افزایش سرعت جستجو در مدل‌های مبتنی بر مجاورت

جواد پاک‌سیما، علی محمد زارع بیدکی و ولی درهمی

چکیده: یکی از اصلی‌ترین چالش‌های مدل‌های مبتنی بر مجاورت مسأله سرعت بازیابی اطلاعات می‌باشد. در مدل‌های مبتنی بر مجاورت مفهومی به نام فاصله تعریف می‌شود که برای محاسبه آن باید موقعیت کلمات پرس و جو در سند استخراج شود. این موضوع یعنی استخراج موقعیت‌ها و محاسبه فاصله‌ها فرایندی زمان‌بر است و چون غالباً در زمان جستجو اجرا می‌شود از دید کاربر اهمیت بیشتری دارد. در صورتی که بتوان تعداد اسناد مورد بررسی را کاهش داد بازیابی سریع‌تر می‌شود. در این مقاله الگوریتمی به نام SNTK۳ برای هرس‌کردن پویای اسناد در موقع جستجوی عبارت ارائه گردیده است. برای اجتناب از تخصیص بیش از حد حافظه و کاهش ریسک بروز خطا در موقع بازیابی، امتیاز تعدادی از اسناد بدون هیچ گونه هرسی محاسبه می‌شود (Skip-N). در این الگوریتم از سه هرم حداقل برای استخراج اسناد دارای بالاترین امتیازها استفاده شده و آزمایش‌ها نشان می‌دهد که استفاده از الگوریتم پیشنهادی باعث بهبود سرعت بازیابی می‌گردد.

کلیدواژه: **موتور جستجو، رتبه‌بندی، فاصله، مدل مجاورت، سرعت بازیابی.**

۱- مقدمه

یکی از پارامترهای مهم برای طراحان موتورهای جستجو مسأله کارایی است. یک موتور جستجو باید قادر باشد در هر ثانیه ده‌ها هزار پرس و جو را روی میلیاردها سند اجرا کند و این در حالی است که کاربر انتظار دارد در کسری از ثانیه پاسخ خود را بیابد [۱]. افزایش زمان پاسخ ممکن است کاربر را از ادامه کار منصرف کند و موتور دیگری را انتخاب کند [۲]، بنابراین باید زمان پاسخ به پرس و جو، محدود باشد. از طرفی محدودکردن زمان پاسخ باعث کاهش کیفیت اسناد بازیابی‌شده می‌گردد [۳].

روش‌های متعددی برای افزایش کارایی ارائه گردیده است. روش‌هایی مثل پردازش موازی، استفاده از حافظه پنهان[1] [۴]، فشرده‌سازی نمایه [۵] و هرس‌کردن ایستا و پویا برای افزایش بازدهی و کاهش زمان پاسخ ارائه گردیده است. در این مقاله هدف افزایش کارایی با استفاده از هرس‌کردن پویا است.

مدل‌های بازیابی اطلاعات به دو دسته تقسیم می‌شوند: مدل مبتنی بر استقلال کلمات و مدل مبتنی بر وابستگی کلمات. در مدل مبتنی بر استقلال کلمات حضور هر کلمه در سند یا پرس و جو مستقل از کلمات دیگر فرض می‌شود. در مدل مبتنی بر وابستگی کلمات حضور هر کلمه در سند یا پرس و جو وابسته به کلمات دیگر فرض می‌شود. این مدل در عمل واقعی‌تر از مدل قبلی است. در این مدل معمولاً برای کلمات پرس و جو که در سند ظاهر می‌شود مفهوم فاصله تعریف می‌شود و بر اساس فاصله‌های به دست آمده اسناد امتیاز دریافت می‌کنند و رتبه‌بندی می‌شوند.

یک عبارت[2]، لیستی از کلمات است که دارای ترتیب مشخصی هستند. به عنوان مثال "موتور جستجو" یک عبارت شامل دو کلمه است که کلمه اول آن "موتور" و کلمه دوم آن "جستجو" می‌باشد. طول اکثر عبارات دو کلمه می‌باشد و کمتر از یک درصد عبارات طولی بیشتر از شش کلمه دارند [۶].

روش اصلی برای جستجوی یک عبارت استفاده از نمایه معکوس[3] می‌باشد [۷]. هر ردیف در نمایه معکوس شامل یک کلمه است که لیستی از سه‌تایی‌های مرتب شامل شماره سند، تعداد رخداد کلمه در سند و موقعیت‌هایی که در آن کلمه مورد نظر در سند ظاهر شده است به صورت زیر می‌باشد

$$\langle d, tf_{d,t}, [p_1, p_2, \ldots] \rangle$$

که در آن d شناسه سندی است که شامل کلمه t می‌باشد، $tf_{d,t}$ تعداد رخداد t در سند d می‌باشد و p_i موقعیت t در سند را نشان می‌دهد. لیست زیر یک نمونه از این سه‌تایی‌ها است

$$\langle ۱۰۱, ۴, [۵, ۱۱, ۲۵, ۷۷] \rangle$$

با استفاده از لیست نمایه‌ی معکوس، زمان پردازش برای شمارش تعداد عبارت به تعداد کلمات پرس و جو و تعداد تکرار کلمات عبارت در سند وابسته می‌باشد. برای ارزیابی یک عبارت باید ابتدا موقعیت‌های کلمات پرس و جو استخراج شود و سپس با ترکیب آنها شمارش انجام شود.

