم‌تر را سریع‌تر از دو الگوریتم دیگر پیدا می‌کند.

الگوریتم OPIC[۴] جز الگوریتم‌های خزش برخط می‌باشد. نحوه‌ی کار این الگوریتم به این صورت است که در شروع خزش کلیه‌ی صفحات دارای اعتبار یکسان‌اند و با انتخاب یک صفحه جهت بارگذاری اعتبار آن بین فرزندانش به‌صورت مساوی تقسیم می‌گردد [۸]. مزیتی که به روش PageRank دارد سرعت بالای آن در اجرا است.

زارع بیدکی و یزدانی [۹] یک الگوریتم خزش هوشمند مبتنی بر یادگیری تقویتی[۵] با عنوان FICA[۶] معرفی کرده‌اند. الگوریتم FICA همانند روش عرض-اول با تعریف جدید فاصله عمل می‌کند.

تعریف۱: اگر صفحه‌ی i به صفحه‌ی j اشاره کند وزن پیوند میان i و j برابر است با logO(i) که O(i) نشان‌دهنده‌ی درجه خروجی صفحه‌ی i می‌باشد.
تعریف۲: فاصله‌ی لگاریتمی میان i و j عبارت است از وزن کوتاه‌ترین مسیر میان آن‌ها (جمع وزن‌های پیوندهای در مسیر) که با d_{ij} نشان داده می‌شود.

در شکل ۱ به ترتیب وزن‌های پیوندهای خروجی در صفحات p، q و s مساوی log2، log3 و log4 می‌باشد. فاصله‌ی میان دو صفحه‌ی p و t برابر log2+log3 و همچنین فاصله‌ی میان v و p برابر log2+log2 خواهد بود؛ بنابراین گرچه t و v در یک سطح از p هستند (دو کلیک) ولی v به p نزدیک‌تر است.



شکل ۱: مثالی از فاصله لگاریتمی در خزش [۹].

در FICA صفحات با فاصله‌ی لگاریتمی کمتر نسبت به ریشه، دارای اولویت بالاتری جهت آوردن هستند. در این الگوریتم فاصله‌ی صفحات طبق رابطه‌ی زیر بدست می‌آید [۹].

$$d_{jt+1} = (1-\alpha) \times d_{jt} + \alpha \times (\log(O(i)) + \gamma \times d_{it})$$
$$i \in B(j),\ 0 \le \gamma \le 1,\ 0 < \alpha \le 1 \quad (۲)$$

α نرخ یادگیری[۷]، d_{jt+1} فاصله‌ی صفحه‌ی j در زمان t+1 و d_{it} و d_{jt} به ترتیب نشان‌دهنده‌ی فاصله‌ی صفحه‌ی i و j در زمان t می‌باشند. در این محیط هدف کمینه کردن جمع توبیخ‌های دریافتی می‌باشد. نرخ یادگیری α با استفاده از رابطه ۳ محاسبه می‌شود که t زمان (شماره تکرار) است و ثابت β جهت تنظیم نرخ یادگیری استفاده می‌شود. آزمایش‌های انجام‌شده نشان می‌دهد که در صورت تنظیم مناسب α، سیستم بازدهی بالاتری خواهد داشت. در ابتدا به علت نداشتن مقدار فاصله‌ی هیچ‌کدام از صفحات، α مساوی یک بوده و با گذشت زمان به‌صورت نمایی-نزولی به صفر میل می‌کند [۱۱].

$$\alpha = e^{-\beta * t} \quad (۳)$$

در [۱۰] الگوریتم IECA[۸] پیشنهاد شده است که دو ضعف الگوریتم FICA را پوشش می‌دهد. این الگوریتم مبتنی بر FICA، Backlink

[4] On-line Page Importance Computation
[5] Reinforcement Learning
[6] Fast Intelligent Crawling Algorithm
[7] Learning Rate
[8] Intelligent Effective Crawling Algorithm

count و ویژگی‌هایی از Breadth-first می‌باشد و یک فرمول جدیدی از فاصله به شرح زیر تعریف می‌کند [۱۰]:

$$d_{jt+1} = (1- \delta_t) \times [(1-\alpha) \times d_{jt} + \alpha \times \min(\log(O(i)) + \gamma \times d_{it})] - \delta_t \quad (۴)$$