در الگوریتم‌های رتبه‌بندی مبتنی بر مجاورت[4] مفهومی به نام فاصله[5] تعریف می‌شود که مشخص‌کننده میزان مجاورت کلمات پرس و جو در سند می‌باشد. تعاریف متعددی برای فاصله وجود دارد، مثلاً Keen برای اولین بار مفهوم فاصله بین دو کلمه را تعریف کرد. طبق تعریف او فاصله عبارت است از تعداد کلمات تطبیق داده نشده بین اولین و آخرین تطبیق در یک جمله [۸].

استفاده از لیست نمایه معکوس در جستجوی عبارت بسته به تعداد کلمات عبارت و فرکانس کلمات در اسناد ممکن است زمان‌بر باشد. یکی از راه‌های افزایش سرعت جستجوی عبارت ایجاد تغییرات در نمایه می‌باشد مثلاً ترکیبات یک کلمه در سند با کلمات اطرافش نمایه شود. این روش دارای دو اشکال است، یکی این که تا چند کلمه و تا چه فاصله‌ای را باید نمایه کرد و مشکل اساسی‌تر حجم نمایه است که ممکن است چندین برابر شود.

روش دیگر برای افزایش سرعت جستجوی عبارت آن است که میزان

این مقاله در تاریخ ۲۷ شهریور ماه ۱۳۹۵ دریافت و در تاریخ ۳ اردیبهشت ماه ۱۳۹۶ بازنگری شد.

جواد پاک‌سیما، دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، (email: paksima@stu.yazd.ac.ir).

علی‌محمد زارع بیدکی، دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، (email: alizareh@yazd.ac.ir).

ولی درهمی، دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، (email: vderhami@yazd.ac.ir).

1. Cache
2. Phrase
3. Inverted Index
4. Proximity
5. Distance

پردازش در زمان جستجو کاهش یابد. مشکل این روش معمولاً کاهش دقت[1] بازیابی است زیرا برای کاهش پردازش باید بخشی از پردازش را حذف کرد که ممکن است بخشی از اسناد مرتبط بازیابی نگردد. در این مقاله سعی شده است با ارائه الگوریتم‌هایی پردازش زمان جستجو کاهش یابد به طوری که تأثیر کمی روی کاهش دقت داشته باشند.

نحوه پیمایش نمایه نیز روی سرعت بازیابی تأثیرگذار است. روش‌های TAAT و DAAT دو روش اصلی برای پیمایش نمایه هستند [۹]. در TAAT امتیاز اسناد به صورت مرحله مرحله به ترتیب کلمات پرس و جو محاسبه می‌گردد. الگوریتم ابتدا کلمات پرس و جو را به صورت نزولی وزن آنها مثلاً idf مرتب می‌کند یعنی ابتدا کلماتی که در اسناد کمتری ظاهر گردیده است بررسی می‌شود. بعد از آن فقط اسنادی بررسی می‌شوند که در هرم قرار دارند.

در روش DAAT ترتیب پردازش بر اساس اسناد می‌باشد. در این روش در هر تکرار یک سند به ازای تمام کلمات پرس و جو پردازش می‌شود [۱۰].

هر کدام از این دو روش دارای مزایا و معایبی می‌باشد. TAAT به حافظه زیادی نیاز دارد در حالی که در DAAT فقط کافی است یک حافظه هرم به تعداد نتایج مورد نیاز به کاربر در نظر گرفته شود. در ضمن در برخی از مدل‌های بازیابی مثل مدل‌های مبتنی بر مجاورت امکان به کارگیری روش TAAT وجود ندارد، بنابراین اغلب موتورهای جستجو از روش DAAT استفاده می‌کنند [۱۱].

تکنیک هرس‌کردن پویا بر اساس هر کدام از روش‌های TAAT و DAAT قابل پیاده‌سازی می‌باشد. برای هرس‌کردن پویای مبتنی بر DAAT باید دو پارامتر مشخص شود: حداقل امتیازی که هر سند باید دارا باشد و حد بالایی امتیاز هر سند. با یک مقایسه ساده سندها دارای حد بالایی امتیاز کمتر از حداقل امتیاز هرس می‌شوند.

در یک موتور جستجوی تجاری به ازای یک پرس و جو ممکن است میلیون‌ها نتیجه پیدا شود در حالی که کاربر غالباً فقط چند نتیجه اول را مشاهده می‌کند [۱۲]. از این رو بهتر است تمرکز روی بازیابی K نتیجه اول باشد (Top-K). K سندی که دارای بالاترین امتیازها هستند در یک هرم حداقل[2] درج می‌شوند و نهایتاً تنها این هرم به صورت نزولی مرتب شده و به کاربر نشان داده می‌شود.

در یک دسته‌بندی دیگر، روش‌های هرس‌کردن به سه دسته تقسیم می‌شوند [۱۳]:

- Rank-safe: الگوریتم تضمین می‌کند که با هرس‌کردن رتبه‌بندی تغییر نمی‌یابد.
- Safe-up-to-rank-n: این روش تضمین می‌کند که ترتیب n نتیجه اول بر طبق رتبه‌بندی بدون هرس باشد. با توجه به این که در اکثر جستجوها کاربر نتیجه مورد نظر را در صفحه اول می‌یابد این روش می‌تواند کارایی خوبی داشته باشد.
- Approximate: این روش تضمینی برای یکسان‌بودن رتبه‌بندی به دست نمی‌دهد.

در این مقاله هدف این است به روشی کاملاً پویا با هرس‌کردن بخشی از نمایه به کارایی قابل قبولی برسیم. الگوریتم پیشنهادی Approximate است اما در عمل آزمایش‌ها نشان می‌دهد که می‌توان با تنظیم پارامترها به ۱۰- Safe-up-to-rank نزدیک شد.

در ادامه ابتدای مروری بر کارهای مرتبط انجام‌شده خواهیم داشت و سپس الگوریتم پیشنهادی ارائه خواهد شد. در پایان با تغییر پارامترهای الگوریتم، نتایج تجربی از نظر سرعت و میزان خطا مقایسه می‌شود.

۲- کارهای مرتبط

روش‌های متعددی برای افزایش کارایی موتورهای جستجو در بازیابی اطلاعات بر اساس مدل‌های مبتنی بر مجاورت ارائه گردیده است. هر کدام از این روش‌ها دارای مزایا و معایبی است. این روش‌ها به سه دسته کلی تقسیم می‌شوند. دسته اول روش‌هایی هستند که با تغییر در نمایه معکوس سعی در بهبود کارایی بازیابی اطلاعات بر اساس مدل‌های مبتنی بر مجاورت دارند. دسته دوم بدون تغییر نمایه و فقط با تغییراتی در فرایند جستجو اقدام به بهبود کارایی می‌کنند. دسته سوم از ترکیب دو روش قبلی استفاده می‌کنند.

بخشی از کارهای انجام‌شده، نمایه‌گذاری‌های متنوعی از زوج کلمات است [۱۴] تا [۱۶]. مشکل اساسی این روش‌ها حجم نمایه می‌باشد که برای تمام زوج کلمات خیلی زیاد است. البته همگی راهکارهای مختلفی برای کاهش حجم نمایه ارائه داده‌اند.

Wang و همکارانش برای افزایش کارایی مدلی چندگذره[3] ارائه کردند [۱۷]. آنها بر اساس ویژگی‌هایی خاص اسنادی را بازیابی می‌کردند و سپس داخل یک حلقه دامنه نتایج را محدود می‌کردند تا نتایج نهایی حاصل شود.

استفاده از حافظه نهان یکی دیگر از روش‌های افزایش کارایی می‌باشد. Baeza-Yates و همکارانش یک معماری دولایه را پیشنهاد کردند [۱۸]. در مدل آنها برای هر پرس و جو k نتیجه اول و لیست نمایه معکوس متناظر ذخیره می‌گردید. همچنین نشان داده شده که ذخیره‌سازی اشتراک لیست نمایه‌های معکوس نیز باعث افزایش کارایی می‌شود [۱۹].

تکنیک دیگر هرس‌کردن است که ممکن است به صورت ایستا با تغییر در ساختار نمایه انجام شود [۵] یا به صورت پویا در موقع جستجو انجام گردد.

در هرس‌کردن پویا تنها تعدادی از اسناد به طور کامل ارزیابی می‌شوند، تعدادی از اسناد به صورت جزئی ارزیابی می‌شوند و تعدادی از اسناد ارزیابی نمی‌شوند. در این تکنیک معمولاً از یک هرم حداقل برای ذخیره‌سازی نتایج مهم‌تر استفاده می‌گردد.

دو روش مشهور مبتنی بر هرس‌کردن پویا روش Max-Score [۲۰] و روش WAND [۲۱] می‌باشند که هر دو روش یک مقدار آستانه برای هرس تعریف می‌کنند. تفاوت اصلی دو روش در ترتیب پیمایش نمایه معکوس می‌باشد. هر دو روش مبتنی بر مدل Bag-of-words بوده و به طور مستقیم در مدل‌های مبتنی بر مجاورت قابل استفاده نیستند.

Strohman و همکارانش مکانیزم Max-Score را با حافظه پنهان ترکیب کردند تا زمان ارزیابی پرس و جو را بیشتر کاهش دهند [۱۰]. نویسندگان مدعی شده‌اند که الگوریتم پیشنهادی rank-safe است و سرعت جستجو را تا دو برابر افزایش می‌دهد.

Zhu و همکارانش تکنیک هرس‌کردن ایستا و پویا را ترکیب کردند [۲۲]. آنها نمایه زوج کلمات را بر اساس احتمال هم‌رخدادیشان هرس کردند و در زمان جستجو نیز هرس Top-K را انجام دادند.

به منظور افزایش کارایی در مدل‌های مبتنی بر مجاورت الگوریتم‌های متعددی بر اساس هرس ایستا و پویا ارائه گردیده است. در این الگوریتم‌ها

1. Precision
2. Min-Heap
3. Multi-Pass

```
Algorithm SNTK3
01. Input Query q , Documents d_1, d_2, ..., d_m
02. Output Heap TopDocs
03. Assumption
04.     q include ⟨t_1, t_2, ...⟩
05.     Heap_1 was used for TotalTF
06.     Heap_2 was used for MinTF
07. Begin Algorithm
08.     SkipCounter ← 0
09.     For i = 1 to m
10.         S_i ← 0
11.         If d_i includes all query terms
12.             SkipCounter ← SkipCounter + 1
13.             If SkipCounter < N
14.                 S_i ← Score(d_i, q)
15.             Else
16.                 TotalTF ← Σ_j TF(d_i, t_j)
17.                 MinTF ← min_j TF(d_i, t_j)
18.                 If TotalTF > min(Heap_1)
19.                     Insert (Heap_1, TotalTF)
20.                     S_i ← Score(d_i, q)
21.                 Else If MinTF ← min(Heap_2)
22.                     Insert (Heap_2, MinTF)
23.                     S_i ← Score(d_i, q)
24.                 End If
25.             End If
26.             If S_i > min(TopDocs)
27.                 Insert (TopDocs, S_i)
28.             End If
29.         End If
30.     End For
31: Return TopDocs
```

شکل ۱: الگوریتم هرس SNTK۳.

در بخش هرس پویا غالباً از روش DAAT استفاده شده است. در مدل‌های مبتنی بر وابستگی استفاده از روش TAAT عملاً ممکن نیست زیرا به حافظه فوق‌العاده زیادی برای نگهداری موقعیت کلمات پرس و جو در اسناد نیاز است [۲۳].