به‌عبارت‌دیگر:

$$d_{jt+1} = (1- \delta_t) \times (\text{the gained knowledge by crawler}) - (\text{the weight of each incoming link which depends on the percentage of crawled pages}) \quad (۵)$$

از δ_t برای برقراری تعادل بین دانش به‌دست آمده توسط عامل خزشگر و ویژگی ساختاری وب استفاده می‌شود. فاکتور بالانس به عامل خزشگر در سیاست انتخاب صفحه بخصوص در مراحل اولیه خزش کمک می‌کند؛ بنابراین در مراحل اولیه‌ی خزش که عامل هیچ پس‌زمینه‌ای راجع به ساختار وب ندارد $\alpha = 1$ و δ_t بالاترین مقدار خودش را دارد. با گذشت زمان عامل دانش بیشتری درباره‌ی محیط کسب کرده، بنابراین δ_t به‌صورت خطی کاهش می‌یابد به‌طوری‌که در مراحل پایانی خزش که عامل خزشگر تقریباً دانش کاملی از محیط به دست آورده δ_t به صفر می‌رسد [۱۰]. آزمایش‌ها نشان داده است که اگر مقدار اولیه δ_t در دامنه [0.35,0.45] قرار داشته باشد، الگوریتم FICA بهترین کارایی را خواهد داشت [۱۰]. فاکتور بالانس، δ_t به‌صورت زیر مدل می‌شود [۱۰]:

$$\delta_t = 0.39 \times \text{Percentage of crawled web pages} + 0.4 \quad (۶)$$

روش پیشنهادی از ایده الگوریتم رتبه‌بندی RL-Rank استفاده می‌کند و همچنین جهت ارزیابی کارایی آن از الگوریتم DistanceRank جهت مقایسه با روش پیشنهادی استفاده شده است؛ بنابراین بخش‌های بعد به توضیح مختصر این دو الگوریتم می‌پردازد.

۲-۱- الگوریتم RL-Rank

در این الگوریتم کاربر و یا همان surfer به‌عنوان عامل و صفحات وب به‌عنوان حالت‌های محیط در نظر گرفته می‌شوند. عامل (کاربر) در هر زمان با کلیک بر روی یکی از لینک‌های صفحه‌ای که در روی آن قرار دارد به حالت (صفحه) بعدی می‌رود. جایزه دریافتی در اثر هر گذر از صفحه‌ی j به صفحه‌ی p به‌صورت زیر تعریف می‌شود [۱۱]:

$$r_{jp} = \frac{1}{O(j)} \quad (۷)$$

تعداد پیوندهای خروجی صفحه‌ی j و یا به عبارتی درجه خروجی صفحه‌ی j می‌باشد. به عبارتی هر چه درجه خروجی صفحه‌ی مبدأ کمتر باشد جایزه بیشتری دریافت می‌کند. رتبه‌ی هر صفحه نیز به نوعی تابع ارزش هر صفحه می‌باشد که پس از دریافت جایزه، باید به‌روز شود. تابع ارزش امید مجموع جایزه‌های تخفیف یافته‌ی دریافتی است.

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \; 0 \le \gamma \le 1 \quad (۸)$$

γ فاکتور تخفیف، E_{π} نشان‌دهنده‌ی امید ریاضی و r_{t+k+1} جایزه‌ای است که عامل در طول گذر بین حالت‌ها دریافت می‌کند [۱۱]. در این الگوریتم ارزش (رتبه) صفحه برابر با امید ریاضی مجموع جایزه‌های تخفیف یافته‌ای است که عامل در طی حرکت بین صفحات تا رسیدن به صفحه‌ی مورد نظر جمع می‌نماید. رتبه‌ی هر صفحه بر مبنای تعریف فوق می‌تواند به‌صورت زیر محاسبه گردد [۱۱].

$$R_{t+1}(p) = \sum_{j \in B(p)} \left[\left(Prob(j) * \frac{1}{O(j)} \right) * \left(r_{jp} + (\gamma * R_t(j)) \right) \right] \quad (۹)$$