Broschar و همکارش با نمایه‌کردن انتخابی زوج کلمات هم‌رخداد در یک پنجره با طول مشخص کارایی را بهبود بخشیدند [۱۴]. آنها موقع جستجو اسنادی که در آن هم‌رخدادی کلمات از یک آستانه کمتر بود را هرس می‌کردند.

زمانی که از مکانیزم Max-Score در مدل‌های مبتنی بر مجاورت استفاده می‌شود باید پارامترهای مدل به گونه‌ای محاسبه گردد یا تخمین زده شود. مانع اصلی در این مکانیزم آن است که پارامترهای مدل زمانی قابل محاسبه است که موقعیت‌های کلمات در سند استخراج گردد. این محدودیت کارایی Max-Score را کاهش می‌دهد. Macdonald و همکارانش روش‌هایی برای تخمین پارامترهای مدل مجاورت ابداع کردند [۲۴] و [۲۵]. آنها حد بالایی تکرار زوج کلمات در سند را برابر با ماکسیمم تکرار هر یک از کلمات در سند در نظر گرفتند و با استفاده از آن حد بالایی امتیاز سند را تخمین زدند.

برخی تحقیقات نشان می‌دهد مکانیزم WAND در مدل‌های مبتنی بر مجاورت زیاد مؤثر نیست [۲۳] و [۲۶]. مشکل اصلی این مکانیزم در مدل‌های مبتنی بر مجاورت، مشخص‌کردن آستانه است. اگر این آستانه زیاد برآورد شود هرس کمی صورت می‌پذیرد و اگر کم در نظر گرفته شود احتمال بروز خطا و کاهش دقت بیشتر می‌شود.

۳- الگوریتم پیشنهادی

در الگوریتم پیشنهادی دو معیار برای هرس کردن یک سند در نظر گرفته شده است: معیار اول مجموع فرکانس کلمات پرس و جو در سند و معیار دوم حداقل فرکانس کلمات پرس و جو در سند.

Kim و همکارش نشان دادند که تعداد انطباق‌های یک عبارت حداکثر به تعداد کلماتی است که در سند پیدا شده است [۲۷]. اگر رابطه مستقیمی بین امتیاز سند و تعداد انطباق‌ها نیز وجود داشت می‌شد بر اساس مجموع فرکانس کلمات پرس و جو آستانه را مشخص کرد اما این رابطه وجود ندارد ولیکن معیار خوبی برای هرس‌کردن می‌باشد.

اگر معیار انتخاب top-K مجموع فرکانس کلمات پرس و جو در نظر گرفته شود باید K به حد کافی بزرگ در نظر گرفته شود تا سند خوبی به اشتباه هرس نشود. افزایش K باعث کاهش هرس می‌گردد و کاهش هرس افزایش سرعت جستجوی کمتری را به دنبال دارد و بنابراین انتخاب K اهمیت زیادی دارد.

اما برای استفاده از معیار دوم یعنی حداقل فرکانس کلمات، استدلال نظری قوی نداریم و دلیل استفاده از آن نتایج تجربی آن چه از نظر سرعت و چه از نظر دقت می‌باشد. این معیار در فرایند آزمایش معیارهای مختلف برای هرس، بر اثر یک خطای برنامه‌نویسی کشف شد. شاید بتوان توجیه ضعیفی بدین صورت داشت که حداقل فرکانس کلمات نشان‌دهنده تعداد انطباق‌های کامل عبارت مورد جستجو می‌باشد.

مسأله دیگری که در عمل به وجود آمد میزان مصرف بالای حافظه در نرخ بالای پرس و جو بود. در الگوریتم پیشنهادی به ازای هر پرس و جو دو یا چند Heap باید ایجاد می‌شد که در یک سیستم توزیع‌شده این عدد چندین برابر می‌شود. تخصیص شدید حافظه Garbage Collector را بیش از حد درگیر می‌کرد و بنابراین در عمل پس از چند دقیقه، فعالیت Garbage Collector باعث کاهش سرعت جستجو می‌گردید.

برای حل مشکل تخصیص و آزادسازی حافظه روشی به نام Skip-N ابداع شد. در این روش به ازای هر پرس و جو درج در هرم از ابتدا صورت نمی‌گرفت و در N سند اول بازیابی شده بدون هیچ پیش‌پردازشی جستجو انجام می‌شد. این روش باعث می‌شد برای همه پرس و جوها بلافاصله حافظه تخصیص نیابد.

در Skip-N برای پرس و جوهایی که تعداد نتایج کمتر از N باشد هیچ گونه هرسی انجام نمی‌شود یعنی برای این گونه پرس و جوها الگوریتم rank-safe است.

با توجه به این که در روش هرس Top-K نیز از هرم استفاده می‌شود، حداقل K نتیجه اول برای پرکردن هرم استفاده می‌گردد. بنابراین الگوریتم Skip-N برای پرس و جوهایی که در هر یک از نمایه‌ها حداکثر $K+N$ نتیجه دارد کاملاً rank-safe است. البته سرعت اجرای این گونه پرس و جوها نیز افزایش نمی‌یابد.

به طور خلاصه این الگوریتم SNTK۳ نام‌گذاری شده که SN مخف Skip-N و TK مخفف Top-K می‌باشد. عدد ۳ انتهایی نیز نشان‌گر سه هرمی است که در این الگوریتم استفاده شده است. دو هرم برای پیاده‌سازی دو معیار هرس می‌باشد. یک هرم هم مربوط به امتیاز نهایی اسناد است که برای اجتناب از مرتب‌سازی کامل اسناد مورد استفاده قرار می‌گیرد. اندازه هرم سوم به درخواست کاربر برای نمایش نتایج بستگی دارد. با توجه به این که در بیش از ۹۰٪ جستجوها، کاربر فقط صفحه اول را مشاهده می‌کند [۱۲] اندازه این هرم غالباً ده می‌باشد. شکل ۱ شبه‌کد مربوط به الگوریتم SNTK۳ را نشان می‌دهد.

در الگوریتم فوق از متغیر SkipCounter برای شمارش اسنادی که

قرار است بدون هیچ پردازشی در فرایند محاسبه امتیاز قرار بگیرند استفاده شده است (بخش Skip-N).

متغیر TotalTF دربردارنده مجموع فرکانس کلمات پرس و جو در سند جاری است و از آن برای هرس‌کردن مرحله اول استفاده می‌شود. عملکرد آن به این نحو است که برای هر سند با جمع‌کردن فرکانس کلمات پرس و جو در سند، TotalTF محاسبه می‌شود و سپس با مقدار حداقل هرم مربوطه ($Heap_1$) یعنی ریشه هرم مقایسه می‌گردد. اگر مقدار TotalTF از حداقل هرم کمتر بود سند هرس می‌شود و در غیر این صورت TotalTF در هرم درج می‌شود و همچنین امتیاز سند محاسبه می‌گردد.

به طور مشابه متغیر MinTF دربردارنده حداقل فرکانس کلمات پرس و جو در سند جاری است و از آن برای هرس‌کردن مرحله دوم استفاده شده است. عملکرد آن مشابه TotalTF می‌باشد یعنی مقدار آن با حداقل هرم مربوطه ($Heap_2$) مقایسه می‌گردد. اگر مقدار MinTF از حداقل هرم کمتر بود سند هرس می‌شود و در غیر این صورت MinTF در هرم درج می‌شود و همچنین امتیاز سند محاسبه می‌گردد.

هرم سوم TopDocs هست و از آن برای ذخیره‌سازی نتایج اول جستجو بسته به صفحه‌ای که توسط کاربر درخواست می‌شود استفاده شده است. برخلاف دو هرم قبلی، اندازه این هرم دقیقاً برابر تعداد نتایج خروجی است. استفاده از این هرم برای اجتناب از مرتب‌سازی نزولی نتایج نهایی جستجو است یعنی بالاترین امتیازها را دربردارد. این هرم برخلاف دو هرم قبل rank-safe است و در پایان به عنوان نتیجه تابع جستجو بازگردانده می‌شود.

تابع Score وظیفه محاسبه امتیاز سند جاری را بر عهده دارد و فرایند هرس در الگوریتم SNTK۳ مستقل از این تابع می‌باشد. این تابع زمان‌برترین بخش جستجو است زیرا در آن Posting Listهای کلمات پرس و جو از دیسک واکشی می‌شود. بنابراین تلاش می‌شود که این بخش کمتر فراخوانی شود. اصلی‌ترین وظیفه هرس کاهش فراخوانی تابع Score می‌باشد.

در هر صورت SNTK۳ یک الگوریتم Approximate هست و احتمال خطا دارد. برای کاهش خطا باید بخش Skip و اندازه $Heap_1$ و $Heap_2$ افزایش یابد که این افزایش‌ها سبب کاهش سرعت بازیابی می‌شود. بنابراین باید تعادلی بین سرعت و دقت برقرار گردد و با پذیرش درصدی از خطا، سرعت قابل قبولی حاصل شود [۳].

۴- تنظیم پارامترها و نتایج ارزیابی

به منظور ارزیابی الگوریتم SNTK۳ از لاگ موتور جستجوی پارسی‌جو به صورت تصادفی ۲۰۰ پرس و جو انتخاب گردید. این پرس و جوها قبل از اعمال الگوریتم SNTK۳ به موتور جستجوی پارسی‌جو داده شد و نتایج ذخیره گردید.

در الگوریتم SNTK۳ ترتیب نتایج جابه‌جا نخواهد شد زیرا یک سند یا بررسی می‌شود یا بررسی نمی‌شود. اصطلاحاً الگوریتم score-safe است [۲۸]. یعنی اگر بررسی شود امتیاز واقعی آن بازگردانده می‌شود و اگر سند بررسی نشود امتیازی نیز دریافت نمی‌کند و از لیست نتایج حذف می‌گردد. پس تغییرات نتایج جستجو پس از اجرای الگوریتم فقط ممکن است منجر به حذف برخی از نتایج شود.

در عمل سه پارامتر قابل تنظیم است:

- مقدار Skip اولیه: تعداد سندی که بدون هیچ پردازشی در فرایند محاسبه امتیاز وارد می‌شوند.
- اندازه $Heap_1$: اندازه هرم مجموع فرکانس کلمات پرس و جو در سند (TotalTF). برای سادگی و افزایش کارایی بیشتر فرض می‌شود هرم کامل است و بنابراین پارامتر توصیف‌کننده برای آن ارتفاع هرم می‌باشد.
- اندازه $Heap_2$: اندازه هرم حداقل فرکانس کلمات پرس و جو در سند (MinTF). به طور مشابه هرم کامل فرض می‌شود و بنابراین پارامتر توصیف‌کننده برای آن ارتفاع هرم می‌باشد.

البته اندازه هرم سوم (TopDocs) نیز می‌تواند به عنوان پارامتر در نظر گرفته شود. این هرم در واقع تعداد نتایج نمایش داده شده به کاربر را نشان می‌دهد و بنابراین در ارزیابی الگوریتم پیشنهادی اندازه هرم سوم مقدار ثابت ده فرض شده است.