در این رابطه $R_{t+1}(p)$ رتبه‌ی صفحه‌ی p در زمان t+1 و $R_t(j)$ رتبه‌ی صفحه‌ی j در زمان t، prob(j) احتمال حضور عامل در صفحه‌ی j، O(j) درجه خروجی صفحه‌ی j و r_{jp} جایزه (تعریف‌شده در رابطه ۷) به ازای انتقال از صفحه‌ی j به صفحه‌ی p است. γ نیز فاکتور تخفیف می‌باشد که جهت تنظیم اثرات رتبه‌ی صفحات قبلی که در مسیر رسیدن به صفحه‌ی p قرار دارند، استفاده می‌شود. عبارت داخل اولین پرانتز در کروشه، احتمال رسیدن به صفحه‌ی p از صفحه‌ی j است این مقدار برابر است با احتمال حضور عامل در صفحه‌ی j، ضربدر احتمال انتخاب صفحه‌ی p هنگامی‌که عامل در j حضور دارد. ازآنجاکه عامل یکی از لینک‌ها را با توزیع احتمال یکنواخت انتخاب می‌کند، احتمال انتخاب صفحه‌ی p از j برابر است با یک تقسیم بر درجه خروجی صفحه‌ی j. عبارت داخل دومین پرانتز هم جمع جایزه‌ی آنی و جایزه‌ی تخفیف داده شده می‌باشد.

۲-۲- الگوریتم DistanceRank

در [۱۲] الگوریتمی مبتنی بر یادگیری تقویتی به نام DistanceRank معرفی شده است و از فاصله لگاریتمی میان صفحات استفاده کرده است. با توجه به آنکه اساس الگوریتم یادگیری تقویتی بر مبنای جایزه و توبیخ است در این روش فاصله لگاریتمی بین صفحات به‌عنوان توبیخ دریافتی استفاده می‌شود. هدف روش ارائه شده، کمینه کردن کل توبیخ‌های دریافتی است، در واقع صفحه‌ای که فاصله آن کمتر است رتبه‌ی بالاتری دارد.

الگوریتم DistanceRank از میانگین فاصله استفاده می‌کند. مطابق شکل ۲ اگر d_{ij} نشان‌دهنده‌ی فاصله‌ی صفحات میان i و j باشد، میانگین فاصله‌ی صفحه‌ی i به‌صورت زیر تعریف می‌شود [۱۲]:

$$d_j = \frac{\sum_{i=1}^{N} d_{ij}}{V} \quad (۱۰)$$

که منظور از V تعداد کل گره‌های (صفحات) وب می‌باشد. به عبارتی وزن یک پیوند به‌جای ۱، log(O(i)) می‌باشد، بنابراین اگر هیچ مسیری بین دو صفحه‌ی i و j نباشد، مقدار d_{ij} آن بسیار بزرگ می‌باشد. برای مثال اگر صفحه‌ی j فقط دارای یک پیوند ورودی i باشد، میانگین فاصله آن به‌صورت زیر به‌دست می‌آید [۱۲]:

$$d_j = \frac{\sum_{k=1}^{V} d_{kj}}{V} = \frac{\sum_{k \neq i}^{V} (d_{ki} + d_{ij}) + d_{ij}}{V} = \frac{\sum_{k \neq i}^{V} d_{ki}}{V} + d_{ij} = \frac{\sum_{k=1}^{V} d_{ki} - d_{ii}}{V} + d_{ij} \overset{\text{Eq (1)}}{\rightarrow} d_j = d_i - \frac{d_{ii}}{V} + d_{ij}$$
$$\approx d_i + d_{ij} = d_i + \log(O(i)) \quad (۱۱)$$

از آنجا که V خیلی بزرگ است، از $\frac{d_{ij}}{V}$ صرف‌نظر می‌شود؛ بنابراین فاصله صفحه‌ی j به‌صورت زیر محاسبه می‌شود [۱۲]:

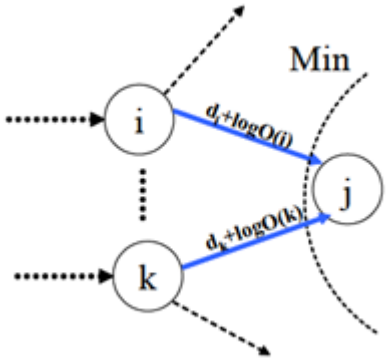
$$d_j = \min_i (d_i + \log(O(i))) \qquad i \in B(j) \quad (۱۲)$$

O(j) نشان‌دهنده‌ی تعداد خروجی‌های صفحه‌ی j می‌باشد، B(j) مجموعه صفحاتی است که به صفحه‌ی i اشاره دارد. برای کامل شدن رابطه‌ی بالا بر