خروجی الگوریتم SNTK۳ از دو جهت قابل ارزیابی است. یکی از جنبه کارایی الگوریتم که به سادگی با مقایسه زمان پاسخ پرس و جوها در حالت فعال‌بودن الگوریتم SNTK۳ و فعال‌نبودن آن قابل ارزیابی است. جنبه دوم ارزیابی که مهم‌تر هم هست حذف برخی از نتایج به خصوص از صفحه اول می‌باشد. در مورد صفحات دوم به بعد می‌توان الگوریتم SNTK۳ را غیر فعال کرد زیرا درصد کمی از پرس و جوها در صفحات دوم به بعد دنبال می‌شود [۱۲].

به منظور ارزیابی الگوریتم پیشنهادی و انتخاب بهترین پارامترها باید معیاری برای میزان خطا مشخص کرد. با توجه به این که کاربران غالباً نتایج را از بالا به پایین دنبال می‌کنند نتایج بالا اهمیت بیشتری دارد. همچنین در صفحه اول اهمیت نتایج اول صفحه اهمیت فوق‌العاده‌ای نسبت به نتایج آخر صفحه دارد و بنابراین تابع خطی پاسخ‌گو نیست و به نظر تابع نمایی معکوس بهتر است.

به عنوان نمونه می‌توان تابع زیر را برای ارزیابی خطای حاصل از الگوریتم استفاده کرد

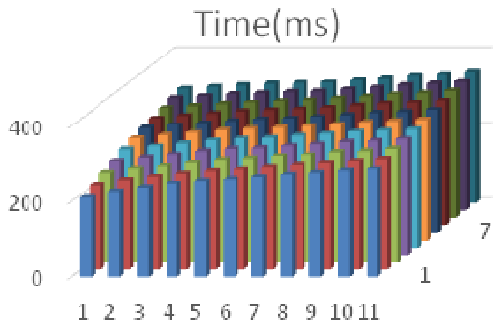
$$E(Q)=\frac{1}{|Q|}\sum_{q\in Q}\sum_{i=1}^{10}\frac{omitted(i)}{2^i} \qquad (1)$$

زمانی که به ازای پرس و جوی q ردیف iام حذف شده باشد تابع $omitted(i)$ برابر یک می‌باشد و در غیر این صورت این تابع صفر است. Q مجموعه پرس و جوها است و $|Q|$ تعداد پرس و جوها را نشان می‌دهد. $E(Q)$ عددی بین صفر و یک خواهد بود. اگر هیچ نتیجه‌ای حذف نگردد $E(Q)$ صفر خواهد بود و اگر تمام ده نتیجه اول حذف گردد $E(Q)$ تقریباً یک است (مقدار دقیق $E(Q)$ در موقع حذف ده نتیجه اول مقدار $1-(1/2^{10})$ می‌باشد).

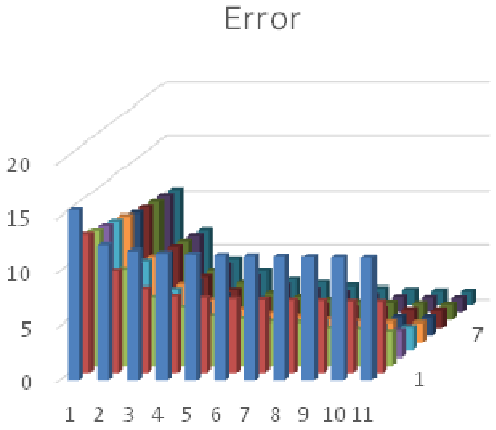
در (۱) اهمیت حذف هر ردیف نسبت به ردیف قبل دو برابر کاهش می‌یابد. یعنی اگر در یک پرس و جو ردیف اول حذف شود این معادل آن است که برای دو پرس و جو ردیف‌های دوم حذف گردد یا برای چهار پرس و جو ردیف سوم حذف گردد. هدف انتخاب پارامترها به گونه‌ای است که $E(Q)$ کمینه باشد.

اگر تنها کمینه‌کردن $E(Q)$ مد نظر قرار گیرد آن گاه بدون هیچ تلاشی نتیجه مشخص است. کافی است مقدار N در Skip-N یا مقدار K در Top-K به اندازه کافی بزرگ فرض شود. در این صورت هیچ هرسی صورت نمی‌پذیرد و الگوریتم از حیز انتفاع ساقط است. بنابراین باید برای $E(Q)$ یک حداکثر تعریف شود تا تنظیم پارامترها مفهوم یابد.

نکته مهم دیگر میزان حافظه مصرفی است. الگوریتم SNTK۳ به منظور هرس برای هر پرس و جو از سه هرم استفاده می‌کند. هرم سوم هرم کوچکی است و حجم حافظه مصرفی آن قابل اغماض است اما نمی‌توان هرم اول و دوم را خیلی کوچک در نظر گرفت زیرا کاهش اندازه



شکل ۲: رابطه بین متوسط زمان جستجو و اندازه $Heap_1$ و $Heap_2$.



شکل ۳: رابطه بین میزان خطا و اندازه $Heap_1$ و $Heap_2$.

هر یک از این دو هرم باعث افزایش خطا می‌شود. همچنین افزایش اندازه هر یک از این هرم‌ها علاوه بر این که میزان هرس را کاهش می‌دهد مصرف حافظه بیشتر را به دنبال دارد. نمودار شکل ۲ رابطه بین متوسط زمان جستجو و اندازه $Heap_1$ و $Heap_2$ و نمودار شکل ۳ رابطه بین میزان خطا و اندازه $Heap_1$ و $Heap_2$ را نشان می‌دهد.