مبنای یادگیری تقویتی، به‌صورت زیر میانگین فاصله محاسبه می‌شود [۱۲]:

$$d_{j_{t+1}} = (1-\alpha) * d_{j_t} + \alpha * \min_i(\log(O(i)) + \gamma * d_{i_t}) \quad (۱۳)$$

α نرخ یادگیری است (در ابتدا برابر یک است به این معنی که عامل کاربر هنوز هیچ دانشی از محیط ندارد)، γ ضریب نزول[9] (ضریب نزول برای تنظیم اثرات صفحات قبلی که در مسیر رسیدن به j قرار دارند، استفاده می‌شود). d_{j_t} و d_{i_t} به ترتیب نشان‌دهنده‌ی فاصله‌ی صفحه‌ی j و i در زمان t و $d_{j_{t+1}}$ فاصله‌ی صفحه‌ی j در زمان t+1 می‌باشد. O(i) نیز درجه خروجی صفحه‌ی i می‌باشد. رابطه‌ی فوق نشان می‌دهد که فاصله‌ی صفحه‌ی j در هر لحظه، به فاصله‌ی خود و ورودی‌هایش در لحظه‌ی قبلی بستگی دارد. به‌عبارتی کاربر در هر لحظه بر اساس آگاهی‌های قبلی و وضعیت فعلی محیط انتخاب خود را انجام می‌دهد.



شکل ۲: فاصله لگاریتمی صفحه‌ی j [۱۲].

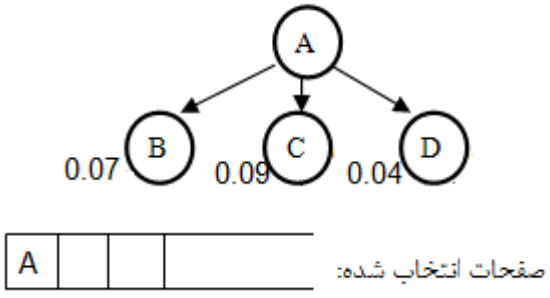
۳- روش پیشنهادی

روش‌های مبتنی بر اتصال از اتصال بین صفحات استفاده می‌کنند و به نوعی بر روی گراف وب عمل می‌کنند، بنابراین حساس به ویژگی‌های گراف‌اند. از آنجا که دو الگوریتم RL-Rank و PageRank نیز مبتنی بر اتصال هستند، به ویژگی‌های گراف مانند درجه ورودی خروجی صفحات وابسته‌اند، لذا این ایده به ذهن آمد که استفاده از مفهوم یادگیری تقویتی و تعریف جایزه شبیه به آنچه در RL-Rank آمده می‌تواند برای گراف‌های چگال مانند گراف وب فارسی مناسب باشد؛ بنابراین در این بخش روش خزش مبتنی بر یادگیری تقویتی که RL-Crawler نامیده می‌شود ارائه می‌گردد.

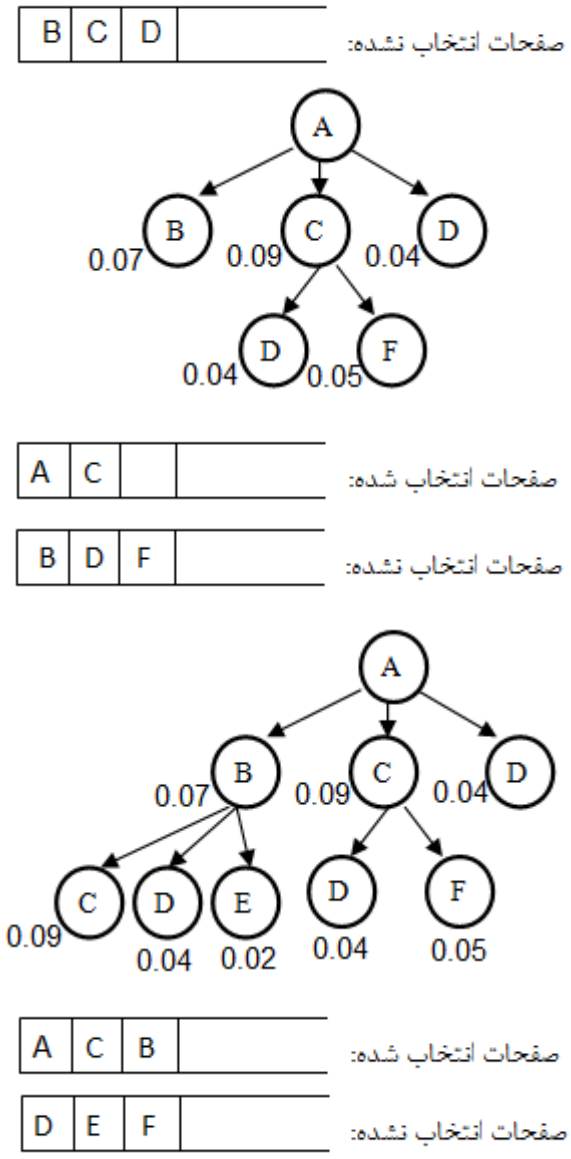
در روش پیشنهادی ابتدا رتبه‌ی هر صفحه‌ی موجود در گراف وب ایران طبق رابطه‌ی ۹ به‌دست می‌آید. سپس صفحه‌ای که بالاترین رتبه دارد به‌عنوان نقطه شروع فرآیند خزش در نظر گرفته می‌شود. تمام صفحاتی که به این صفحه پیوند دارند در لیستی ذخیره می‌شوند؛ لذا چالشی که به وجود می‌آید این است که کدام‌یک از صفحات فرزند جهت خزش انتخاب شوند و استراتژی انتخاب صفحه به چه صورت باشد.

مقدار ارزش صفحه براساس فرمول ارزش استفاده شده در روش RL-Rank (فرمول شماره ۹) محاسبه می‌شود. جهت انتخاب صفحه برای خزش در RL-Crawler از روش حریصانه استفاده شده است. روش حریصانه یکی از ساده‌ترین رویه‌ها برای انتخاب عمل توسط عامل در محیط یادگیری تقویتی است. در این روش عامل در هر مرحله عملی را انتخاب می‌کند که بالاترین ارزش تخمین زده شده را داشته باشد. این روش از دانش موجود برای به حداکثر رساندن پاداش استفاده می‌کند. همچنین اعمال دیگر که ممکن است بهتر باشند را در نظر نمی‌گیرد. از آنجایی که در روش حریصانه رسیدن به هدف در هر گام مستقل از گام قبلی و بعدی است و در هر مرحله برای رسیدن به هدف نهایی، مستقل از این‌که در مراحل قبلی چه انتخاب‌هایی صورت گرفته است، انتخاب انجام می‌پذیرد؛ بنابراین در روش حریصانه سیاست انتخاب صفحه به این صورت است که صفحات با بالاترین رتبه در ابتدا بارگذاری می‌شوند و از جمع‌آوری صفحات تکراری جلوگیری می‌شود.

برای روشن شدن موضوع نمونه‌ای از مراحل عملکرد خزشگر در شکل آورده شده است. اگر در شکل ۳ فرض شود صفحه‌ی A بالاترین رتبه را دارد، در لیست صفحات انتخاب شده قرار می‌گیرد و فرزندان آن (B، C و D) در لیست صفحات انتخاب نشده نگهداری می‌شوند. همانطور که در شکل مشخص شده است از بین فرزندان A، صفحه‌ی C بالاترین رتبه را دارد پس در مرحله‌ی بعد صفحه‌ی C جهت خزش انتخاب می‌شود و از لیست صفحات انتخاب نشده به لیست صفحات انتخاب شده انتقال می‌یابد. همچنین از بین فرزندان آن (D و F) فقط صفحه‌ی F به لیست صفحات انتخاب نشده اضافه می‌شوند چرا که صفحه‌ی D از قبل در لیست وجود دارد. همانطور که مشاهده می‌شود در مرحله‌ی سوم با توجه به لیست صفحات انتخاب نشده که در مرحله‌ی قبل بروز شد، صفحه‌ی B که از بین صفحات انتخاب نشده بالاترین رتبه را دارد، جهت خزش انتخاب می‌شود. همچنین صفحه‌ی E بعنوان فرزند صفحه‌ی B به لیست صفحات انتخاب نشده اضافه می‌شوند. این روند باعث می‌شود که در هر مرحله خزشگر به دنبال انتخاب صفحات با بالاترین رتبه باشد و همچنین از خزش صفحات تکراری جلوگیری کند. البته ناگفته نماند در الگوریتم پیشنهادی اگر خزشگر در یکی از مراحل کار خود به صفحه‌ای برسد که لینک خروجی نداشته باشد، صفحه‌ای را انتخاب می‌کند که در بین صفحات موجود در لیست صفحات انتخاب نشده بالاترین رتبه داشته باشد.