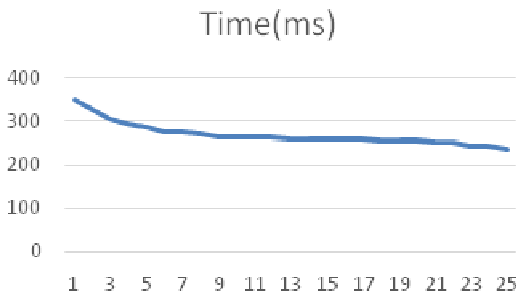
نمودار شکل ۴ رابطه بین متوسط زمان جستجو و حداکثر خطای قابل قبول را نشان می‌دهد. این نمودار مشخص می‌کند که بین کارایی و دقت تضاد وجود دارد.

جدول ۱ بهترین پارامترها بر اساس درصد خطای قابل پذیرش را نشان می‌دهد.

۵- نتیجه‌گیری و کارهای آینده

در این مقاله یکی از موضوعات موتورهای جستجو یعنی افزایش کارایی در جستجوی عبارت بدون علامت نقل قول بررسی گردید. یکی از روش‌های افزایش سرعت جستجو هرس‌کردن پویاست که مهم‌ترین تکنیک در این روش تکنیک Top-K می‌باشد. در الگوریتم پیشنهادی در این مقاله (SNTK۳) از سه هرم برای استخراج اسنادی که با احتمال بیشتری بالاترین امتیازها را دارند استفاده شده است. همچنین برای اجتناب از تخصیص بیش از حد حافظه و کاهش ریسک بروز خطا، امتیاز تعدادی از اسناد بدون هیچ گونه هرسی محاسبه گردید (Skip-N).

به عنوان چشم‌انداز آینده می‌توان پارامترهای بیشتری را برای هرس‌کردن آزمایش کرد و همچنین می‌توان با استفاده از روش‌های پیش‌بینی زمان اجرای پرس و جوها [۲۹] اندازه هرم‌ها و مقدار Skip-N را به صورت پویا مشخص نمود.



شکل ۴: رابطه بین متوسط زمان جستجو و حداکثر خطای قابل قبول.

جدول ۱: بهترین پارامترهای الگوریتم SNTK۳ بر اساس درصد خطای قابل قبول.

Error	N	K_1	K_2	Time (q/s)
٪۱	۲۵۰	۶۴	۲۵۶	۲۸۸
٪۲	۲۰۰	۳۲	۶۴	۲۶۷
٪۳	۲۰۰	۳۲	۱۶	۲۵۲
٪۴	۲۰۰	۱۶	۱۶	۲۴۵
٪۵	۱۵۰	۱۶	۱۶	۲۳۹
٪۱۰	۱۷۵	۸	۸	۲۱۰
٪۱۵	۵۰	۴	۲	۱۹۹

مراجع

[1] S. Tatikonda, B. B. Cambazoglu, and F. P. Junqueira, "Posting list intersection on multicore architectures," in *Proc. of the 34th International ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 963-972, Jul. 2011.

[2] J. Teevan, K. Collins-Thompson, R. W. White, S. T. Dumais, and Y. Kim, "Slow search: information retrieval without time constraints," in *Proc. of the Symp. on Human-Computer Interaction and Information Retrieval*, Article 1, Oct. 2013.

[3] B. B. Cambazoglu, F. P. Junqueira, V. Plachouras, S. Banachowski, B. Cui, S. Lim, and B. Bridge, "A refreshing perspective of search engine caching," in *Proc. of the 19th Int.Conf. on World Wide Web*, pp. 181-190, Apr. 2010.

[4] Q. Gan and T. Suel, "Improved techniques for result caching in web search engines," in *Proc. of the 18th Int. Conf. on World Wide Web*, pp. 431-440, Apr. 2009.

[5] R. Blanco Gonzalez, *Index Compression for Information Retrieval Systems*, Ph.D. Thesis, University of A Coruna, Spain, 2008.

[6] D. Bahle, H. Williams, and J. Zobel, "Compaction techniques for nextword indexes," in *Proc. Int. Symp. on String Processing and Information Retrieval,* p. 33, Nov. 2001.

[7] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Morgan Kaufmann, 1999.

[8] E. M. Keen, "The use of term position devices in ranked output experiments," *Journal of Documentation*, vol. 47, no. 1, pp. 1-22, Mar. 1991.

[9] K. Jiang and Y. Yang, "Efficient dynamic pruning on largest scores first (LSF) retrieval," *Front. Inf. Technol. Electron. Eng.*, vol. 17, pp. 1-14, Jan. 2016.

[10] T. Strohman, H. Turtle, and W. B. Croft, "Optimization strategies for complex queries," in *Proc. of the 28th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 219-225, Aug. 2005.

[11] D. Carmel and E. Amitay, Juru at TREC 2006: TAAT versus DAAT in the Terabyte Track, in TREC, 2006.

[12] A. Chuklin, P. Serdyukov, and M. De Rijke, "Modeling clicks beyond the first result page," in *Proc. of the 22nd ACM Int. Conf. on Information & Knowledge Management*, pp. 1217-1220, Oct. 2013.

[13] H. Turtle and J. Flood, "Query evaluation: strategies and optimizations," *Inf. Process. Manag.*, vol. 31, no. 6, pp. 831-850, Nov. 1995.

[14] A. Broschart and R. Schenkel, "High-performance processing of text queries with tunable pruned term and term pair indexes," *ACM Trans. Inf. Syst.*, vol. 30, no. 1, p. 5, Feb. 2012.