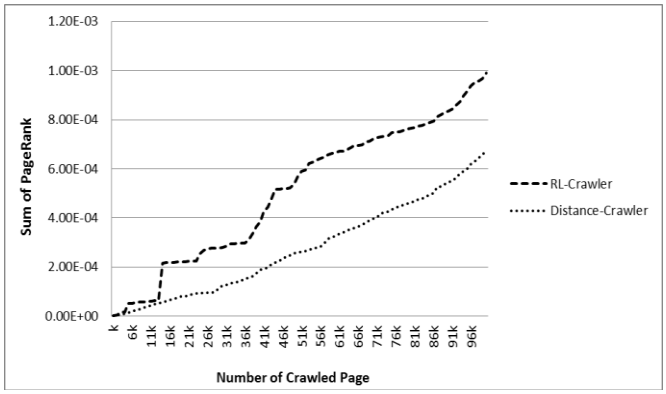
[9] Discount Factor



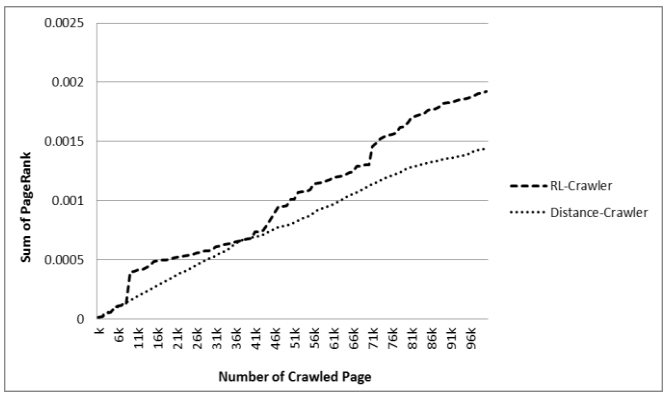
شکل ۳: نمونه‌ای از مراحل خزش RL-Crawler.

جهت ارزیابی RL_Crawler، الگوریتم DistanceRank روی مجموعه محک dotIR و تحت شرایط یکسان با روش پیشنهادی اعمال می‌گردد. از معیار PageRank برای مقایسه الگوریتم خزش پیشنهادی و الگویتم خزش مبتی بر DistanceRank استفاده شده است. همانطور که گفته شد در الگوریتم RL_Crawler در هر مرحله از خزش صفحه‌ای انتخاب می‌شود که بالاترین رتبه (RL-Rank) را داشته باشد، در حالی که موقع استفاده از الگوریتم DistanceRank برای خزش روی گراف وب ایران باید در هر مرحله صفحه‌ای انتخاب شود که کمترین DistanceRank را دارد. دلیل این امر آن است که الگوریتم DistanceRank براساس کمترین فاصله لگاریتمی تعریف شده است [۱۲].

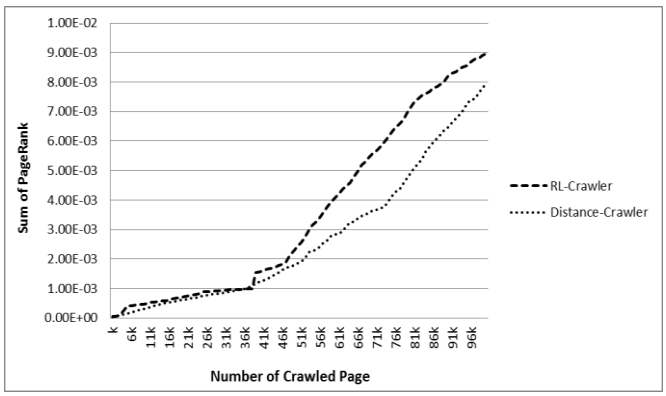
آزمایش‌ها با تعداد نقاط شروع متفاوت انجام شده است؛ بدین صورت که به ترتیب ۱، ۴ و ۸ صفحه‌ای که براساس الگوریتم Rl-Rank و همچنین DistanceRank بالاترین ارزش را داشتند به عنوان نقطه شروع فرآیند خزش در نظر گرفته شدند. نتایج در شکل‌های ۴، ۵ و ۶ نشان داده شده است. نمودار افقی در شکل‌ها نشان‌دهنده‌ی تعداد مراحل خزش (یا به‌عبارتی تعداد صفحات خزش شده) و نمودار عمودی بیانگر مجموع PageRank صفحاتی است که در هر مرحله خزش شده‌اند. در تمام آزمایش‌ها تا ۱۰۰ مرحله خزش انجام شده است.



شکل ۴: کارایی الگوریتم‌های خزش با سیاست انتخاب حریصانه و k=1.



شکل ۵: کارایی الگوریتم‌های خزش با سیاست انتخاب حریصانه و k=4.



شکل ۶: کارایی الگوریتم‌های خزش با سیاست انتخاب حریصانه و k=8.

لازم به ذکر است برای زمانیکه تعدا نقاط شروع فرآیند خزش بیش از ۱ باشد به تعداد نقاط شروع، خزشگر در نظر گرفته می‌شود که به صورت موازی فرآیند خزش را انجام دهند.

۴- نتیجه‌گیری و پیشنهادات

در این مقاله یک الگوریتم خزش با عنوان RL-Crawler پیشنهاد شد. این الگوریتم براساس سیاست حریصانه صفحاتی را برای خزش انتخاب می‌کند

که بالاترین RL-Rank را داشته باشند (صفحات داغ[10]) و از جمع‌آوری صفحات تکراری جلوگیری می‌کند. به منظور ارزیابی روش پیشنهادی الگویتم خزشی مبتنی بر DistanceRank پیاده‌سازی شد. نتایج آزمایشات حاکی از آن است که الگوریتم RL-Crawler کاراتر از الگوریتم خزشی است که بر مبنای الگوریتم DistanceRank می‌باشد چرا که صفحات با PageRank بالا را سریع‌تر جمع‌آوری می‌کند.

در آینده می‌توان الگوریتم پیشنهادی را روی داده‌های محک دیگر همچون LETOR اجرا و بررسی کرد. همچنین از آنجایی که الگوریتم PageRank براساس گراف می‌باشد و مبتنی بر محتوا نیست، می‌توان از راهکارهای دیگری جهت ارزش‌گذاری صفحات استفاده کرد.

مراجع

[۱] زارع بیدکی، علی‌محمد، **رتبه‌بندی و خزش مؤثر در وب**، دکتری، دانشگاه تهران، تهران، ۱۳۸۸.

[2] D. Lewandowski, "A three-year study on the freshness of web search engine databases," Journal of Information Science, pp. 1–17, January 2008.

[3] M. Najork and J. L. Wiener, "Breadth–First Search Crawling Yields High-Quality Pages,"10th International conference on World Wide Web, pp. 114–118, April 2001.

[4] H. Ali, "Effective Web Crawlers," Phd thesis, School Computer of Science and Information Technology, Science, Engineering, and Technology Portfolio, Melbourne, Victoria, Aus, March 2008.

[5] J. Cho, "Crawling the web," Phd thesis, Department of computer science, Stanford University, 2001.

[6] S. Pandey, C. Olstan, "User-Centric Web Crawling," 14th International Conference on World Wide Web, pp. 401-411, 2005.

[7] J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling Through URL Ordering," 7th International Conference on World Wide Web, Brisbane, Australia, pp. 14–18, April 2001.

[8] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive On-line Page importance computation," 12th International Conference on World Wide Web, pp. 280-290, 2003.

[9] A. M. ZarehBidoki, N. Yazdani, P. Ghodsnia, "A novel intelligent crawling algorithm based on reinforcement learning," Web Intelligence and Agent Systems (WIAS) Journal 7, pp. 363-373, 2009.

[10] V. Derhami, A. M. ZarehBidoki, A. M. Golshani, "A Novel Crawling Algorithm for Web Pages," LNCS 7097, pp. 263-272, 2011.

[11] V. Derhami, E. Khodadadian, M. Ghasemzadeh, A. M. Zareh Bidoki, "Applying reinforcement learning for web pages ranking algorithm," Apllied Soft Computting, pp. 1-7, 2013.

[12] A. M. ZarehBidoki, N. Yazdani, "DistanceRank: An Intelligent Ranking Algorithm for Web Pages," Information Processing and Management, pp. 877-892, 2008.

[10] Hot Pages