[15] R. Schenkel, A. Broschart, S. Hwang, M. Theobald, and G. Weikum, "Efficient text proximity search," in *Proc. 14th Int. Conf. on String Processing and Information Retrieval*, pp. 287-299, Oct. 2007.
[16] H. Yan, S. Shi, F. Zhang, T. Suel, and J. R. Wen, "Efficient term proximity search with term-pair indexes," in *Proc. of the 19th ACM Int. Conf. on Information and Knowledge Management*, pp. 1229-1238, ???. 2010.
[17] L. Wang, J. Lin, and D. Metzler, "Learning to efficiently rank," in *Proc. of the 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 138-145, Oct. 2010.
[18] R. Baeza-Yate, F. Junqueira, V. Plachouras, and H. F. Witschel, "Admission policies for caches of search engine results," in *Proc. 14th Int. Conf. on String Processing and Information Retrieval*, vol. pp. 74-85, Jul. 2007.
[19] R. Ozcan, I. S. Altingovde, B. B. Cambazoglu, F. P. Junqueira, and O. Ulusoy, "A five-level static cache architecture for web search engines," *Inf. Process. Manag.*, vol. 48, no. 5, pp. 828-840, Sept. 2012.
[20] S. Jonassen, "Improving dynamic index pruning via linear programming," in *Proc. of the 2015 Workshop on Large-Scale and Distributed System for Information Retrieval*, pp. 21-25, Oct. 2015.
[21] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien, "Efficient query evaluation using a two-level retrieval process," in *Proc. of the Twelfth Int. Conf. on Information and Knowledge Managemen,t CIKM'03*, p. 426, Nov. 2003.
[22] M. Zhu, S. Shi, N. Yu, and J. R. Wen, "Can phrase indexing help to process non-phrase queries?," in *Proc. of the 17th ACM Conf. on Information and Knowledge Management*, vol. ???, pp. 679-688, ???. 2008.
[23] S. Huston, *Indexing Proximity-Based Dependencies for Information Retrieva*l, Ph.D. Dissertation, University of Massachusetts, 2014.
[24] C. Macdonald, I. Ounis, and N. Tonellotto, "Upper-bound approximations for dynamic pruning," *ACM Trans. Inf. Syst.*, vol. 29, no. 4, p. 17, Oct. 2011.
[25] C. Macdonald, N. Tonellotto, and I. Ounis, "On upper bounds for dynamic pruning," in *Proc. Conf. on the Theory of Information Retrieval*, vol. 6931, pp. 313-317, 2011.
[26] N. Tonellotto, C. Macdonald, and I. Ounis, "Efficient dynamic pruning with proximity support," in *Proc. of the 8th Workshop on Large-Scale Distributed Systems for Information Retrieval*, pp. 31-35, Geneva, Switzerland, Jul. 2010.
[27] S. R. Kim and J. Hong, "An efficient and practical algorithm for the many-keyword proximity problem by offsets," in *Proc. Int. Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, vol. 3642, pp. 477-483, 2005.
[28] M. Petri, J. S. Culpepper, and A. Moffat, "Exploring the magic of WAND," in *Proc. of the 18th Australasian Document Computing Symp.*, pp. 58-65, Oct. 2013.
[29] O. Rojas, V. Gil-Costa, and M. Marin, "Running time prediction for web search queries," in *Proc. Parallel Processing and Applied Mathematics*, pp. 210-220, 2016.

جواد پاک سیما در سال ۱۳۷۵ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه صنعتی شریف و مقطع کارشناسی ارشد مهندسی کامپیوتر خود را در سال ۱۳۷۷ از دانشگاه صنعتی شریف به پایان رسانده است. او در سال ۱۳۸۷ به عنوان هیأت علمی دانشگاه پیام نور یزد مشغول به فعالیت گردید و درسال ۱۳۹۱ به عنوان دانشجوی دکتری در رشته مهندسی کامپیوتر دانشگاه یزد مشغول به تحصیل گردید که هم‌اکنون نیز ادامه دارد.

علی‌محمد زارع بیدکی در سال ۱۳۷۸ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه صنعتی اصفهان و مقاطع کارشناسی ارشد و دکتری مهندسی کامپیوتر خود را به‌ترتیب در سال‌های ۱۳۸۱ و ۱۳۸۸ از دانشگاه تهران به پایان رسانده است. دکتر زارع بیدکی از سال ۱۳۸۸ در دانشکده مهندسی برق و کامپیوتر دانشگاه یزد مشغول به فعالیت گردید و اینک نیز عضو هیأت علمی این دانشگاه می‌باشد. زمینه‌های علمی مورد علاقه نامبرده شامل موضوعاتی مانند موتورهای جستجو، رتبه بندی صفحات وب، خزش و پردازش داده‌های با حجم بالا می‌باشد.

ولی درهمی دانشیار گروه مهندسی کامپیوتر دانشگاه یزد می‌باشد. او مدرک لیسانس مهندسی برق-کنترل خود را در سال ۱۳۷۵ از دانشگاه صنعتی اصفهان دریافت نمود. سپس موفق به دریافت مدرک فوق لیسانس و دکترای مهندسی برق- کنترل از دانشگاه تربیت مدرس به‌ترتیب در سال ۱۳۷۷ و ۱۳۸۶شد. وی از سال ۱۳۸۶ بعنوان عضو هیئت علمی گروه مهندسی کامپیوتر در دانشگاه یزد فعالیت دارد. دکتر درهمی تاکنون ۱۹ مقاله در مجلات معتبر علمی- پژوهشی و بیش از ۶۰ مقاله در کنفرانس‌های ملی و بین المللی چاپ نموده است. زمینه‌های تحقیقاتی وی شامل: سیستم‌های فازی، کنترل هوشمند، یادگیری تقویتی، رباتیک، موتورهای جستجو، و فناوری اطلاعات می‌باشد.